"Why Should I Trust You?" Explaining the Predictions of Any Classifier

Marco Tulio Ribeiro
University of Washington
Seattle, WA 98105, USA
marcotcr@cs.uw.edu

Sameer Singh
University of Washington
Seattle, WA 98105, USA
sameer@cs.uw.edu

Carlos Guestrin
University of Washington
Seattle, WA 98105, USA
guestrin@cs.uw.edu

ABSTRACT

Despite widespread adoption, machine learning models remain mostly black boxes. Understanding the reasons behind predictions is, however, quite important in assessing *trust*, which is fundamental if one plans to take action based on a prediction, or when choosing whether to deploy a new model. Such understanding also provides insights into the model, which can be used to transform an untrustworthy model or prediction into a trustworthy one.

In this work, we propose LIME, a novel explanation technique that explains the predictions of *any* classifier in an interpretable and faithful manner, by learning an interpretable model locally around the prediction. We also propose a method to explain models by presenting representative individual predictions and their explanations in a non-redundant way, framing the task as a submodular optimization problem. We demonstrate the flexibility of these methods by explaining different models for text (e.g. random forests) and image classification (e.g. neural networks). We show the utility of explanations via novel experiments, both simulated and with human subjects, on various scenarios that require trust: deciding if one should trust a prediction, choosing between models, improving an untrustworthy classifier, and identifying why a classifier should not be trusted.

1. INTRODUCTION

Machine learning is at the core of many recent advances in science and technology. Unfortunately, the important role of humans is an oft-overlooked aspect in the field. Whether humans are directly using machine learning classifiers as tools, or are deploying models within other products, a vital concern remains: *if the users do not trust a model or a prediction, they will not use it.* It is important to differentiate between two different (but related) definitions of trust: (1) *trusting a prediction*, i.e. whether a user trusts an individual prediction sufficiently to take some action based on it, and (2) *trusting a model*, i.e. whether the user trusts a model to behave in reasonable ways if deployed. Both are directly impacted by how much the human understands a model's behaviour, as opposed to seeing it as a black box.

Determining trust in individual predictions is an important problem when the model is used for decision making. When using machine learning for medical diagnosis [6] or terrorism detection, for example, predictions cannot be acted upon on blind faith, as the consequences may be catastrophic.

Apart from trusting individual predictions, there is also a need to evaluate the model as a whole before deploying it "in the wild". To make this decision, users need to be confident that the model will perform well on real-world data, according to the metrics of interest. Currently, models are evaluated using accuracy metrics on an available validation dataset. However, real-world data is often significantly different, and further, the evaluation metric may not be indicative of the product's goal. Inspecting individual predictions and their explanations is a worthwhile solution, in addition to such metrics. In this case, it is important to aid users by suggesting which instances to inspect, especially for large datasets.

In this paper, we propose providing explanations for individual predictions as a solution to the "trusting a prediction" problem, and selecting multiple such predictions (and explanations) as a solution to the "trusting the model" problem. Our main contributions are summarized as follows.

- LIME, an algorithm that can explain the predictions of *any* classifier or regressor in a faithful way, by approximating it locally with an interpretable model.
- SP-LIME, a method that selects a set of representative instances with explanations to address the "trusting the model" problem, via submodular optimization.
- Comprehensive evaluation with simulated and human subjects, where we measure the impact of explanations on trust and associated tasks. In our experiments, non-experts using LIME are able to pick which classifier from a pair generalizes better in the real world. Further, they are able to greatly improve an untrustworthy classifier trained on 20 newsgroups, by doing feature engineering using LIME. We also show how understanding the predictions of a neural network on images helps practitioners know when and why they should not trust a model.

2. THE CASE FOR EXPLANATIONS

By "explaining a prediction", we mean presenting textual or visual artifacts that provide qualitative understanding of the relationship between the instance's components (e.g. words in text, patches in an image) and the model's prediction. We argue that explaining predictions is an important aspect in

KDD 2016 San Francisco, CA, USA

ISBN 978-1-4503-4232-2/16/08…$15.00

DOI: http://dx.doi.org/10.1145/2939672.2939778

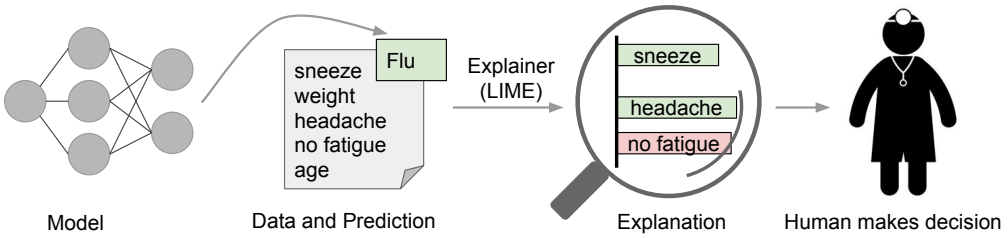


Figure 1: Explaining individual predictions. A model predicts that a patient has the flu, and LIME highlights the symptoms in the patient's history that led to the prediction. Sneeze and headache are portrayed as contributing to the "flu" prediction, while "no fatigue" is evidence against it. With these, a doctor can make an informed decision about whether to trust the model's prediction.

getting humans to trust and use machine learning effectively, if the explanations are faithful and intelligible.

The process of explaining individual predictions is illustrated in Figure 1. It is clear that a doctor is much better positioned to make a decision with the help of a model if intelligible explanations are provided. In this case, an explanation is a small list of symptoms with relative weights – symptoms that either contribute to the prediction (in green) or are evidence against it (in red). Humans usually have prior knowledge about the application domain, which they can use to accept (trust) or reject a prediction if they understand the reasoning behind it. It has been observed, for example, that providing explanations can increase the acceptance of movie recommendations [12] and other automated systems [8].

Every machine learning application also requires a certain measure of overall trust in the model. Development and evaluation of a classification model often consists of collecting annotated data, of which a held-out subset is used for automated evaluation. Although this is a useful pipeline for many applications, evaluation on validation data may not correspond to performance "in the wild", as practitioners often overestimate the accuracy of their models [20], and thus trust cannot rely solely on it. Looking at examples offers an alternative method to assess truth in the model, especially if the examples are explained. We thus propose explaining several representative individual predictions of a model as a way to provide a global understanding.

There are several ways a model or its evaluation can go wrong. Data leakage, for example, defined as the unintentional leakage of signal into the training (and validation) data that would not appear when deployed [14], potentially increases accuracy. A challenging example cited by Kaufman et al. [14] is one where the patient ID was found to be heavily correlated with the target class in the training and validation data. This issue would be incredibly challenging to identify just by observing the predictions and the raw data, but much easier if explanations such as the one in Figure 1 are provided, as patient ID would be listed as an explanation for predictions. Another particularly hard to detect problem is dataset shift [5], where training data is different than test data (we give an example in the famous 20 newsgroups dataset later on). The insights given by explanations are particularly helpful in identifying what must be done to convert an untrustworthy model into a trustworthy one – for example, removing leaked data or changing the training data to avoid dataset shift.

Machine learning practitioners often have to select a model from a number of alternatives, requiring them to assess the relative trust between two or more models. In Figure

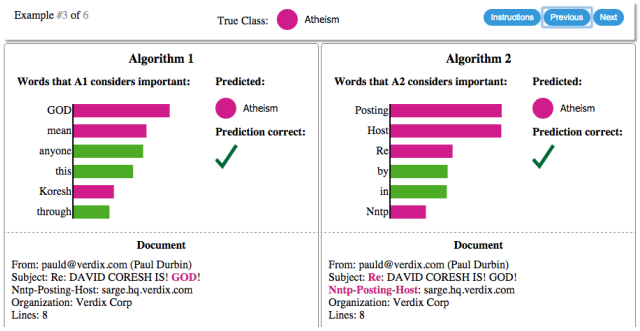


Figure 2: Explaining individual predictions of competing classifiers trying to determine if a document is about "Christianity" or "Atheism". The bar chart represents the importance given to the most relevant words, also highlighted in the text. Color indicates which class the word contributes to (green for "Christianity", magenta for "Atheism").

2, we show how individual prediction explanations can be used to select between models, in conjunction with accuracy. In this case, the algorithm with higher accuracy on the validation set is actually much worse, a fact that is easy to see when explanations are provided (again, due to human prior knowledge), but hard otherwise. Further, there is frequently a mismatch between the metrics that we can compute and optimize (e.g. accuracy) and the actual metrics of interest such as user engagement and retention. While we may not be able to measure such metrics, we have knowledge about how certain model behaviors can influence them. Therefore, a practitioner may wish to choose a less accurate model for content recommendation that does not place high importance in features related to "clickbait" articles (which may hurt user retention), even if exploiting such features increases the accuracy of the model in cross validation. We note that explanations are particularly useful in these (and other) scenarios if a method can produce them for *any* model, so that a variety of models can be compared.

Desired Characteristics for Explainers

We now outline a number of desired characteristics from explanation methods.

An essential criterion for explanations is that they must be **interpretable**, i.e., provide qualitative understanding between the input variables and the response. We note that interpretability must take into account the user's limitations. Thus, a linear model [24], a gradient vector [2] or an additive model [6] may or may not be interpretable. For example, if

hundreds or thousands of features significantly contribute to a prediction, it is not reasonable to expect any user to comprehend why the prediction was made, even if individual weights can be inspected. This requirement further implies that explanations should be easy to understand, which is not necessarily true of the features used by the model, and thus the "input variables" in the explanations may need to be different than the features. Finally, we note that the notion of interpretability also depends on the target audience. Machine learning practitioners may be able to interpret small Bayesian networks, but laymen may be more comfortable with a small number of weighted features as an explanation.

Another essential criterion is **local fidelity**. Although it is often impossible for an explanation to be completely faithful unless it is the complete description of the model itself, for an explanation to be meaningful it must at least be *locally faithful*, i.e. it must correspond to how the model behaves in the vicinity of the instance being predicted. We note that local fidelity does not imply global fidelity: features that are globally important may not be important in the local context, and vice versa. While global fidelity would imply local fidelity, identifying globally faithful explanations that are interpretable remains a challenge for complex models.

While there are models that are inherently interpretable [6, 17, 26, 27], an explainer should be able to explain *any* model, and thus be **model-agnostic** (i.e. treat the original model as a black box). Apart from the fact that many state-of-the-art classifiers are not currently interpretable, this also provides flexibility to explain future classifiers.

In addition to explaining predictions, providing a **global perspective** is important to ascertain trust in the model. As mentioned before, accuracy may often not be a suitable metric to evaluate the model, and thus we want to *explain the model*. Building upon the explanations for individual predictions, we select a few explanations to present to the user, such that they are representative of the model.

3. LOCAL INTERPRETABLE MODEL-AGNOSTIC EXPLANATIONS

We now present Local Interpretable Model-agnostic Explanations (**LIME**). The overall goal of LIME is to identify an **interpretable** model over the *interpretable representation* that is **locally faithful** to the classifier.

3.1 Interpretable Data Representations

Before we present the explanation system, it is important to distinguish between features and interpretable data representations. As mentioned before, **interpretable** explanations need to use a representation that is understandable to humans, regardless of the actual features used by the model. For example, a possible *interpretable representation* for text classification is a binary vector indicating the presence or absence of a word, even though the classifier may use more complex (and incomprehensible) features such as word embeddings. Likewise for image classification, an *interpretable representation* may be a binary vector indicating the "presence" or "absence" of a contiguous patch of similar pixels (a super-pixel), while the classifier may represent the image as a tensor with three color channels per pixel. We denote $x \in \mathbb{R}^d$ be the original representation of an instance being explained, and we use $x' \in \{0,1\}^{d'}$ to denote a binary vector for its interpretable representation.

3.2 Fidelity-Interpretability Trade-off

Formally, we define an explanation as a model $g \in G$, where G is a class of potentially *interpretable* models, such as linear models, decision trees, or falling rule lists [27], i.e. a model $g \in G$ can be readily presented to the user with visual or textual artifacts. The domain of g is $\{0,1\}^{d'}$, i.e. g acts over absence/presence of the *interpretable components*. As not every $g \in G$ may be simple enough to be interpretable - thus we let $\Omega(g)$ be a measure of *complexity* (as opposed to *interpretability*) of the explanation $g \in G$. For example, for decision trees $\Omega(g)$ may be the depth of the tree, while for linear models, $\Omega(g)$ may be the number of non-zero weights.

Let the model being explained be denoted $f : \mathbb{R}^d \rightarrow \mathbb{R}$. In classification, $f(x)$ is the probability (or a binary indicator) that x belongs to a certain class[1]. We further use $\pi_x(z)$ as a proximity measure between an instance z to x, so as to define locality around x. Finally, let $\mathcal{L}(f, g, \pi_x)$ be a measure of how unfaithful g is in approximating f in the locality defined by π_x. In order to ensure both **interpretability** and **local fidelity**, we must minimize $\mathcal{L}(f, g, \pi_x)$ while having $\Omega(g)$ be low enough to be interpretable by humans. The explanation produced by **LIME** is obtained by the following:

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \quad \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{1}$$

This formulation can be used with different explanation families G, fidelity functions $\mathcal{L}$, and complexity measures Ω. Here we focus on sparse linear models as explanations, and on performing the search using perturbations.

3.3 Sampling for Local Exploration

We want to minimize the locality-aware loss $\mathcal{L}(f, g, \pi_x)$ without making any assumptions about f, since we want the explainer to be **model-agnostic**. Thus, in order to learn the local behavior of f as the interpretable inputs vary, we approximate $\mathcal{L}(f, g, \pi_x)$ by drawing samples, weighted by π_x. We sample instances around x' by drawing nonzero elements of x' uniformly at random (where the number of such draws is also uniformly sampled). Given a perturbed sample $z' \in \{0,1\}^{d'}$ (which contains a fraction of the nonzero elements of x'), we recover the sample in the original representation $z \in R^d$ and obtain $f(z)$, which is used as a *label* for the explanation model. Given this dataset $\mathcal{Z}$ of perturbed samples with the associated labels, we optimize Eq. (1) to get an explanation $\xi(x)$. The primary intuition behind LIME is presented in Figure 3, where we sample instances both in the vicinity of x (which have a high weight due to π_x) and far away from x (low weight from π_x). Even though the original model may be too complex to explain globally, LIME presents an explanation that is locally faithful (linear in this case), where the locality is captured by π_x. It is worth noting that our method is fairly robust to sampling noise since the samples are weighted by π_x in Eq. (1). We now present a concrete instance of this general framework.

3.4 Sparse Linear Explanations

For the rest of this paper, we let G be the class of linear models, such that $g(z') = w_g \cdot z'$. We use the locally weighted square loss as $\mathcal{L}$, as defined in Eq. (2), where we let $\pi_x(z) = exp(-D(x, z)^2/\sigma^2)$ be an exponential kernel defined on some

[1]For multiple classes, we explain each class separately, thus $f(x)$ is the prediction of the relevant class.

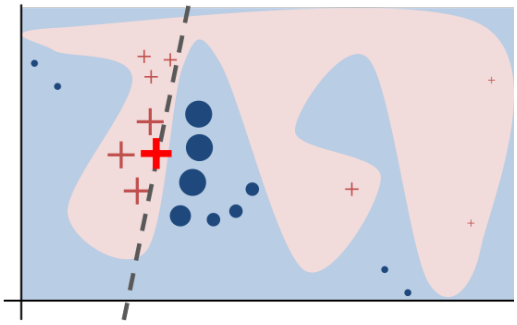


Figure 3: Toy example to present intuition for LIME. The black-box model's complex decision function f (unknown to LIME) is represented by the blue/pink background, which cannot be approximated well by a linear model. The bold red cross is the instance being explained. LIME samples instances, gets predictions using f, and weighs them by the proximity to the instance being explained (represented here by size). The dashed line is the learned explanation that is locally (but not globally) faithful.

distance function D (e.g. cosine distance for text, $L2$ distance for images) with width σ.

$$\mathcal{L}(f, g, \pi_x) = \sum_{z, z' \in \mathcal{Z}} \pi_x(z) \left(f(z) - g(z')\right)^2 \qquad (2)$$

For text classification, we ensure that the explanation is **interpretable** by letting the *interpretable representation* be a bag of words, and by setting a limit K on the number of words, i.e. $\Omega(g) = \infty \mathbb{1}[\|w_g\|_0 > K]$. Potentially, K can be adapted to be as big as the user can handle, or we could have different values of K for different instances. In this paper we use a constant value for K, leaving the exploration of different values to future work. We use the same Ω for image classification, using "super-pixels" (computed using any standard algorithm) instead of words, such that the interpretable representation of an image is a binary vector where 1 indicates the original super-pixel and 0 indicates a grayed out super-pixel. This particular choice of Ω makes directly solving Eq. (1) intractable, but we approximate it by first selecting K features with Lasso (using the regularization path [9]) and then learning the weights via least squares (a procedure we call K-LASSO in Algorithm 1). Since Algorithm 1 produces an explanation for an individual prediction, its complexity does not depend on the size of the dataset, but instead on time to compute $f(x)$ and on the number of samples N. In practice, explaining random forests with 1000 trees using scikit-learn (http://scikit-learn.org) on a laptop with $N = 5000$ takes under 3 seconds without any optimizations such as using gpus or parallelization. Explaining each prediction of the Inception network [25] for image classification takes around 10 minutes.

Any choice of interpretable representations and G will have some inherent drawbacks. First, while the underlying model can be treated as a black-box, certain interpretable representations will not be powerful enough to explain certain behaviors. For example, a model that predicts sepia-toned images to be *retro* cannot be explained by presence of absence of super pixels. Second, our choice of G (sparse linear models) means that if the underlying model is highly non-linear even in the locality of the prediction, there may not be a faithful explanation. However, we can estimate the faithfulness of the explanation on $\mathcal{Z}$, and present this information to the user. This estimate of faithfulness can also be used for selecting an appropriate family of explanations from a set of multiple interpretable model classes, thus adapting to the given dataset and the classifier. We leave such exploration for future work, as linear explanations work quite well for multiple black-box models in our experiments.

Algorithm 1 Sparse Linear Explanations using LIME

Require: Classifier f, Number of samples N
Require: Instance x, and its interpretable version x'
Require: Similarity kernel π_x, Length of explanation K
  $\mathcal{Z} \leftarrow \{\}$
  for $i \in \{1, 2, 3, ..., N\}$ **do**
    $z'_i \leftarrow sample\_around(x')$
    $\mathcal{Z} \leftarrow \mathcal{Z} \cup \langle z'_i, f(z_i), \pi_x(z_i)\rangle$
  end for
  $w \leftarrow$ K-Lasso$(\mathcal{Z}, K)$ ▷ with z'_i as features, $f(z)$ as target
  return w

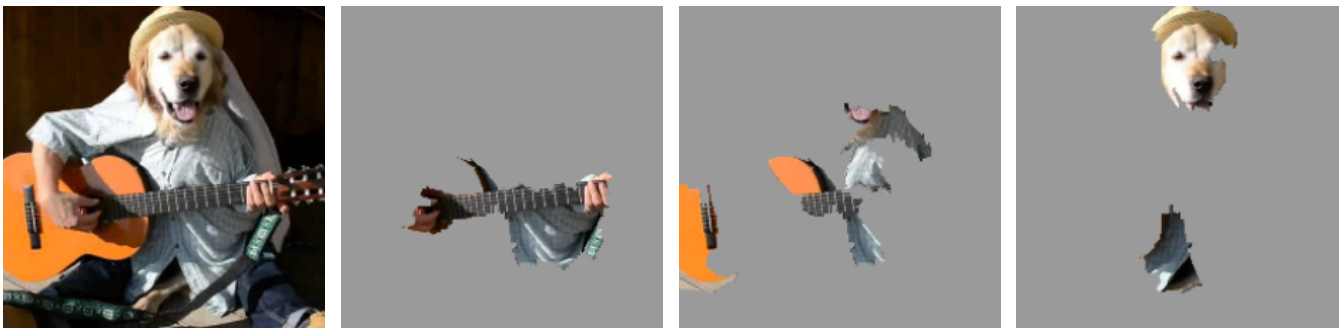
3.5 Example 1: Text classification with SVMs

In Figure 2 (right side), we explain the predictions of a support vector machine with RBF kernel trained on unigrams to differentiate "Christianity" from "Atheism" (on a subset of the 20 newsgroup dataset). Although this classifier achieves 94% held-out accuracy, and one would be tempted to trust it based on this, the explanation for an instance shows that predictions are made for quite arbitrary reasons (words "Posting", "Host", and "Re" have no connection to either Christianity or Atheism). The word "Posting" appears in 22% of examples in the training set, 99% of them in the class "Atheism". Even if headers are removed, proper names of prolific posters in the original newsgroups are selected by the classifier, which would also not generalize.

After getting such insights from explanations, it is clear that this dataset has serious issues (which are not evident just by studying the raw data or predictions), and that this classifier, or held-out evaluation, cannot be trusted. It is also clear what the problems are, and the steps that can be taken to fix these issues and train a more trustworthy classifier.

3.6 Example 2: Deep networks for images

When using sparse linear explanations for image classifiers, one may wish to just highlight the super-pixels with positive weight towards a specific class, as they give intuition as to why the model would think that class may be present. We explain the prediction of Google's pre-trained Inception neural network [25] in this fashion on an arbitrary image (Figure 4a). Figures 4b, 4c, 4d show the superpixels explanations for the top 3 predicted classes (with the rest of the image grayed out), having set $K = 10$. What the neural network picks up on for each of the classes is quite natural to humans - Figure 4b in particular provides insight as to why acoustic guitar was predicted to be electric: due to the fretboard. This kind of explanation enhances trust in the classifier (even if the top predicted class is wrong), as it shows that it is not acting in an unreasonable manner.



(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Figure 4: Explaining an image classification prediction made by Google's Inception neural network. The top 3 classes predicted are "Electric Guitar" ($p = 0.32$), "Acoustic guitar" ($p = 0.24$) and "Labrador" ($p = 0.21$)

4. SUBMODULAR PICK FOR EXPLAINING MODELS

Although an explanation of a single prediction provides some understanding into the reliability of the classifier to the user, it is not sufficient to evaluate and assess trust in the model as a whole. We propose to give a global understanding of the model by explaining a set of individual instances. This approach is still model agnostic, and is complementary to computing summary statistics such as held-out accuracy.

Even though explanations of multiple instances can be insightful, these instances need to be selected judiciously, since users may not have the time to examine a large number of explanations. We represent the time/patience that humans have by a budget B that denotes the number of explanations they are willing to look at in order to understand a model. Given a set of instances X, we define the **pick step** as the task of selecting B instances for the user to inspect.

The pick step is not dependent on the existence of explanations - one of the main purpose of tools like Modeltracker [1] and others [11] is to assist users in selecting instances themselves, and examining the raw data and predictions. However, since looking at raw data is not enough to understand predictions and get insights, the pick step should take into account the explanations that accompany each prediction. Moreover, this method should pick a diverse, representative set of explanations to show the user – i.e. non-redundant explanations that represent how the model behaves globally.

Given the explanations for a set of instances X ($|X| = n$), we construct an $n \times d'$ *explanation matrix* $\mathcal{W}$ that represents the local importance of the interpretable components for each instance. When using linear models as explanations, for an instance x_i and explanation $g_i = \xi(x_i)$, we set $\mathcal{W}_{ij} = |w_{g_{ij}}|$. Further, for each component (column) j in $\mathcal{W}$, we let I_j denote the *global* importance of that component in the explanation space. Intuitively, we want I such that features that explain many different instances have higher importance scores. In Figure 5, we show a toy example $\mathcal{W}$, with $n = d' = 5$, where $\mathcal{W}$ is binary (for simplicity). The importance function I should score feature f2 higher than feature f1, i.e. $I_2 > I_1$, since feature f2 is used to explain more instances. Concretely for the text applications, we set $I_j = \sqrt{\sum_{i=1}^{n} \mathcal{W}_{ij}}$. For images, I must measure something that is comparable across the super-pixels in different images, such as color histograms or other features of super-pixels; we leave further exploration of these ideas for future work.

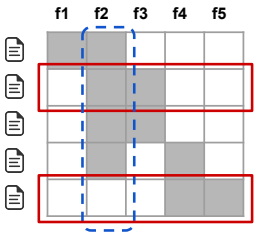


Figure 5: Toy example $\mathcal{W}$. Rows represent instances (documents) and columns represent features (words). Feature f2 (dotted blue) has the highest importance. Rows 2 and 5 (in red) would be selected by the pick procedure, covering all but feature f1.

Algorithm 2 Submodular pick (SP) algorithm

Require: Instances X, Budget B
  for all $x_i \in X$ **do**
    $\mathcal{W}_i \leftarrow$ **explain**(x_i, x_i') ▷ Using Algorithm 1
  end for
  for $j \in \{1 \dots d'\}$ **do**
    $I_j \leftarrow \sqrt{\sum_{i=1}^{n} |\mathcal{W}_{ij}|}$ ▷ Compute feature importances
  end for
  $V \leftarrow \{\}$
  while $|V| < B$ **do** ▷ Greedy optimization of Eq (4)
    $V \leftarrow V \cup \text{argmax}_i\, c(V \cup \{i\}, \mathcal{W}, I)$
  end while
  return V

While we want to pick instances that cover the important components, the set of explanations must not be redundant in the components they show the users, i.e. avoid selecting instances with similar explanations. In Figure 5, after the second row is picked, the third row adds no value, as the user has already seen features f2 and f3 - while the last row exposes the user to completely new features. Selecting the second and last row results in the coverage of almost all the features. We formalize this non-redundant coverage intuition in Eq. (3), where we define coverage as the set function c that, given $\mathcal{W}$ and I, computes the total importance of the features that appear in at least one instance in a set V.

$$c(V, \mathcal{W}, I) = \sum_{j=1}^{d'} \mathbb{1}_{[\exists i \in V : \mathcal{W}_{ij} > 0]} I_j \quad (3)$$

The pick problem, defined in Eq. (4), consists of finding the set $V, |V| \leq B$ that achieves highest coverage.

$$Pick(\mathcal{W}, I) = \underset{V, |V| \leq B}{\operatorname{argmax}} c(V, \mathcal{W}, I) \quad (4)$$

The problem in Eq. (4) is maximizing a weighted coverage function, and is NP-hard [10]. Let $c(V \cup \{i\}, \mathcal{W}, I) - c(V, \mathcal{W}, I)$ be the marginal coverage gain of adding an instance i to a set V. Due to submodularity, a greedy algorithm that iteratively adds the instance with the highest marginal coverage gain to the solution offers a constant-factor approximation guarantee of $1-1/e$ to the optimum [15]. We outline this approximation in Algorithm 2, and call it **submodular pick**.

5. SIMULATED USER EXPERIMENTS

In this section, we present simulated user experiments to evaluate the utility of explanations in trust-related tasks. In particular, we address the following questions: (1) Are the explanations faithful to the model, (2) Can the explanations aid users in ascertaining trust in predictions, and (3) Are the explanations useful for evaluating the model as a whole. Code and data for replicating our experiments are available at https://github.com/marcotcr/lime-experiments.

5.1 Experiment Setup

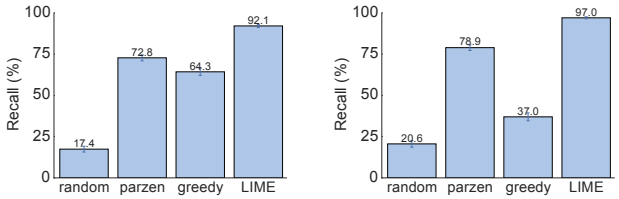
We use two sentiment analysis datasets (*books* and *DVDs*, 2000 instances each) where the task is to classify product reviews as positive or negative [4]. We train decision trees (**DT**), logistic regression with L2 regularization (**LR**), nearest neighbors (**NN**), and support vector machines with RBF kernel (**SVM**), all using bag of words as features. We also include random forests (with 1000 trees) trained with the average word2vec embedding [19] (**RF**), a model that is impossible to interpret without a technique like LIME. We use the implementations and default parameters of scikit-learn, unless noted otherwise. We divide each dataset into train (1600 instances) and test (400 instances).

To explain individual predictions, we compare our proposed approach (**LIME**), with **parzen** [2], a method that approximates the black box classifier globally with Parzen windows, and explains individual predictions by taking the gradient of the prediction probability function. For parzen, we take the K features with the highest absolute gradients as explanations. We set the hyper-parameters for parzen and LIME using cross validation, and set $N = 15,000$. We also compare against a **greedy** procedure (similar to Martens and Provost [18]) in which we greedily remove features that contribute the most to the predicted class until the prediction changes (or we reach the maximum of K features), and a **random** procedure that randomly picks K features as an explanation. We set K to 10 for our experiments.

For experiments where the pick procedure applies, we either do random selection (random pick, **RP**) or the procedure described in §4 (submodular pick, **SP**). We refer to pick-explainer combinations by adding RP or SP as a prefix.

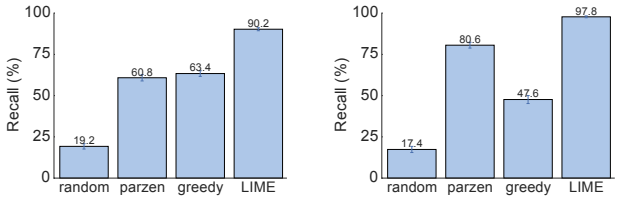
5.2 Are explanations faithful to the model?

We measure faithfulness of explanations on classifiers that are by themselves interpretable (sparse logistic regression and decision trees). In particular, we train both classifiers such that the maximum number of features they use for any instance is 10, and thus we know the *gold* set of features that the are considered important by these models. For each prediction on the test set, we generate explanations and compute the fraction of these *gold* features that are recovered by the explanations. We report this recall averaged over all the test instances in Figures 6 and 7. We observe that the greedy approach is comparable to parzen on logistic regression, but is substantially worse on decision trees since changing a single feature at a time often does not have an effect on the prediction. The overall recall by parzen is low, likely due to the difficulty in approximating the original high-dimensional classifier. LIME consistently provides $> 90\%$ recall for both classifiers on both datasets, demonstrating that LIME explanations are faithful to the models.



(a) Sparse LR (b) Decision Tree

Figure 6: Recall on truly important features for two interpretable classifiers on the books dataset.



(a) Sparse LR (b) Decision Tree

Figure 7: Recall on truly important features for two interpretable classifiers on the DVDs dataset.

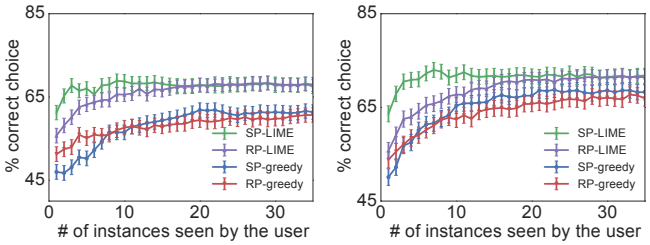
5.3 Should I trust this prediction?

In order to simulate trust in individual predictions, we first randomly select 25% of the features to be "untrustworthy", and assume that the users can identify and would not want to trust these features (such as the headers in 20 newsgroups, leaked data, etc). We thus develop *oracle* "trustworthiness" by labeling test set predictions from a black box classifier as "untrustworthy" if the prediction changes when untrustworthy features are removed from the instance, and "trustworthy" otherwise. In order to simulate users, we assume that users deem predictions untrustworthy from LIME and parzen explanations if the prediction from the linear approximation changes when all untrustworthy features that appear in the explanations are removed (the simulated human "discounts" the effect of untrustworthy features). For greedy and random, the prediction is mistrusted if any untrustworthy features are present in the explanation, since these methods do not provide a notion of the contribution of each feature to the prediction. Thus for each test set prediction, we can evaluate whether the simulated user trusts it using each explanation method, and compare it to the trustworthiness oracle.

Using this setup, we report the F1 on the trustworthy

Table 1: Average F1 of *trustworthiness* for different explainers on a collection of classifiers and datasets.

	Books				DVDs			
	LR	NN	RF	SVM	LR	NN	RF	SVM
Random	14.6	14.8	14.7	14.7	14.2	14.3	14.5	14.4
Parzen	84.0	87.6	94.3	92.3	87.0	81.7	94.2	87.3
Greedy	53.7	47.4	45.0	53.3	52.4	58.1	46.6	55.1
LIME	**96.6**	**94.5**	**96.2**	**96.7**	**96.6**	**91.8**	**96.1**	**95.6**



(a) Books dataset (b) DVDs dataset

Figure 8: Choosing between two classifiers, as the number of instances shown to a simulated user is varied. Averages and standard errors from 800 runs.

predictions for each explanation method, averaged over 100 runs, in Table 1. The results indicate that LIME dominates others (all results are significant at $p = 0.01$) on both datasets, and for all of the black box models. The other methods either achieve a lower recall (i.e. they mistrust predictions more than they should) or lower precision (i.e. they trust too many predictions), while LIME maintains both high precision and high recall. Even though we artificially select which features are untrustworthy, these results indicate that LIME is helpful in assessing trust in individual predictions.

5.4 Can I trust this model?

In the final simulated user experiment, we evaluate whether the explanations can be used for model selection, simulating the case where a human has to decide between two competing models with similar accuracy on validation data. For this purpose, we add 10 artificially "noisy" features. Specifically, on training and validation sets (80/20 split of the original training data), each artificial feature appears in 10% of the examples in one class, and 20% of the other, while on the test instances, each artificial feature appears in 10% of the examples in each class. This recreates the situation where the models use not only features that are informative in the real world, but also ones that introduce spurious correlations. We create pairs of competing classifiers by repeatedly training pairs of random forests with 30 trees until their validation accuracy is within 0.1% of each other, but their test accuracy differs by at least 5%. Thus, it is not possible to identify the *better* classifier (the one with higher test accuracy) from the accuracy on the validation data.

The goal of this experiment is to evaluate whether a user can identify the better classifier based on the explanations of B instances from the validation set. The simulated human marks the set of artificial features that appear in the B explanations as untrustworthy, following which we evaluate how many total predictions in the validation set should be trusted (as in the previous section, treating only marked features as untrustworthy). Then, we select the classifier with fewer untrustworthy predictions, and compare this choice to the classifier with higher held-out test set accuracy.

We present the accuracy of picking the correct classifier as B varies, averaged over 800 runs, in Figure 8. We omit SP-parzen and RP-parzen from the figure since they did not produce useful explanations, performing only slightly better than random. LIME is consistently better than greedy, irrespective of the pick method. Further, combining submodular pick with LIME outperforms all other methods, in particular it is much better than RP-LIME when only a few examples are shown to the users. These results demonstrate that the trust assessments provided by SP-selected LIME explanations are good indicators of generalization, which we validate with human experiments in the next section.

6. EVALUATION WITH HUMAN SUBJECTS

In this section, we recreate three scenarios in machine learning that require trust and understanding of predictions and models. In particular, we evaluate LIME and SP-LIME in the following settings: (1) Can users choose which of two classifiers generalizes better (§ 6.2), (2) based on the explanations, can users perform feature engineering to improve the model (§ 6.3), and (3) are users able to identify and describe classifier irregularities by looking at explanations (§ 6.4).

6.1 Experiment setup

For experiments in §6.2 and §6.3, we use the "Christianity" and "Atheism" documents from the 20 newsgroups dataset mentioned beforehand. This dataset is problematic since it contains features that do not generalize (e.g. very informative header information and author names), and thus validation accuracy considerably overestimates real-world performance.

In order to estimate the real world performance, we create a new *religion dataset* for evaluation. We download Atheism and Christianity websites from the DMOZ directory and human curated lists, yielding 819 webpages in each class. High accuracy on this dataset by a classifier trained on 20 newsgroups indicates that the classifier is generalizing using semantic content, instead of placing importance on the data specific issues outlined above. Unless noted otherwise, we use SVM with RBF kernel, trained on the 20 newsgroups data with hyper-parameters tuned via the cross-validation.

6.2 Can users select the best classifier?

In this section, we want to evaluate whether explanations can help users decide which classifier generalizes better, i.e., which classifier would the user deploy "in the wild". Specifically, users have to decide between two classifiers: SVM trained on the original 20 newsgroups dataset, and a version of the same classifier trained on a "cleaned" dataset where many of the features that do not generalize have been manually removed. The original classifier achieves an accuracy score of 57.3% on the *religion dataset*, while the "cleaned" classifier achieves a score of 69.0%. In contrast, the test accuracy on the original 20 newsgroups split is 94.0% and 88.6%, respectively – suggesting that the worse classifier would be selected if accuracy alone is used as a measure of trust.

We recruit human subjects on Amazon Mechanical Turk – by no means machine learning experts, but instead people with basic knowledge about religion. We measure their ability to choose the better algorithm by seeing side-by-side explanations with the associated raw data (as shown in Figure 2). We restrict both the number of words in each explanation (K) and the number of documents that each

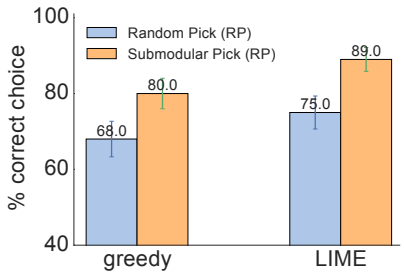


Figure 9: Average accuracy of human subject (with standard errors) in choosing between two classifiers.

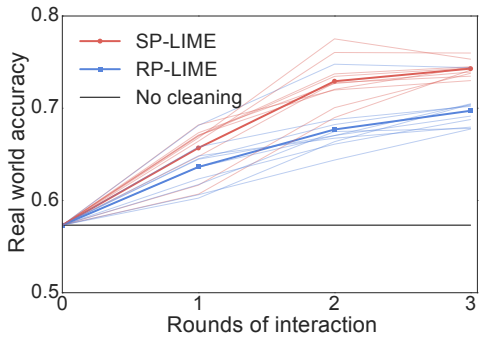


Figure 10: Feature engineering experiment. Each shaded line represents the average accuracy of subjects in a path starting from one of the initial 10 subjects. Each solid line represents the average across all paths per round of interaction.

person inspects (B) to 6. The position of each algorithm and the order of the instances seen are randomized between subjects. After examining the explanations, users are asked to select which algorithm will perform best in the real world. The explanations are produced by either greedy (chosen as a baseline due to its performance in the simulated user experiment) or LIME, and the instances are selected either by random (RP) or submodular pick (SP). We modify the greedy step in Algorithm 2 slightly so it alternates between explanations of the two classifiers. For each setting, we repeat the experiment with 100 users.

The results are presented in Figure 9. Note that all of the methods are good at identifying the better classifier, demonstrating that the explanations are useful in determining which classifier to trust, while using test set accuracy would result in the selection of the wrong classifier. Further, we see that the submodular pick (SP) greatly improves the user's ability to select the best classifier when compared to random pick (RP), with LIME outperforming greedy in both cases.

6.3 Can non-experts improve a classifier?

If one notes that a classifier is untrustworthy, a common task in machine learning is feature engineering, i.e. modifying the set of features and retraining in order to improve generalization. Explanations can aid in this process by presenting the important features, particularly for removing features that the users feel do not generalize.

We use the 20 newsgroups data here as well, and ask Amazon Mechanical Turk users to identify which words from the explanations should be removed from subsequent training, for the worse classifier from the previous section (§6.2). In each round, the subject marks words for deletion after observing $B = 10$ instances with $K = 10$ words in each explanation (an interface similar to Figure 2, but with a single algorithm). As a reminder, the users here are not experts in machine learning and are unfamiliar with feature engineering, thus are only identifying words based on their semantic content. Further, users do not have any access to the *religion* dataset – they do not even know of its existence. We start the experiment with 10 subjects. After they mark words for deletion, we train 10 different classifiers, one for each subject (with the corresponding words removed). The explanations for each classifier are then presented to a set of 5 users in a new round of interaction, which results in 50 new classifiers. We do a final round, after which we have 250 classifiers, each with a path of interaction tracing back to the first 10 subjects.

The explanations and instances shown to each user are produced by **SP-LIME** or **RP-LIME**. We show the average accuracy on the *religion* dataset at each interaction round for the paths originating from each of the original 10 subjects (shaded lines), and the average across all paths (solid lines) in Figure 10. It is clear from the figure that the crowd workers are able to improve the model by removing features they deem unimportant for the task. Further, **SP-LIME** outperforms **RP-LIME**, indicating selection of the instances to show the users is crucial for efficient feature engineering.

Each subject took an average of 3.6 minutes per round of cleaning, resulting in just under 11 minutes to produce a classifier that generalizes much better to real world data. Each path had on average 200 words removed with **SP**, and 157 with **RP**, indicating that incorporating coverage of important features is useful for feature engineering. Further, out of an average of 200 words selected with SP, 174 were selected by at least half of the users, while 68 by *all* the users. Along with the fact that the variance in the accuracy decreases across rounds, this high agreement demonstrates that the users are converging to similar *correct* models. This evaluation is an example of how explanations make it easy to improve an untrustworthy classifier – in this case easy enough that machine learning knowledge is not required.

6.4 Do explanations lead to insights?

Often artifacts of data collection can induce undesirable correlations that the classifiers pick up during training. These issues can be very difficult to identify just by looking at the raw data and predictions. In an effort to reproduce such a setting, we take the task of distinguishing between photos of Wolves and Eskimo Dogs (huskies). We train a logistic regression classifier on a training set of 20 images, hand selected such that all pictures of wolves had snow in the background, while pictures of huskies did not. As the features for the images, we use the first max-pooling layer of Google's pre-trained Inception neural network [25]. On a collection of additional 60 images, the classifier predicts "Wolf" if there is snow (or light background at the bottom), and "Husky" otherwise, regardless of animal color, position, pose, etc. We trained this *bad* classifier intentionally, to evaluate whether subjects are able to detect it.

The experiment proceeds as follows: we first present a balanced set of 10 test predictions (without explanations), where one wolf is not in a snowy background (and thus the prediction is "Husky") and one husky is (and is thus predicted as "Wolf"). We show the "Husky" mistake in Figure 11a. The other 8 examples are classified correctly. We then ask the subject three questions: (1) Do they trust this algorithm

(a) Husky classified as wolf (b) Explanation

Figure 11: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task.

	Before	After
Trusted the bad model	10 out of 27	3 out of 27
Snow as a potential feature	12 out of 27	25 out of 27

Table 2: "Husky vs Wolf" experiment results.

to work well in the real world, (2) why, and (3) how do they think the algorithm is able to distinguish between these photos of wolves and huskies. After getting these responses, we show the same images with the associated explanations, such as in Figure 11b, and ask the same questions.

Since this task requires some familiarity with the notion of spurious correlations and generalization, the set of subjects for this experiment were graduate students who have taken at least one graduate machine learning course. After gathering the responses, we had 3 independent evaluators read their reasoning and determine if each subject mentioned snow, background, or equivalent as a feature the model may be using. We pick the majority to decide whether the subject was correct about the insight, and report these numbers before and after showing the explanations in Table 2.

Before observing the explanations, more than a third trusted the classifier, and a little less than half mentioned the snow pattern as something the neural network was using – although all speculated on other patterns. After examining the explanations, however, almost all of the subjects identified the correct insight, with much more certainty that it was a determining factor. Further, the trust in the classifier also dropped substantially. Although our sample size is small, this experiment demonstrates the utility of explaining individual predictions for getting insights into classifiers knowing when not to trust them and why.

7. RELATED WORK

The problems with relying on validation set accuracy as the primary measure of trust have been well studied. Practitioners consistently overestimate their model's accuracy [20], propagate feedback loops [23], or fail to notice data leaks [14]. In order to address these issues, researchers have proposed tools like Gestalt [21] and Modeltracker [1], which help users navigate individual instances. These tools are complementary to LIME in terms of explaining models, since they do not address the problem of explaining individual predictions. Further, our submodular pick procedure can be incorporated in such tools to aid users in navigating larger datasets.

Some recent work aims to anticipate failures in machine learning, specifically for vision tasks [3, 29]. Letting users know when the systems are likely to fail can lead to an increase in trust, by avoiding "silly mistakes" [8]. These solutions either require additional annotations and feature engineering that is specific to vision tasks or do not provide insight into why a decision should not be trusted. Furthermore, they assume that the current evaluation metrics are reliable, which may not be the case if problems such as data leakage are present. Other recent work [11] focuses on exposing users to different kinds of mistakes (our pick step). Interestingly, the subjects in their study did not notice the serious problems in the 20 newsgroups data even after looking at many mistakes, suggesting that examining raw data is not sufficient. Note that Groce et al. [11] are not alone in this regard, many researchers in the field have unwittingly published classifiers that would not generalize for this task. Using LIME, we show that even non-experts are able to identify these irregularities when explanations are present. Further, LIME can complement these existing systems, and allow users to assess trust even when a prediction seems "correct" but is made for the wrong reasons.

Recognizing the utility of explanations in assessing trust, many have proposed using interpretable models [27], especially for the medical domain [6, 17, 26]. While such models may be appropriate for some domains, they may not apply equally well to others (e.g. a supersparse linear model [26] with $5 - 10$ features is unsuitable for text applications). Interpretability, in these cases, comes at the cost of flexibility, accuracy, or efficiency. For text, EluciDebug [16] is a full human-in-the-loop system that shares many of our goals (interpretability, faithfulness, etc). However, they focus on an already interpretable model (Naive Bayes). In computer vision, systems that rely on object detection to produce candidate alignments [13] or attention [28] are able to produce explanations for their predictions. These are, however, constrained to specific neural network architectures or incapable of detecting "non object" parts of the images. Here we focus on general, model-agnostic explanations that can be applied to any classifier or regressor that is appropriate for the domain - even ones that are yet to be proposed.

A common approach to model-agnostic explanation is learning a potentially interpretable model on the predictions of the original model [2, 7, 22]. Having the explanation be a gradient vector [2] captures a similar locality intuition to that of LIME. However, interpreting the coefficients on the gradient is difficult, particularly for confident predictions (where gradient is near zero). Further, these explanations approximate the original model *globally*, thus maintaining local fidelity becomes a significant challenge, as our experiments demonstrate. In contrast, LIME solves the much more feasible task of finding a model that approximates the original model *locally*. The idea of perturbing inputs for explanations has been explored before [24], where the authors focus on learning a specific *contribution* model, as opposed to our general framework. None of these approaches explicitly take cognitive limitations into account, and thus may produce non-interpretable explanations, such as a gradients or linear models with thousands of non-zero weights. The problem becomes worse if the original features are nonsensical to humans (e.g. word embeddings). In contrast, LIME incorporates interpretability both in the optimization and in our notion of *interpretable representation*, such that domain and task specific interpretability criteria can be accommodated.

8. CONCLUSION AND FUTURE WORK

In this paper, we argued that trust is crucial for effective human interaction with machine learning systems, and that explaining individual predictions is important in assessing trust. We proposed LIME, a modular and extensible approach to faithfully explain the predictions of *any* model in an interpretable manner. We also introduced SP-LIME, a method to select representative and non-redundant predictions, providing a global view of the model to users. Our experiments demonstrated that explanations are useful for a variety of models in trust-related tasks in the text and image domains, with both expert and non-expert users: deciding between models, assessing trust, improving untrustworthy models, and getting insights into predictions.

There are a number of avenues of future work that we would like to explore. Although we describe only sparse linear models as explanations, our framework supports the exploration of a variety of explanation families, such as decision trees; it would be interesting to see a comparative study on these with real users. One issue that we do not mention in this work was how to perform the pick step for images, and we would like to address this limitation in the future. The domain and model agnosticism enables us to explore a variety of applications, and we would like to investigate potential uses in speech, video, and medical domains, as well as recommendation systems. Finally, we would like to explore theoretical properties (such as the appropriate number of samples) and computational optimizations (such as using parallelization and GPU processing), in order to provide the accurate, real-time explanations that are critical for any human-in-the-loop machine learning system.

Acknowledgements

We would like to thank Scott Lundberg, Tianqi Chen, and Tyler Johnson for helpful discussions and feedback. This work was supported in part by ONR awards #W911NF-13-1-0246 and #N00014-13-1-0023, and in part by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

9. REFERENCES

[1] S. Amershi, M. Chickering, S. M. Drucker, B. Lee, P. Simard, and J. Suh. Modeltracker: Redesigning performance analysis tools for machine learning. In *Human Factors in Computing Systems (CHI)*, 2015.

[2] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. Müller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11, 2010.

[3] A. Bansal, A. Farhadi, and D. Parikh. Towards transparent systems: Semantic characterization of failure modes. In *European Conference on Computer Vision (ECCV)*, 2014.

[4] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Association for Computational Linguistics (ACL)*, 2007.

[5] J. Q. Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset Shift in Machine Learning*. MIT, 2009.

[6] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Knowledge Discovery and Data Mining (KDD)*, 2015.

[7] M. W. Craven and J. W. Shavlik. Extracting tree-structured representations of trained networks. *Neural information processing systems (NIPS)*, pages 24–30, 1996.

[8] M. T. Dzindolet, S. A. Peterson, R. A. Pomranky, L. G. Pierce, and H. P. Beck. The role of trust in automation reliance. *Int. J. Hum.-Comput. Stud.*, 58(6), 2003.

[9] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[10] U. Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4), July 1998.

[11] A. Groce, T. Kulesza, C. Zhang, S. Shamasunder, M. Burnett, W.-K. Wong, S. Stumpf, S. Das, A. Shinsel, F. Bice, and K. McIntosh. You are the only possible oracle: Effective test selection for end users of interactive machine learning systems. *IEEE Trans. Softw. Eng.*, 40(3), 2014.

[12] J. L. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Conference on Computer Supported Cooperative Work (CSCW)*, 2000.

[13] A. Karpathy and F. Li. Deep visual-semantic alignments for generating image descriptions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[14] S. Kaufman, S. Rosset, and C. Perlich. Leakage in data mining: Formulation, detection, and avoidance. In *Knowledge Discovery and Data Mining (KDD)*, 2011.

[15] A. Krause and D. Golovin. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, February 2014.

[16] T. Kulesza, M. Burnett, W.-K. Wong, and S. Stumpf. Principles of explanatory debugging to personalize interactive machine learning. In *Intelligent User Interfaces (IUI)*, 2015.

[17] B. Letham, C. Rudin, T. H. McCormick, and D. Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *Annals of Applied Statistics*, 2015.

[18] D. Martens and F. Provost. Explaining data-driven document classifications. *MIS Q.*, 38(1), 2014.

[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Neural Information Processing Systems (NIPS)*. 2013.

[20] K. Patel, J. Fogarty, J. A. Landay, and B. Harrison. Investigating statistical machine learning as a tool for software development. In *Human Factors in Computing Systems (CHI)*, 2008.

[21] K. Patel, N. Bancroft, S. M. Drucker, J. Fogarty, A. J. Ko, and J. Landay. Gestalt: Integrated support for implementation and analysis in machine learning. In *User Interface Software and Technology (UIST)*, 2010.

[22] I. Sanchez, T. Rocktaschel, S. Riedel, and S. Singh. Towards extracting faithful and descriptive representations of latent variable models. In *AAAI Spring Syposium on Knowledge Representation and Reasoning (KRR): Integrating Symbolic and Neural Approaches*, 2015.

[23] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, and J.-F. Crespo. Hidden technical debt in machine learning systems. In *Neural Information Processing Systems (NIPS)*. 2015.

[24] E. Strumbelj and I. Kononenko. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11, 2010.

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[26] B. Ustun and C. Rudin. Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 2015.

[27] F. Wang and C. Rudin. Falling rule lists. In *Artificial Intelligence and Statistics (AISTATS)*, 2015.

[28] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning (ICML)*, 2015.

[29] P. Zhang, J. Wang, A. Farhadi, M. Hebert, and D. Parikh. Predicting failures of vision systems. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.

A Unified Approach to Interpreting Model Predictions

Scott M. Lundberg
Paul G. Allen School of Computer Science
University of Washington
Seattle, WA 98105
slund1@cs.washington.edu

Su-In Lee
Paul G. Allen School of Computer Science
Department of Genome Sciences
University of Washington
Seattle, WA 98105
suinlee@cs.washington.edu

Abstract

Understanding why a model makes a certain prediction can be as crucial as the prediction's accuracy in many applications. However, the highest accuracy for large modern datasets is often achieved by complex models that even experts struggle to interpret, such as ensemble or deep learning models, creating a tension between *accuracy* and *interpretability*. In response, various methods have recently been proposed to help users interpret the predictions of complex models, but it is often unclear how these methods are related and when one method is preferable over another. To address this problem, we present a unified framework for interpreting predictions, SHAP (SHapley Additive exPlanations). SHAP assigns each feature an importance value for a particular prediction. Its novel components include: (1) the identification of a new class of additive feature importance measures, and (2) theoretical results showing there is a unique solution in this class with a set of desirable properties. The new class unifies six existing methods, notable because several recent methods in the class lack the proposed desirable properties. Based on insights from this unification, we present new methods that show improved computational performance and/or better consistency with human intuition than previous approaches.

1 Introduction

The ability to correctly interpret a prediction model's output is extremely important. It engenders appropriate user trust, provides insight into how a model may be improved, and supports understanding of the process being modeled. In some applications, simple models (e.g., linear models) are often preferred for their ease of interpretation, even if they may be less accurate than complex ones. However, the growing availability of big data has increased the benefits of using complex models, so bringing to the forefront the trade-off between accuracy and interpretability of a model's output. A wide variety of different methods have been recently proposed to address this issue [5, 8, 9, 3, 4, 1]. But an understanding of how these methods relate and when one method is preferable to another is still lacking.

Here, we present a novel unified approach to interpreting model predictions.[1] Our approach leads to three potentially surprising results that bring clarity to the growing space of methods:

1. We introduce the perspective of viewing *any* explanation of a model's prediction as a model itself, which we term the *explanation model*. This lets us define the class of *additive feature attribution methods* (Section 2), which unifies six current methods.

[1] https://github.com/slundberg/shap

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

2. We then show that game theory results guaranteeing a unique solution apply to the *entire class* of additive feature attribution methods (Section 3) and propose *SHAP values* as a unified measure of feature importance that various methods approximate (Section 4).
3. We propose new SHAP value estimation methods and demonstrate that they are better aligned with human intuition as measured by user studies and more effectually discriminate among model output classes than several existing methods (Section 5).

2 Additive Feature Attribution Methods

The best explanation of a simple model is the model itself; it perfectly represents itself and is easy to understand. For complex models, such as ensemble methods or deep networks, we cannot use the original model as its own best explanation because it is not easy to understand. Instead, we must use a simpler *explanation model*, which we define as any interpretable approximation of the original model. We show below that six current explanation methods from the literature all use the same explanation model. This previously unappreciated unity has interesting implications, which we describe in later sections.

Let f be the original prediction model to be explained and g the explanation model. Here, we focus on *local methods* designed to explain a prediction $f(x)$ based on a single input x, as proposed in LIME [5]. Explanation models often use *simplified inputs* x' that map to the original inputs through a mapping function $x = h_x(x')$. Local methods try to ensure $g(z') \approx f(h_x(z'))$ whenever $z' \approx x'$. (Note that $h_x(x') = x$ even though x' may contain less information than x because h_x is specific to the current input x.)

Definition 1 Additive feature attribution methods *have an explanation model that is a linear function of binary variables:*

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i, \tag{1}$$

where $z' \in \{0,1\}^M$*,* M *is the number of simplified input features, and* $\phi_i \in \mathbb{R}$*.*

Methods with explanation models matching Definition 1 attribute an effect ϕ_i to each feature, and summing the effects of all feature attributions approximates the output $f(x)$ of the original model. Many current methods match Definition 1, several of which are discussed below.

2.1 LIME

The *LIME* method interprets individual model predictions based on locally approximating the model around a given prediction [5]. The local linear explanation model that LIME uses adheres to Equation 1 exactly and is thus an additive feature attribution method. LIME refers to simplified inputs x' as "interpretable inputs," and the mapping $x = h_x(x')$ converts a binary vector of interpretable inputs into the original input space. Different types of h_x mappings are used for different input spaces. For bag of words text features, h_x converts a vector of 1's or 0's (present or not) into the original word count if the simplified input is one, or zero if the simplified input is zero. For images, h_x treats the image as a set of super pixels; it then maps 1 to leaving the super pixel as its original value and 0 to replacing the super pixel with an average of neighboring pixels (this is meant to represent being missing).

To find ϕ, LIME minimizes the following objective function:

$$\xi = \underset{g \in \mathcal{G}}{\arg\min}\ L(f, g, \pi_{x'}) + \Omega(g). \tag{2}$$

Faithfulness of the explanation model $g(z')$ to the original model $f(h_x(z'))$ is enforced through the loss L over a set of samples in the simplified input space weighted by the local kernel $\pi_{x'}$. Ω penalizes the complexity of g. Since in LIME g follows Equation 1 and L is a squared loss, Equation 2 can be solved using penalized linear regression.

2.2 DeepLIFT

DeepLIFT was recently proposed as a recursive prediction explanation method for deep learning [8, 7]. It attributes to each input x_i a value $C_{\Delta x_i \Delta y}$ that represents the effect of that input being set to a reference value as opposed to its original value. This means that for DeepLIFT, the mapping $x = h_x(x')$ converts binary values into the original inputs, where 1 indicates that an input takes its original value, and 0 indicates that it takes the reference value. The reference value, though chosen by the user, represents a typical uninformative background value for the feature.

DeepLIFT uses a "summation-to-delta" property that states:

$$\sum_{i=1}^{n} C_{\Delta x_i \Delta o} = \Delta o, \tag{3}$$

where $o = f(x)$ is the model output, $\Delta o = f(x) - f(r)$, $\Delta x_i = x_i - r_i$, and r is the reference input. If we let $\phi_i = C_{\Delta x_i \Delta o}$ and $\phi_0 = f(r)$, then DeepLIFT's explanation model matches Equation 1 and is thus another additive feature attribution method.

2.3 Layer-Wise Relevance Propagation

The *layer-wise relevance propagation* method interprets the predictions of deep networks [1]. As noted by Shrikumar et al., this menthod is equivalent to DeepLIFT with the reference activations of all neurons fixed to zero. Thus, $x = h_x(x')$ converts binary values into the original input space, where 1 means that an input takes its original value, and 0 means an input takes the 0 value. Layer-wise relevance propagation's explanation model, like DeepLIFT's, matches Equation 1.

2.4 Classic Shapley Value Estimation

Three previous methods use classic equations from cooperative game theory to compute explanations of model predictions: Shapley regression values [4], Shapley sampling values [9], and Quantitative Input Influence [3].

Shapley regression values are feature importances for linear models in the presence of multicollinearity. This method requires retraining the model on all feature subsets $S \subseteq F$, where F is the set of all features. It assigns an importance value to each feature that represents the effect on the model prediction of including that feature. To compute this effect, a model $f_{S\cup\{i\}}$ is trained with that feature present, and another model f_S is trained with the feature withheld. Then, predictions from the two models are compared on the current input $f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S)$, where x_S represents the values of the input features in the set S. Since the effect of withholding a feature depends on other features in the model, the preceding differences are computed for all possible subsets $S \subseteq F \setminus \{i\}$. The Shapley values are then computed and used as feature attributions. They are a weighted average of all possible differences:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S) \right]. \tag{4}$$

For Shapley regression values, h_x maps 1 or 0 to the original input space, where 1 indicates the input is included in the model, and 0 indicates exclusion from the model. If we let $\phi_0 = f_\varnothing(\varnothing)$, then the Shapley regression values match Equation 1 and are hence an additive feature attribution method.

Shapley sampling values are meant to explain any model by: (1) applying sampling approximations to Equation 4, and (2) approximating the effect of removing a variable from the model by integrating over samples from the training dataset. This eliminates the need to retrain the model and allows fewer than $2^{|F|}$ differences to be computed. Since the explanation model form of Shapley sampling values is the same as that for Shapley regression values, it is also an additive feature attribution method.

Quantitative input influence is a broader framework that addresses more than feature attributions. However, as part of its method it independently proposes a sampling approximation to Shapley values that is nearly identical to Shapley sampling values. It is thus another additive feature attribution method.

3 Simple Properties Uniquely Determine Additive Feature Attributions

A surprising attribute of the class of additive feature attribution methods is the presence of a single unique solution in this class with three desirable properties (described below). While these properties are familiar to the classical Shapley value estimation methods, they were previously unknown for other additive feature attribution methods.

The first desirable property is *local accuracy*. When approximating the original model f for a specific input x, local accuracy requires the explanation model to at least match the output of f for the simplified input x' (which corresponds to the original input x).

Property 1 (Local accuracy)

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i \tag{5}$$

The explanation model $g(x')$ matches the original model $f(x)$ when $x = h_x(x')$.

The second property is *missingness*. If the simplified inputs represent feature presence, then missingness requires features missing in the original input to have no impact. All of the methods described in Section 2 obey the missingness property.

Property 2 (Missingness)

$$x'_i = 0 \implies \phi_i = 0 \tag{6}$$

Missingness constrains features where $x'_i = 0$ to have no attributed impact.

The third property is *consistency*. Consistency states that if a model changes so that some simplified input's contribution increases or stays the same regardless of the other inputs, that input's attribution should not decrease.

Property 3 (Consistency) *Let $f_x(z') = f(h_x(z'))$ and $z' \setminus i$ denote setting $z'_i = 0$. For any two models f and f', if*

$$f'_x(z') - f'_x(z' \setminus i) \geq f_x(z') - f_x(z' \setminus i) \tag{7}$$

for all inputs $z' \in \{0,1\}^M$, then $\phi_i(f', x) \geq \phi_i(f, x)$.

Theorem 1 *Only one possible explanation model g follows Definition 1 and satisfies Properties 1, 2, and 3:*

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!(M - |z'| - 1)!}{M!} \left[f_x(z') - f_x(z' \setminus i) \right] \tag{8}$$

where $|z'|$ is the number of non-zero entries in z', and $z' \subseteq x'$ represents all z' vectors where the non-zero entries are a subset of the non-zero entries in x'.

Theorem 1 follows from combined cooperative game theory results, where the values ϕ_i are known as Shapley values [6]. Young (1985) demonstrated that Shapley values are the only set of values that satisfy three axioms similar to Property 1, Property 3, and a final property that we show to be redundant in this setting (see Supplementary Material). Property 2 is required to adapt the Shapley proofs to the class of additive feature attribution methods.

Under Properties 1-3, for a given simplified input mapping h_x, Theorem 1 shows that there is only one possible additive feature attribution method. This result implies that methods not based on Shapley values violate local accuracy and/or consistency (methods in Section 2 already respect missingness). The following section proposes a unified approach that improves previous methods, preventing them from unintentionally violating Properties 1 and 3.

4 SHAP (SHapley Additive exPlanation) Values

We propose SHAP values as a unified measure of feature importance. These are the Shapley values of a conditional expectation function of the original model; thus, they are the solution to Equation

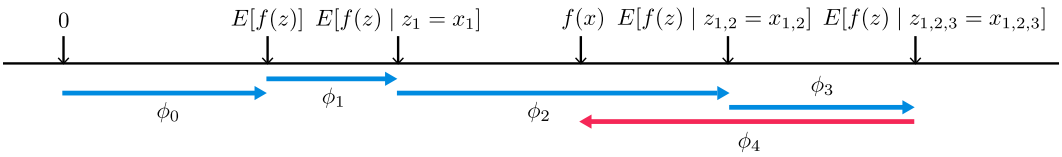


Figure 1: SHAP (SHapley Additive exPlanation) values attribute to each feature the change in the expected model prediction when conditioning on that feature. They explain how to get from the base value $E[f(z)]$ that would be predicted if we did not know any features to the current output $f(x)$. This diagram shows a single ordering. When the model is non-linear or the input features are not independent, however, the order in which features are added to the expectation matters, and the SHAP values arise from averaging the ϕ_i values across all possible orderings.

8, where $f_x(z') = f(h_x(z')) = E[f(z) \mid z_S]$, and S is the set of non-zero indexes in z' (Figure 1). Based on Sections 2 and 3, SHAP values provide the unique additive feature importance measure that adheres to Properties 1-3 and uses conditional expectations to define simplified inputs. Implicit in this definition of SHAP values is a simplified input mapping, $h_x(z') = z_S$, where z_S has missing values for features not in the set S. Since most models cannot handle arbitrary patterns of missing input values, we approximate $f(z_S)$ with $E[f(z) \mid z_S]$. This definition of SHAP values is designed to closely align with the Shapley regression, Shapley sampling, and quantitative input influence feature attributions, while also allowing for connections with LIME, DeepLIFT, and layer-wise relevance propagation.

The exact computation of SHAP values is challenging. However, by combining insights from current additive feature attribution methods, we can approximate them. We describe two model-agnostic approximation methods, one that is already known (Shapley sampling values) and another that is novel (Kernel SHAP). We also describe four model-type-specific approximation methods, two of which are novel (Max SHAP, Deep SHAP). When using these methods, feature independence and model linearity are two optional assumptions simplifying the computation of the expected values (note that $\bar{S}$ is the set of features not in S):

$$
\begin{aligned}
f(h_x(z')) &= E[f(z) \mid z_S] && \text{SHAP explanation model simplified input mapping} && (9)\\
&= E_{z_{\bar{S}}|z_S}[f(z)] && \text{expectation over } z_{\bar{S}} \mid z_S && (10)\\
&\approx E_{z_{\bar{S}}}[f(z)] && \text{assume feature independence (as in [9, 5, 7, 3])} && (11)\\
&\approx f([z_S, E[z_{\bar{S}}]]). && \text{assume model linearity} && (12)
\end{aligned}
$$

4.1 Model-Agnostic Approximations

If we assume feature independence when approximating conditional expectations (Equation 11), as in [9, 5, 7, 3], then SHAP values can be estimated directly using the Shapley sampling values method [9] or equivalently the Quantitative Input Influence method [3]. These methods use a sampling approximation of a permutation version of the classic Shapley value equations (Equation 8). Separate sampling estimates are performed for each feature attribution. While reasonable to compute for a small number of inputs, the Kernel SHAP method described next requires fewer evaluations of the original model to obtain similar approximation accuracy (Section 5).

Kernel SHAP (Linear LIME + Shapley values)

Linear LIME uses a linear explanation model to locally approximate f, where local is measured in the simplified binary input space. At first glance, the regression formulation of LIME in Equation 2 seems very different from the classical Shapley value formulation of Equation 8. However, since linear LIME is an additive feature attribution method, we know the Shapley values are the only possible solution to Equation 2 that satisfies Properties 1-3 – local accuracy, missingness and consistency. A natural question to pose is whether the solution to Equation 2 recovers these values. The answer depends on the choice of loss function L, weighting kernel $\pi_{x'}$ and regularization term Ω. The LIME choices for these parameters are made heuristically; using these choices, Equation 2 does not recover the Shapley values. One consequence is that local accuracy and/or consistency are violated, which in turn leads to unintuitive behavior in certain circumstances (see Section 5).

Below we show how to avoid heuristically choosing the parameters in Equation 2 and how to find the loss function L, weighting kernel $\pi_{x'}$, and regularization term Ω that recover the Shapley values.

Theorem 2 (Shapley kernel) *Under Definition 1, the specific forms of $\pi_{x'}$, L, and Ω that make solutions of Equation 2 consistent with Properties 1 through 3 are:*

$$\begin{aligned}\Omega(g) &= 0,\\ \pi_{x'}(z') &= \frac{(M-1)}{(M \text{ choose } |z'|)|z'|(M-|z'|)},\\ L(f,g,\pi_{x'}) &= \sum_{z'\in Z}\left[f(h_x^{-1}(z')) - g(z')\right]^2 \pi_{x'}(z'),\end{aligned}$$

where $|z'|$ is the number of non-zero elements in z'.

The proof of Theorem 2 is shown in the Supplementary Material.

It is important to note that $\pi_{x'}(z') = \infty$ when $|z'| \in \{0, M\}$, which enforces $\phi_0 = f_x(\varnothing)$ and $f(x) = \sum_{i=0}^{M}\phi_i$. In practice, these infinite weights can be avoided during optimization by analytically eliminating two variables using these constraints.

Since $g(z')$ in Theorem 2 is assumed to follow a linear form, and L is a squared loss, Equation 2 can still be solved using linear regression. As a consequence, the Shapley values from game theory can be computed using weighted linear regression.[2] Since LIME uses a simplified input mapping that is equivalent to the approximation of the SHAP mapping given in Equation 12, this enables regression-based, model-agnostic estimation of SHAP values. Jointly estimating all SHAP values using regression provides better sample efficiency than the direct use of classical Shapley equations (see Section 5).

The intuitive connection between linear regression and Shapley values is that Equation 8 is a difference of means. Since the mean is also the best least squares point estimate for a set of data points, it is natural to search for a weighting kernel that causes linear least squares regression to recapitulate the Shapley values. This leads to a kernel that distinctly differs from previous heuristically chosen kernels (Figure 2A).

4.2 Model-Specific Approximations

While Kernel SHAP improves the sample efficiency of model-agnostic estimations of SHAP values, by restricting our attention to specific model types, we can develop faster model-specific approximation methods.

Linear SHAP

For linear models, if we assume input feature independence (Equation 11), SHAP values can be approximated directly from the model's weight coefficients.

Corollary 1 (Linear SHAP) *Given a linear model $f(x) = \sum_{j=1}^{M} w_j x_j + b$: $\phi_0(f,x) = b$ and*

$$\phi_i(f,x) = w_j(x_j - E[x_j])$$

This follows from Theorem 2 and Equation 11, and it has been previously noted by Štrumbelj and Kononenko [9].

Low-Order SHAP

Since linear regression using Theorem 2 has complexity $O(2^M + M^3)$, it is efficient for small values of M if we choose an approximation of the conditional expectations (Equation 11 or 12).

[2] During the preparation of this manuscript we discovered this parallels an equivalent constrained quadratic minimization formulation of Shapley values proposed in econometrics [2].

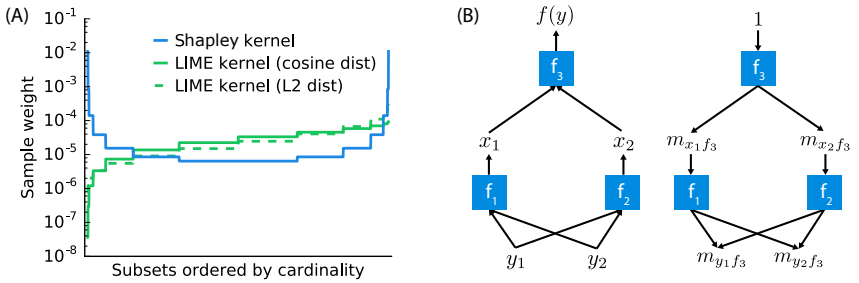


Figure 2: (A) The Shapley kernel weighting is symmetric when all possible z' vectors are ordered by cardinality there are 2^{15} vectors in this example. This is distinctly different from previous heuristically chosen kernels. (B) Compositional models such as deep neural networks are comprised of many simple components. Given analytic solutions for the Shapley values of the components, fast approximations for the full model can be made using DeepLIFT's style of back-propagation.

Max SHAP

Using a permutation formulation of Shapley values, we can calculate the probability that each input will increase the maximum value over every other input. Doing this on a sorted order of input values lets us compute the Shapley values of a max function with M inputs in $O(M^2)$ time instead of $O(M2^M)$. See Supplementary Material for the full algorithm.

Deep SHAP (DeepLIFT + Shapley values)

While Kernel SHAP can be used on any model, including deep models, it is natural to ask whether there is a way to leverage extra knowledge about the compositional nature of deep networks to improve computational performance. We find an answer to this question through a previously unappreciated connection between Shapley values and DeepLIFT [8]. If we interpret the reference value in Equation 3 as representing $E[x]$ in Equation 12, then DeepLIFT approximates SHAP values assuming that the input features are independent of one another and the deep model is linear. DeepLIFT uses a linear composition rule, which is equivalent to linearizing the non-linear components of a neural network. Its back-propagation rules defining how each component is linearized are intuitive but were heuristically chosen. Since DeepLIFT is an additive feature attribution method that satisfies local accuracy and missingness, we know that Shapley values represent the only attribution values that satisfy consistency. This motivates our adapting DeepLIFT to become a compositional approximation of SHAP values, leading to Deep SHAP.

Deep SHAP combines SHAP values computed for smaller components of the network into SHAP values for the whole network. It does so by recursively passing DeepLIFT's multipliers, now defined in terms of SHAP values, backwards through the network as in Figure 2B:

$$m_{x_j f_3} = \frac{\phi_i(f_3, x)}{x_j - E[x_j]} \tag{13}$$

$$\forall_{j \in \{1,2\}} \quad m_{y_i f_j} = \frac{\phi_i(f_j, y)}{y_i - E[y_i]} \tag{14}$$

$$m_{y_i f_3} = \sum_{j=1}^{2} m_{y_i f_j} m_{x_j f_3} \qquad \text{chain rule} \tag{15}$$

$$\phi_i(f_3, y) \approx m_{y_i f_3}(y_i - E[y_i]) \qquad \text{linear approximation} \tag{16}$$

Since the SHAP values for the simple network components can be efficiently solved analytically if they are linear, max pooling, or an activation function with just one input, this composition rule enables a fast approximation of values for the whole model. Deep SHAP avoids the need to heuristically choose ways to linearize components. Instead, it derives an effective linearization from the SHAP values computed for each component. The max function offers one example where this leads to improved attributions (see Section 5).

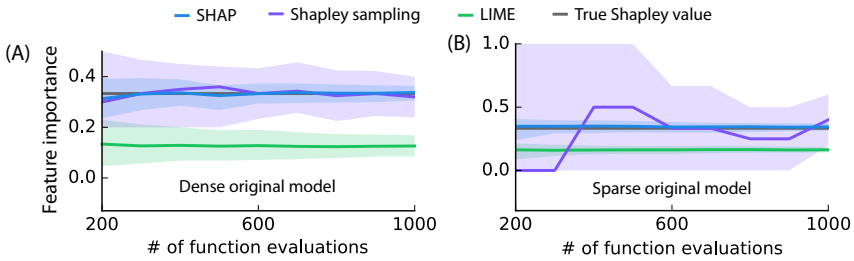


Figure 3: Comparison of three additive feature attribution methods: Kernel SHAP (using a debiased lasso), Shapley sampling values, and LIME (using the open source implementation). Feature importance estimates are shown for one feature in two models as the number of evaluations of the original model function increases. The 10th and 90th percentiles are shown for 200 replicate estimates at each sample size. (A) A decision tree model using all 10 input features is explained for a single input. (B) A decision tree using only 3 of 100 input features is explained for a single input.

5 Computational and User Study Experiments

We evaluated the benefits of SHAP values using the Kernel SHAP and Deep SHAP approximation methods. First, we compared the computational efficiency and accuracy of Kernel SHAP vs. LIME and Shapley sampling values. Second, we designed user studies to compare SHAP values with alternative feature importance allocations represented by DeepLIFT and LIME. As might be expected, SHAP values prove more consistent with human intuition than other methods that fail to meet Properties 1-3 (Section 2). Finally, we use MNIST digit image classification to compare SHAP with DeepLIFT and LIME.

5.1 Computational Efficiency

Theorem 2 connects Shapley values from game theory with weighted linear regression. Kernal SHAP uses this connection to compute feature importance. This leads to more accurate estimates with fewer evaluations of the original model than previous sampling-based estimates of Equation 8, particularly when regularization is added to the linear model (Figure 3). Comparing Shapley sampling, SHAP, and LIME on both dense and sparse decision tree models illustrates both the improved sample efficiency of Kernel SHAP and that values from LIME can differ significantly from SHAP values that satisfy local accuracy and consistency.

5.2 Consistency with Human Intuition

Theorem 1 provides a strong incentive for all additive feature attribution methods to use SHAP values. Both LIME and DeepLIFT, as originally demonstrated, compute different feature importance values. To validate the importance of Theorem 1, we compared explanations from LIME, DeepLIFT, and SHAP with user explanations of simple models (using Amazon Mechanical Turk). Our testing assumes that good model explanations should be consistent with explanations from humans who understand that model.

We compared LIME, DeepLIFT, and SHAP with human explanations for two settings. The first setting used a sickness score that was higher when only one of two symptoms was present (Figure 4A). The second used a max allocation problem to which DeepLIFT can be applied. Participants were told a short story about how three men made money based on the maximum score any of them achieved (Figure 4B). In both cases, participants were asked to assign credit for the output (the sickness score or money won) among the inputs (i.e., symptoms or players). We found a much stronger agreement between human explanations and SHAP than with other methods. SHAP's improved performance for max functions addresses the open problem of max pooling functions in DeepLIFT [7].

5.3 Explaining Class Differences

As discussed in Section 4.2, DeepLIFT's compositional approach suggests a compositional approximation of SHAP values (Deep SHAP). These insights, in turn, improve DeepLIFT, and a new version

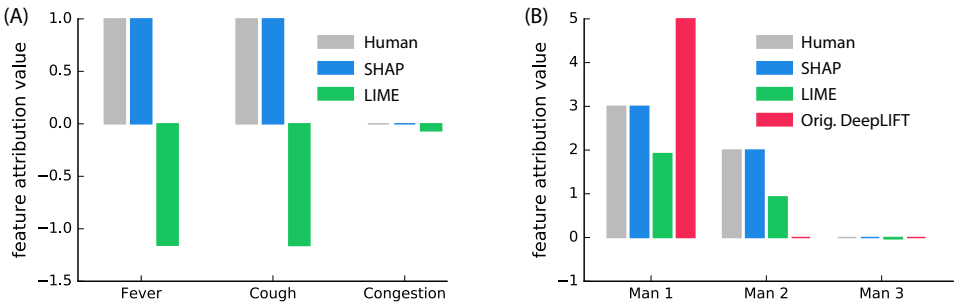


Figure 4: Human feature impact estimates are shown as the most common explanation given among 30 (A) and 52 (B) random individuals, respectively. (A) Feature attributions for a model output value (sickness score) of 2. The model output is 2 when fever and cough are both present, 5 when only one of fever or cough is present, and 0 otherwise. (B) Attributions of profit among three men, given according to the maximum number of questions any man got right. The first man got 5 questions right, the second 4 questions, and the third got none right, so the profit is $5.

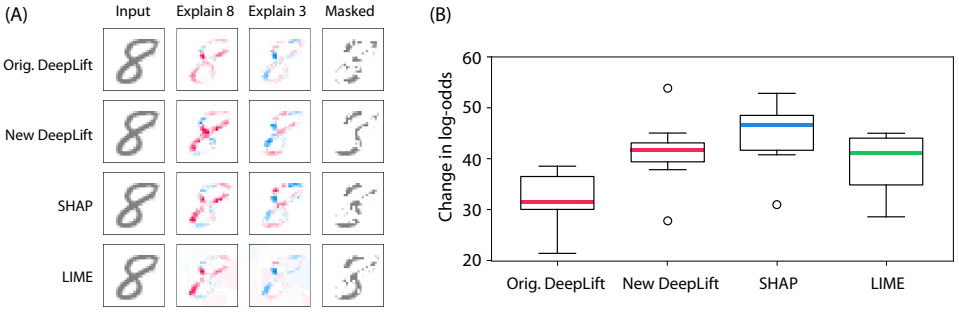


Figure 5: Explaining the output of a convolutional network trained on the MNIST digit dataset. Orig. DeepLIFT has no explicit Shapley approximations, while New DeepLIFT seeks to better approximate Shapley values. (A) Red areas increase the probability of that class, and blue areas decrease the probability. Masked removes pixels in order to go from 8 to 3. (B) The change in log odds when masking over 20 random images supports the use of better estimates of SHAP values.

includes updates to better match Shapley values [7]. Figure 5 extends DeepLIFT's convolutional network example to highlight the increased performance of estimates that are closer to SHAP values. The pre-trained model and Figure 5 example are the same as those used in [7], with inputs normalized between 0 and 1. Two convolution layers and 2 dense layers are followed by a 10-way softmax output layer. Both DeepLIFT versions explain a normalized version of the linear layer, while SHAP (computed using Kernel SHAP) and LIME explain the model's output. SHAP and LIME were both run with 50k samples (Supplementary Figure 1); to improve performance, LIME was modified to use single pixel segmentation over the digit pixels. To match [7], we masked 20% of the pixels chosen to switch the predicted class from 8 to 3 according to the feature attribution given by each method.

6 Conclusion

The growing tension between the accuracy and interpretability of model predictions has motivated the development of methods that help users interpret predictions. The SHAP framework identifies the class of additive feature importance methods (which includes six previous methods) and shows there is a unique solution in this class that adheres to desirable properties. The thread of unity that SHAP weaves through the literature is an encouraging sign that common principles about model interpretation can inform the development of future methods.

We presented several different estimation methods for SHAP values, along with proofs and experiments showing that these values are desirable. Promising next steps involve developing faster model-type-specific estimation methods that make fewer assumptions, integrating work on estimating interaction effects from game theory, and defining new explanation model classes.

Acknowledgements

This work was supported by a National Science Foundation (NSF) DBI-135589, NSF CAREER DBI-155230, American Cancer Society 127332-RSG-15-097-01-TBG, National Institute of Health (NIH) AG049196, and NSF Graduate Research Fellowship. We would like to thank Marco Ribeiro, Erik Štrumbelj, Avanti Shrikumar, Yair Zick, the Lee Lab, and the NIPS reviewers for feedback that has significantly improved this work.

References

[1] Sebastian Bach et al. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation". In: *PloS One* 10.7 (2015), e0130140.

[2] A Charnes et al. "Extremal principle solutions of games in characteristic function form: core, Chebychev and Shapley value generalizations". In: *Econometrics of Planning and Efficiency* 11 (1988), pp. 123–133.

[3] Anupam Datta, Shayak Sen, and Yair Zick. "Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems". In: *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE. 2016, pp. 598–617.

[4] Stan Lipovetsky and Michael Conklin. "Analysis of regression in game theory approach". In: *Applied Stochastic Models in Business and Industry* 17.4 (2001), pp. 319–330.

[5] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should i trust you?: Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2016, pp. 1135–1144.

[6] Lloyd S Shapley. "A value for n-person games". In: *Contributions to the Theory of Games* 2.28 (1953), pp. 307–317.

[7] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. "Learning Important Features Through Propagating Activation Differences". In: *arXiv preprint arXiv:1704.02685* (2017).

[8] Avanti Shrikumar et al. "Not Just a Black Box: Learning Important Features Through Propagating Activation Differences". In: *arXiv preprint arXiv:1605.01713* (2016).

[9] Erik Štrumbelj and Igor Kononenko. "Explaining prediction models and individual predictions with feature contributions". In: *Knowledge and information systems* 41.3 (2014), pp. 647–665.

[10] H Peyton Young. "Monotonic solutions of cooperative games". In: *International Journal of Game Theory* 14.2 (1985), pp. 65–72.

electronics

MDPI

Article

Explainable Quantum Neural Networks: Example-Based and Feature-Based Methods

Jinkai Tian [1,*] and Wenjing Yang [2,*]

[1] Intelligent Game and Decision Lab, Beijing 100071, China
[2] Department of Intelligent Data Science, College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China
* Correspondence: tianjinkai13@nudt.edu.cn (J.T.); wenjing.yang@nudt.edu.cn (W.Y.)

Abstract: Quantum neural networks (QNNs) are gaining attention for their potential, but their lack of interpretability remains a barrier to wider adoption. In this paper, we adapt and extend explainability techniques commonly used in classical neural networks to the quantum domain, making QNNs more transparent and interpretable. By applying both feature-based and example-based methods, we provide a comprehensive analysis of how QNNs generate predictions. Our results demonstrate that these adapted techniques offer valuable insights into the internal mechanisms of QNNs, paving the way for more reliable and trustworthy quantum machine learning models. This work contributes to improving the explainability of QNNs, enhancing their applicability in complex, real-world scenarios.

Keywords: quantum neural networks; explainable artificial intelligence; feature-based methods; example-based methods

Citation: Tian, J.; Yang, W. Explainable Quantum Neural Networks: Example-Based and Feature-Based Methods. *Electronics* **2024**, *13*, 4136. https://doi.org/10.3390/electronics13204136

Academic Editor: Wiesław Leonski

Received: 20 September 2024
Revised: 17 October 2024
Accepted: 19 October 2024
Published: 21 October 2024

1. Introduction

The rapid advancement of quantum computing has spurred significant interest in the field of quantum machine learning (QML) as a promising approach to leverage quantum algorithms in machine learning tasks [1]. Quantum computing models such as quantum support vector machines [2] and quantum kernel methods [3] have gained traction due to their solid theoretical foundations and potential computational advantages. While these models are often favored for their interpretability, their precision can be limited, particularly in the noisy intermediate-scale quantum era. In contrast, quantum neural networks (QNNs) have emerged as powerful tools capable of handling large-scale datasets and offering superior performance [4–6]. However, the increased complexity of QNNs presents a unique challenge in balancing interpretability with model performance.

QNN development has paralleled advancements in classical AI [4,7–11]. Despite the potential benefits QNNs offer, their internal processes remain obscure, raising questions about their learning mechanisms and ability to assimilate knowledge for prediction generation. Unlike classical models, where interpretability can often be traced to factors like linear weights or decision rules [12], quantum models are inherently more opaque. Quantum phenomena such as superposition and entanglement complicate interpretability, presenting barriers to their application in critical domains where transparency, reliability, and trust are essential.

The opacity of QNNs parallels concerns in classical machine learning, where deep neural networks (DNNs) have been criticized for their lack of transparency. DNNs, often labeled as "black boxes", have raised concerns in sensitive fields such as healthcare, finance, and autonomous systems, where understanding model decisions is crucial [13–16]. Consequently, researchers have focused on developing explainable artificial intelligence (XAI) techniques to provide insights into the inner workings of DNNs, enhancing their interpretability, trustworthiness, and compliance with regulatory standards [17–21].

In remote sensing, the volume and complexity of data generated by modern sensors pose significant challenges for data analysis and interpretation. QNNs have the potential to process large-scale remote sensing data more efficiently than classical models, but their opacity hinders their adoption in critical applications where interpretability is essential [22–25]. For instance, in disaster response scenarios, understanding why a model predicts certain areas as high risk is crucial for decision-makers. Similarly, in environmental monitoring, transparent models are needed to justify actions based on detected changes in land use or vegetation. Beyond remote sensing, other domains also benefit from explainable quantum models. In healthcare, QNNs could potentially analyze complex genomic data to identify disease markers, but clinicians require interpretable models to trust and act upon such predictions. In finance, quantum models might detect subtle patterns in market data for investment strategies, yet explainability is necessary to comply with regulatory standards and to gain user trust.

This paper advocates for the development of explainable quantum artificial intelligence (XQAI) to bridge the gap between quantum model performance and interpretability. XQAI aims to provide clear and interpretable insights into the predictions of QNNs, similar to the XAI techniques used for classical models. By enhancing the understanding of QNNs, XQAI can guide the future of quantum technologies and their application across a range of fields, from finance to healthcare.

QNNs, however, present distinct challenges. For example, they are prone to barren plateau problems [26], which make optimization difficult by flattening the loss landscape. Yet, a careful selection of QNN architecture and cost functions [27–29] can mitigate such issues. Moreover, efforts to make QNNs more interpretable must account for the specific quantum characteristics of the model and the dataset being used.

This paper contributes to advancing XQAI by exploring a comprehensive set of explainability methods applied to QNNs, integrating both example-based and feature-based approaches. These methods provide interpretable insights into QNN predictions, offering new tools to enhance the trustworthiness of quantum models. Figure 1 illustrates the fundamental concepts and differences between feature-based and instance-based interpretability methods. The trained quantum neural network is capable of processing various types of data inputs, including quantum data and classical data. The figure also summarizes the related methods and evaluation metrics of the explainable quantum artificial intelligence approaches proposed in this work. Among these, the methods in green font (occlusion analysis, deletion diagnostics) represent approaches that require modifying the model input or retraining the model. In contrast, the methods in blue font (gradient-based interpretability methods, influence functions) represent interpretability techniques that do not require changing the model input and rely solely on the model's internal gradient information.

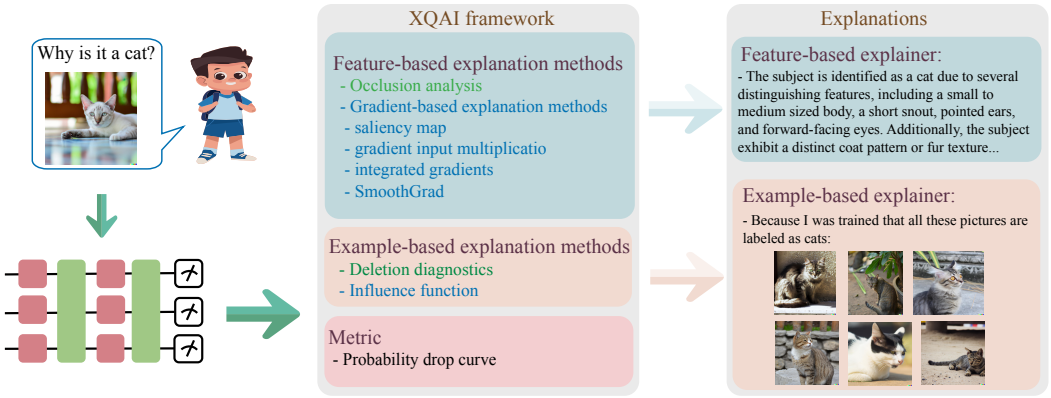


Figure 1. Comparison of feature-based and example-based explanation methods.

The primary contributions of this paper are as follows:

- A comprehensive analysis of a diverse set of explainability methods is conducted, selectively applying appropriate techniques to elucidate QNN models. By integrating both example-based and feature-based approaches, this study broadens the range of explanation techniques available in quantum machine learning.
- A quantitative evaluation of the proposed feature-based explanation techniques through the probability drop curve is provided. The findings reveal that QNNs are generally more challenging to explain than classical neural networks (NNs), highlighting the unique complexities posed by quantum models in explainable AI.
- The study also demonstrates that QNNs exhibit higher sensitivity to data compared to classical models. The interdependence of features in QNNs, due to quantum phenomena such as entanglement, makes them more reliant on complete data, thus emphasizing the need for robust and interpretable explanation techniques to better understand how quantum models process information.

The remainder of this paper is structured as follows. In Section 2, we introduce key concepts related to QNNs, explanation methods, and deep neural networks. Appendix A reviews the relevant literature. In Section 3, we discuss suitable explanation methods for quantum machine learning models and detail the metrics used to assess them. In Section 4, we present experimental results, followed by concluding remarks and suggestions for future work in Section 5.

2. Preliminary

2.1. Quantum Neural Networks

Quantum neural networks are a class of quantum algorithms that leverage quantum computation to process information and perform machine learning tasks. QNNs are implemented as hybrid quantum–classical algorithms, utilizing both quantum and classical resources. These algorithms consist of a quantum subroutine that evaluates an objective function and a classical subroutine that optimizes the parameters based on the quantum output. Hybrid quantum–classical algorithms are more resilient to noise and limitations in current quantum devices, as they require fewer quantum resources and employ shorter-depth circuits compared to fully quantum algorithms.

Qubits can be physically realized using various systems, such as superconducting circuits [30] and trapped ions [31]. Superconducting qubits, like transmons, are implemented using Josephson junctions, where quantum states are manipulated using microwave pulses. Trapped ion qubits use atomic ions confined by electromagnetic fields, with quantum operations performed using laser pulses. These physical systems support the quantum gates and operations used in QNNs, ensuring coherent quantum state evolution.

For a QNN with n qubits, the input is a quantum state typically prepared by encoding classical data $\boldsymbol{x}$ into a quantum state through an encoding method $|\psi_x\rangle = U(\boldsymbol{x})\,|0\rangle$. The parameterized quantum circuit (PQC) $U(\boldsymbol{\theta})$ with L layers is a sequence of quantum gates, typically consisting of single-qubit rotations and two-qubit entangling gates. In the hardware-efficient ansatz, $U(\boldsymbol{\theta})$ is constructed as a product of parameterized single-qubit gates and entangling operations as follows: $U(\boldsymbol{\theta}) = \prod_{i=1}^{L}\left(\prod_{k=1}^{n} R_z(\theta_k^{(i)})R_y(\theta_k^{(i)})\right)W$, where $R_z(\theta_k^{(i)})$ and $R_y(\theta_k^{(i)})$ represent parameterized single-qubit rotations around the z-axis and y-axis, respectively, for the k-th qubit in the i-th layer. The entangling operations are represented by W, typically consisting of controlled-NOT (CNOT) gates applied between neighboring qubits or according to some predefined topology. This ansatz allows the QNN to learn complex quantum correlations with relatively shallow circuits, making it suitable for near-term quantum devices.

The output of the PQC is the transformed quantum state $|\psi_y\rangle = U(\boldsymbol{\theta})\,|\psi_x\rangle$. After processing the input state, a quantum measurement is performed to extract classical information from the output state. The measurement yields the expectation value $\hat{y} = Q(x, y; \boldsymbol{\theta}) = \langle O_y\rangle = \langle\psi_y|O_y|\psi_y\rangle$, with O_y being the observable, which varies depend-

ing on the label under consideration. This result is used to estimate the label y associated with the input data.

Entanglement plays a fundamental role in quantum algorithms and has been crucial in achieving quantum speedups. For instance, the Deutsch–Josza algorithm [32], which provides exponential speedup over classical algorithms, relies on the quantum superposition and entanglement to evaluate Boolean functions efficiently. Shor's algorithm for factoring large integers [33] and Grover's search algorithm [34] also demonstrate the power of quantum entanglement and non-local operations, where multiple quantum states can be processed simultaneously, leveraging quantum parallelism. Similarly, the Harrow–Hassidim–Lloyd (HHL) algorithm [35] uses quantum entanglement to solve linear systems exponentially faster, leading to advances in quantum machine learning applications such as quantum support vector machines (QSVMs) [2] and quantum principal component analysis (QPCA) [36].

In the context of QNNs, quantum entanglement and non-local operations also play a key role in the network's ability to capture and process complex data patterns. These quantum phenomena allow QNNs to create correlations between qubits that cannot be replicated by classical models, which is particularly beneficial for high-dimensional and quantum-native datasets. Although QNNs may not yet have the same rigorous theoretical guarantees of quantum advantage as algorithms like Shor's, quantum entanglement and non-local operations remain essential for increasing the expressiveness and efficiency of QNNs, enabling them to leverage quantum superposition and non-classical correlations to solve certain tasks more effectively than their classical counterparts [3,37,38].

2.2. Data Encoding and Two-Qubit Operations

To process classical data using a QNN, it must first be encoded into a quantum state. This can be accomplished using various encoding methods, each offering distinct advantages and limitations. We briefly describe three commonly used encoding methods in the experiments.

Amplitude encoding encodes classical data into the amplitudes of a quantum state, requiring a number of qubits logarithmic with respect to the input data size. This makes it highly efficient for high-dimensional data. However, it is challenging to implement in practice due to the computational cost of state preparation. Furthermore, calculating gradients with respect to the input data, as required by gradient-based methods, is non-trivial.

In gate encoding, classical data are encoded into the parameters of Pauli rotation gates applied to a fixed initial quantum state, such as the computational basis state. This method offers flexibility and expressiveness but may require a large number of gates to represent complex data structures. Figure 2 illustrates data encoding on a three-qubit system using the R_x rotation gate. For a vector of length $3N$, N layers of interleaved rotation gates and two-qubit entanglement gates are required. The Pauli rotation gates R_x, R_y, R_z have a period of 4π, and their cyclic behavior ($R(\theta + 2\pi) = -R(\theta)$ for $R \in \{R_x, R_y, R_z\}$) makes the expected value of an observable periodic with 2π. Hence, input features should be normalized to the range $[0, 2\pi]$ using Min-Max scaling.

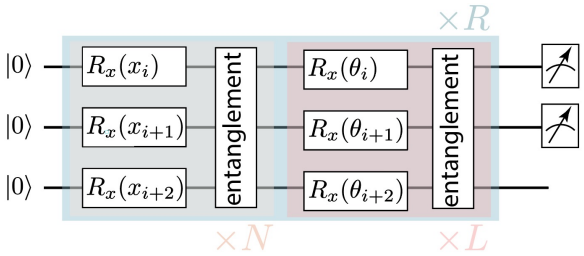


Figure 2. The circuit diagram of the QNN with data re-uploading.

Data re-uploading is a technique that encodes classical data by applying a series of gate encodings multiple times with intermediate parameter updates [39]. As shown in Figure 2,

data are encoded using the same mechanics as gate encoding, and the circuit is applied R times with trainable parameters. Re-uploading the data multiple times increases the expressiveness of the quantum circuit, potentially improving classification performance. However, this approach introduces additional computational overhead due to the repeated encoding. A summary of these encoding methods, highlighting their advantages, disadvantages, and use cases, is provided in Table 1.

Table 1. Comparison of encoding methods.

Method	Advantages	Disadvantages	Use Cases
Amplitude Encoding	Efficient for high-dimensional data	Computationally expensive state preparation; Non-trivial gradient calculation	Large-scale data processing requiring logarithmic qubit scaling
GateEncoding	Flexible and expressive; Adaptable to various quantum circuits	Requires many gates for complex data structures	Suitable for small- to medium-sized datasets
Data Re-uploading	Increases the expressiveness of the quantum circuit	Introduces additional computational overhead	Enhances classification performance in quantum circuits

This study compares three layouts of two-qubit operations: nearest-neighbor (NN), circuit-block (CB), and all-to-all (AA) layouts, as illustrated in Figure 3, following a similar approach to that proposed by [38].

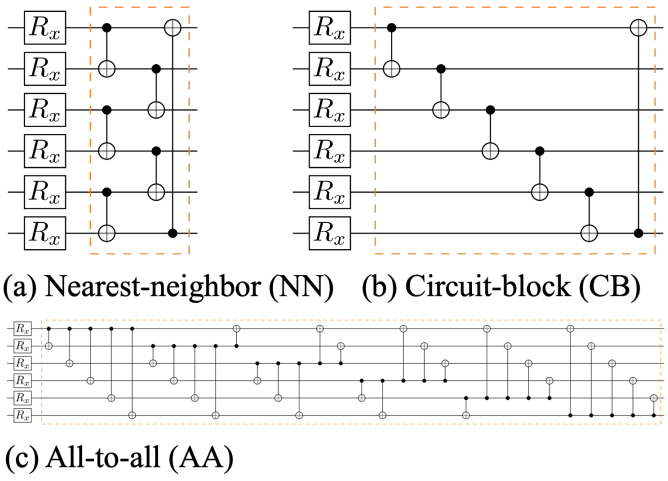


Figure 3. Three distinct types of two-qubit entangling operator layouts are presented: (**a**) nearest-neighbor (NN), (**b**) circuit-block (CB), and (**c**) all-to-all (AA). The two-qubit entangling operators are depicted within regions demarcated by an orange dashed box. Each layout corresponds to a quantum circuit depth of $2, n, n(n-1)$, respectively, thereby resulting in divergent computational challenges, deployment difficulties, and performance implications. These circuits are illustrated using the $\langle q|pic\rangle$ package [40].

2.3. Training and Optimization of QNNs

The objective of training a QNN is to optimize the parameters $\boldsymbol{\theta}$ in the PQC to minimize a loss function $\mathcal{L}(\boldsymbol{\theta}, (\boldsymbol{x}, y))$. The parameters $\boldsymbol{\theta}$ represent the trainable weights that control the quantum gate operations within the PQC. In hardware-efficient ansätze, $\boldsymbol{\theta}$ typically refers to the angles of rotation gates like $R_x(\theta)$, $R_y(\theta)$, and $R_z(\theta)$, and these parameters are usually restricted to the range $[0, 2\pi]$ to reflect the periodicity of the quantum operations.

The loss function $\mathcal{L}$ measures the difference between the predicted outcome $\hat{y}$ and the true outcome y. For classification tasks, cross-entropy loss is commonly used. It is defined as

$$\mathcal{L}(\boldsymbol{\theta}, (\boldsymbol{x}, y)) = -\sum_i y_i \log(\hat{y}_i), \tag{1}$$

where $\hat{y}_i$ is the predicted probability of class i, and y_i is the true label. For regression tasks, the mean squared error loss is typically used, while for quantum-specific tasks, such as unitary learning, fidelity distance is often employed to measure the similarity between predicted and actual quantum states.

Training a QNN requires optimizing the parameters $\boldsymbol{\theta}$ to minimize the selected loss function. This is achieved through optimization algorithms like gradient-based or zero-order methods. For a dataset $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, the objective is to minimize the empirical risk:

$$\min_{\theta} \mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) = \min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\boldsymbol{\theta}, (\boldsymbol{x}_i, y_i)). \tag{2}$$

In gradient-based optimization, the gradient of the loss function with respect to the parameters is calculated using techniques such as the parameter-shift rule [41,42], which allows for estimating gradients of quantum expectation values using only a fixed number of quantum evaluations. The parameters are updated iteratively using optimizers like gradient descent or its variants (e.g., Adam [43]):

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t, (\boldsymbol{x}_i, y_i)), \tag{3}$$

where η is the learning rate and t denotes the iteration step.

Decoherence is a significant challenge in QNNs, as the loss of quantum coherence can reduce the fidelity of the quantum states, leading to suboptimal or incorrect outputs. Non-ideal unitary gate operations, which occur due to hardware noise, further exacerbate this issue by introducing errors during the execution of quantum circuits. These factors complicate the training of QNNs, requiring error mitigation techniques and the design of noise-resilient quantum algorithms to maintain model accuracy in noisy environments [44].

While we have briefly touched on the impact of quantum noise on interpretability in this work, we recognize that a more detailed exploration is necessary. Future research will focus on the interplay between quantum noise and explainability, along with the development of noise-mitigation strategies to ensure the robustness and transparency of QNN models in practical applications.

2.4. Deep Neural Networks

Deep neural networks are a class of artificial neural networks composed of multiple layers of interconnected neurons. A DNN can be mathematically represented as a composition of several non-linear functions, with each function corresponding to a layer in the network.

Consider a DNN with L hidden layers. Each layer can be represented by a function f_l, where $l = 1, 2, \ldots, L$. The output of each layer is computed as the weighted sum of inputs from the previous layer, followed by the application of a non-linear activation function. Let x be the input vector, W_l the weight matrix, and b_l the bias vector for layer l. The output h_l of layer l can be expressed as

$$h_l = f_l(h_{l-1}; W_l, b_l) = g_l(W_l \times h_{l-1} + b_l), \tag{4}$$

where $h_0 = x$ is the input to the first layer, and g_l represents the activation function for layer l. Common activation functions include sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU).

The output of the final layer, L, is passed through an output activation function or decision function to generate the final prediction, y. In classification tasks, a softmax function is often used to produce class probabilities:

$$y = f_{L+1}(h_L; W_{L+1}, b_{L+1}) = \text{softmax}(W_{L+1} \times h_L + b_{L+1}), \quad (5)$$

where f_{L+1} represents the output function, and softmax normalizes the output into a probability distribution over the classes.

Training a DNN involves minimizing a loss function, $L(y, y')$, which quantifies the discrepancy between the predicted output y and the true target y'. The goal is to adjust the network's weights and biases using an optimization algorithm, such as gradient descent. During training, the gradients of the loss function with respect to the parameters (W_l and b_l) are computed using backpropagation, which applies the chain rule to efficiently calculate the necessary gradients.

In summary, a DNN is a composition of multiple non-linear functions, where each function corresponds to a layer in the network. It is trained using a loss function and optimization algorithm to minimize the error between predictions and target outputs.

3. Methods

Explanation methods vary in their applicability to QNNs. Some methods, like deletion diagnostics and influence functions, are effective with both classical and quantum datasets, making them suitable for example-based approaches. However, for feature-based methods, the situation is more complex. Techniques tailored for specific architectures, such as class activation mapping (CAM) for convolutional neural networks (CNNs), are not applicable to QNNs. Additionally, methods requiring information about intermediate quantum states, such as layer-wise relevance propagation, are impractical for QNNs due to the difficulty of calculating quantum state gradients.

Table 2 provides a summary of the explanation methods we will discuss in the following sections, categorized into feature-based and example-based approaches. We propose occlusion analysis and gradient-based explanation methods for QNNs, with the choice of data type and encoding method being crucial factors. For classical datasets, a QNN with gate encoding can be effectively analyzed using both occlusion and gradient-based methods. In contrast, amplitude encoding presents challenges for gradient-based methods due to the difficulty in computing gradients. For occlusion analysis, the lack of physical meaning when perturbing individual computational basis amplitudes further complicates its implementation. Similar challenges arise when deleting or perturbing the amplitudes of quantum data.

Table 2. Categories of explanation methods.

Category	Explanation Methods
Feature-based	Occlusion Analysis Saliency Map Gradient Input Multiplication Integrated Gradients SmoothGrad
Example-based	Deletion Diagnostics Influence Function

3.1. Feature-Based Explanation Methods

This section introduces two key feature-based methods: occlusion analysis and gradient-based approaches. Both provide valuable insights into the inner workings of deep learning models, each with distinct advantages and computational requirements.

3.1.1. Occlusion Analysis

Occlusion analysis involves systematically masking parts of the input and observing the resulting changes in the model's output. By analyzing how the occlusion affects the model's predictions, we can identify which regions of the input are most important for the model's decision-making process.

For example, consider a two-dimensional input $\boldsymbol{x}$ with dimensions $h \times w$. We use an occlusion window (mask) M with dimensions $h_M \times w_M$, applied to the input with a stride s. This generates a set of modified inputs, $\{\boldsymbol{x}_{p,q}\}$, where p, q index the position of the occlusion window. For each occluded input $I_{p,q}$, we calculate the QNN model's output, typically as the probability over the target class $P(y \mid \boldsymbol{x}_{p,q}; \theta) = \langle 0|U(\boldsymbol{x}_{p,q}, \boldsymbol{\theta})^{\dagger} O_y U(\boldsymbol{x}_{p,q}, \boldsymbol{\theta})|0\rangle$, where O_y is the observable for the target class, and $\boldsymbol{\theta}$ represents the model parameters.

The occlusion importance score $s^{OS}_{p,q}$ quantifies the impact of occlusion at position (p, q) and is computed as the difference between the original output and the occluded output:

$$s^{OS}_{p,q}(\boldsymbol{x}, y) = P(y \mid \boldsymbol{x}; \boldsymbol{\theta}) - P(y \mid \boldsymbol{x}_{p,q}; \boldsymbol{\theta}) \tag{6}$$

Larger positive values of $s^{OS}_{p,q}$ indicate that the occluded region is more important for the model's decision-making process. Visualizing the occlusion importance scores as a heatmap provides a clearer understanding of which areas in the input contribute most to the model's prediction, enhancing interpretability.

The number of occluded inputs to be processed is determined by the size of the occlusion window and the stride. For an input of dimensions $h \times w$ and an occlusion window of size $h_M \times w_M$, the total number of occluded inputs is given by

$$N = \left\lceil \frac{h - h_M}{s} \right\rceil \times \left\lceil \frac{w - w_M}{s} \right\rceil \tag{7}$$

Here, s is the stride with which the occlusion window is shifted across the input. For example, if the stride is 1, the window is moved one pixel at a time, resulting in a higher number of occluded inputs. Increasing the stride reduces the number of occluded inputs but may lead to a less granular analysis of feature importance.

For each occluded input, the model's output must be computed. The complexity of each forward pass of the QNN is $O(C)$, where C represents the number of measurements required to achieve a certain precision. Thus, the total complexity of occlusion analysis is $O(NC)$. This complexity depends on the input dimensions, the size of the occlusion window, the stride, and the number of measurements needed by the QNN. Consequently, this approach can be computationally expensive, especially for large inputs. Parallelizing the process can help mitigate the computational cost. On the other hand, gradient-based methods typically require one or a few backward passes and generally offer better computational efficiency.

3.1.2. Gradient-Based Explanation Methods

Gradient-based explanation methods leverage the gradients of the model's output with respect to the input features to provide explanations. The *saliency map* method is introduced here, with other gradient-based methods (e.g., gradient input multiplication, integrated gradients, and SmoothGrad) discussed in Appendix C.

Given a QNN model $Q(\boldsymbol{x}; \boldsymbol{\theta})$ mapping input $\boldsymbol{x}$ to a scalar output, the goal is to explain the model's prediction for a specific input x. The saliency map [45] is obtained by computing the gradients of the output with respect to the input features:

$$s^{VG}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} Q(\boldsymbol{x}; \boldsymbol{\theta}) \tag{8}$$

The resulting saliency map is a matrix with the same dimensions as the input, representing the importance of each feature in the model's prediction.

By visualizing the saliency map as a heatmap superimposed on the input, one can intuitively understand the contribution of each feature to the model's decision. Regions with higher intensity in the saliency map correspond to features with a greater impact on the model's output, while lower-intensity areas indicate less important features.

3.2. Example-Based Explanation Methods

3.2.1. Deletion Diagnostics

To define the influence of a specific training data point for a QNN, we consider the loss function $\mathcal{L}(\boldsymbol{\theta}, (\boldsymbol{x}_i, y_i))$, which quantifies the discrepancy between the true labels y_i and the QNN's predictions. A standard training scheme assigns equal weight to each training example, with the objective function given by

$$\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\boldsymbol{\theta}, (\boldsymbol{x}_i, y_i)) \tag{9}$$

To study the influence of each training data point on test samples, we modify the objective by re-weighting it with a vector $\boldsymbol{w}$:

$$L(\boldsymbol{w}) = \sum_{i=1}^{N} w_i \mathcal{L}(\boldsymbol{\theta}, (\boldsymbol{x}_i, y_i)) \tag{10}$$

where $\boldsymbol{w} = (w_1, \ldots, w_N)$ is the weight assigned to each training data point, with $\boldsymbol{w}_0 = (\frac{1}{N}, \ldots, \frac{1}{N})$ representing the standard training scheme. Denote $\boldsymbol{\theta}^*(\boldsymbol{w})$ as the model parameter retrained from scratch by minimizing $L(\boldsymbol{w})$.

Using deletion diagnostics, the influence of the training data point $(\boldsymbol{x}_i, y_i)$ on a test data point $(\boldsymbol{x}', y')$ is defined as

$$\mathcal{L}\big(\boldsymbol{\theta}^*(\boldsymbol{w}_0), (\boldsymbol{x}', y')\big) - \mathcal{L}\big(\boldsymbol{\theta}^*(\boldsymbol{w}_{-i}), (\boldsymbol{x}', y')\big) \tag{11}$$

where $\boldsymbol{w}_{-i} = (\frac{1}{N+1}, \ldots, 0, \ldots, \frac{1}{N+1})$ assigns zero weight to the i-th data point. If $(\boldsymbol{x}_i, y_i)$ has a positive influence on $(\boldsymbol{x}', y')$, the loss under $L(\boldsymbol{w}_0)$ is generally larger than that under $L(\boldsymbol{w}_{-i})$.

The process of retraining the model for each data point can be computationally prohibitive for large datasets, as it requires training multiple models from scratch.

3.2.2. Influence Function

The influence function, introduced by [46], provides an infinitesimal approximation to bypass the computational difficulties of deletion diagnostics. Assume that $\boldsymbol{\theta}$ is differentiable with respect to $\boldsymbol{w}$ at $\boldsymbol{w}_0$, and that $\mathcal{L}(\boldsymbol{\theta}, (\boldsymbol{x}, y))$ is twice differentiable in $\boldsymbol{\theta}$. The influence of $(\boldsymbol{x}_i, y_i)$ on the test data $(\boldsymbol{x}', y')$ is defined as

$$\begin{aligned} \mathcal{I}(\boldsymbol{x}_i, y_i; \boldsymbol{x}', y') &= -\left.\frac{\mathrm{d}\mathcal{L}(\boldsymbol{\theta}^*(\epsilon, i), (\boldsymbol{x}', y'))}{\mathrm{d}\epsilon}\right|_{\epsilon=0} \\ &= -\boldsymbol{v}'^{T} \left.\frac{\mathrm{d}\boldsymbol{\theta}^*(\epsilon, i)}{\mathrm{d}\epsilon}\right|_{\epsilon=0} \end{aligned} \tag{12}$$

where $\boldsymbol{v}' = \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^*(\epsilon, i), (\boldsymbol{x}', y'))$, and $\boldsymbol{\theta}^*(\epsilon, i) = \boldsymbol{\theta}^*(\boldsymbol{w}_0 + \epsilon \boldsymbol{e}_i)$, with $\boldsymbol{e}_i$ being the ith standard basis vector in the space of $\boldsymbol{w}$. Here, ϵ is a small perturbation, analyzed at zero for local behavior.

Applying the implicit function theorem yields the following:

$$\left.\frac{\mathrm{d}\boldsymbol{\theta}^*(\epsilon, i)}{\mathrm{d}\epsilon}\right|_{\epsilon=0} = -\mathcal{H}^{-1} \boldsymbol{v}_i \tag{13}$$

where $\mathcal{H}$ is the Hessian, or its quantum equivalent, $\frac{1}{n}\sum_i \nabla^2_{\theta}\mathcal{L}(\theta^*(\epsilon,i),(x_i,y_i))$, and $v_i = \nabla_{\theta}\mathcal{L}(\theta^*(\epsilon,i),(x_i,y_i))$. Estimating $\mathcal{H}$ in the quantum context can be challenging, but this approach offers computational benefits over retraining. Thus, the influence function is simplified to

$$\mathcal{I}(x_i,y_i;x',y') = v'^T\mathcal{H}^{-1}v_i \tag{14}$$

A limitation of influence functions is that they often identify outliers or mislabeled data as highly influential, making them suboptimal for explanations. To address this, the relative influence function distinguishes between global and local influence by assessing the local impact of an example on a prediction relative to its overall effect on the model [47]. The relative influence function is defined as

$$\mathcal{I}_r(x_i,y_i;x',y') = \frac{\mathcal{I}(x_i,y_i;x',y')}{|\mathcal{H}^{-1}v_i|} = \frac{v'^T\mathcal{H}^{-1}v_i}{|\mathcal{H}^{-1}v_i|} \tag{15}$$

Liu et al. [48] discusses the relationship between the influence function and the quantum Fisher information matrix. For example, using the fidelity distance as the cost function, the quantum Hessian is equivalent to the quantum Fisher information matrix. Fidelity distance is defined as $\mathcal{L}(\theta,(|\psi\rangle,|\phi\rangle)) = 1 - |\langle\phi|U(\theta)|\psi\rangle|^2$, where $|\langle\phi|U(\theta)|\psi\rangle|^2$ represents the fidelity between two pure states. While approximating the quantum Hessian matrix $\mathcal{H}$ is often difficult, techniques like those in [49] can be useful.

The method of encoding classical data into quantum states can also influence the impact of individual data points. Furthermore, the noisy nature of current quantum devices introduces potential errors in the computation of the influence function. Error mitigation techniques may be necessary to ensure accurate results. Despite these challenges, influence functions in QNNs can provide valuable insights into model behavior, helping to identify influential data points, detect anomalies, and improve the interpretability of quantum machine learning models.

While the influence function provides significant insights into the impact of individual data points, its computational complexity presents challenges, particularly as the number of qubits increases in larger QNN models. The core computational bottleneck lies in calculating the inverse of the Hessian matrix $\mathcal{H}$, which grows quadratically with the number of parameters in the quantum circuit. For QNNs with many parameters, especially those with deep quantum circuits or multiple layers, computing $\mathcal{H}^{-1}$ becomes prohibitively expensive in terms of both time and memory.

Moreover, the noisy nature of current quantum devices exacerbates the difficulty of accurately estimating gradients and Hessians. Noise introduces additional variance into the computation, further increasing the complexity of reliably calculating influence functions in practical quantum settings. To mitigate this, noise-resilient methods and error-mitigation strategies must be integrated into the calculation process. These may include techniques like Richardson extrapolation and error correction codes, which can reduce the impact of noise on the gradients and Hessians.

While influence functions offer a valuable method for explainability in QNNs, their applicability to larger models is limited by computational complexity and noise. Future research should focus on developing scalable and noise-tolerant techniques for calculating influence functions in large-scale quantum systems.

3.3. Metrics: Faithfulness

Achieving an unbiased assessment of an explanation's quality is critical in practice. However, evaluating explanations is often challenging due to the lack of universally accepted "ground truth" explanations. The concept of human explainability remains ambiguous and difficult to define [21]. Users' ability to comprehend explanations and discern underlying features may vary significantly depending on their expertise. For instance, a non-expert may prefer a simple visual representation, while an expert might seek a more detailed explanation with precise scientific terminology.

A key criterion for evaluating an explanation is how accurately and comprehensively it represents the local decision structure of the quantum neural network (QNN) model being analyzed. One practical approach to assess this is by examining how the removal of features identified by the explanation leads to a significant decrease in the model's predictive capabilities [50]. This method involves iteratively eliminating input features, starting with the most relevant, and monitoring changes in the model's output. The changes in prediction scores can be visualized as a score drop curve. This curve can be calculated for a single instance or averaged across an entire dataset to estimate the faithfulness of the explanation algorithm on a global scale.

In the case of QNNs, where the output is the probability of certain computational bases, the probability drop curve is used as a metric. Two key indicators on the curve warrant attention: (a) the steepness of the initial descent and (b) the minimum value reached by the curve. A steeper initial drop and a smaller minimum value suggest that the explanation method more faithfully reflects the network's decision-making process.

An alternative evaluation metric involves taking the result of one explanation method as the ground truth and calculating the faithfulness of other methods relative to it. Since the result of occlusion analysis tends to be stable and can reflect true feature importance, we use it as the ground truth. We then calculate the rank correlation of feature order results from gradient-based methods. By comparing these rank correlations, we can evaluate the relative performance of various explanation methods and assess their ability to provide meaningful insights into the QNN's decision-making process. This method helps researchers and practitioners select the most suitable explanation techniques for enhancing the interpretability and trustworthiness of quantum machine learning models.

3.4. Interplay Between XQAI and Quantum Adversarial Machine Learning

XQAI methods and quantum adversarial examples [51] are closely linked in their shared goal of developing robust, transparent, and trustworthy quantum machine learning models. The connection between XQAI and adversarial examples lies in their respective roles in improving the understanding and resilience of quantum models.

XQAI methods seek to offer valuable insights into the complex mechanisms of quantum machine learning models. In contrast, adversarial examples are intentionally crafted inputs designed to mislead a machine learning model into making incorrect predictions or classifications. These examples exploit weaknesses in the model's decision boundaries, exposing vulnerabilities and demonstrating a lack of robustness. Adversarial examples have been shown to be effective against both quantum classifiers and DNNs [52–56].

3.4.1. Feature-Based Explanation Methods

As depicted in Figure 4, feature-based explanation methods can be used to generate adversarial examples. Experiments in Section 4 demonstrate that masking approximately 10% of pixels, selected based on feature-based explanation methods, results in a prediction probability smaller than a random guess. This suggests that adversarial examples can be created by targeting key features, as identified by explanation methods.

3.4.2. Example-Based Explanation Methods

While feature-based explanation methods can generate adversarial test examples that deceive models, adversarial training examples can also be crafted to manipulate predictions while remaining visually indistinguishable [57]. Influence functions can determine how to subtly perturb training data to maximize loss on targeted test examples, leading to flipped predictions.

For a target test data point $(\boldsymbol{x}', y')$, an adversarial version of a training data point $(\boldsymbol{x}_i, y_i)$ is initialized as $(\tilde{\boldsymbol{x}}_i, \tilde{y}_i) = (\boldsymbol{x}_i, y_i)$. Small perturbations are added iteratively, informed by the influence function $\mathcal{I}$, to maximize the loss on the test data while preserving its visual appearance:

$$\tilde{\boldsymbol{x}}_i := \tilde{\boldsymbol{x}}_i + \alpha \operatorname{sign}\left(\mathcal{I}\left(\tilde{\boldsymbol{x}}_i, \tilde{y}_i; \boldsymbol{x}', y'\right)\right) \tag{16}$$

where α is a small step size and

$$\mathcal{I}(\tilde{x}_i, \tilde{y}_i; x', y') = -v'^T \mathcal{H}^{-1} \nabla_x \tilde{v}_i, \tag{17}$$

with $\tilde{v}_i = \nabla_\theta \mathcal{L}(\theta^*(\epsilon, i), (x_i, y_i))$. The model is retrained after each iteration. Though perturbations are small in pixel space, they have a significant impact in feature space.

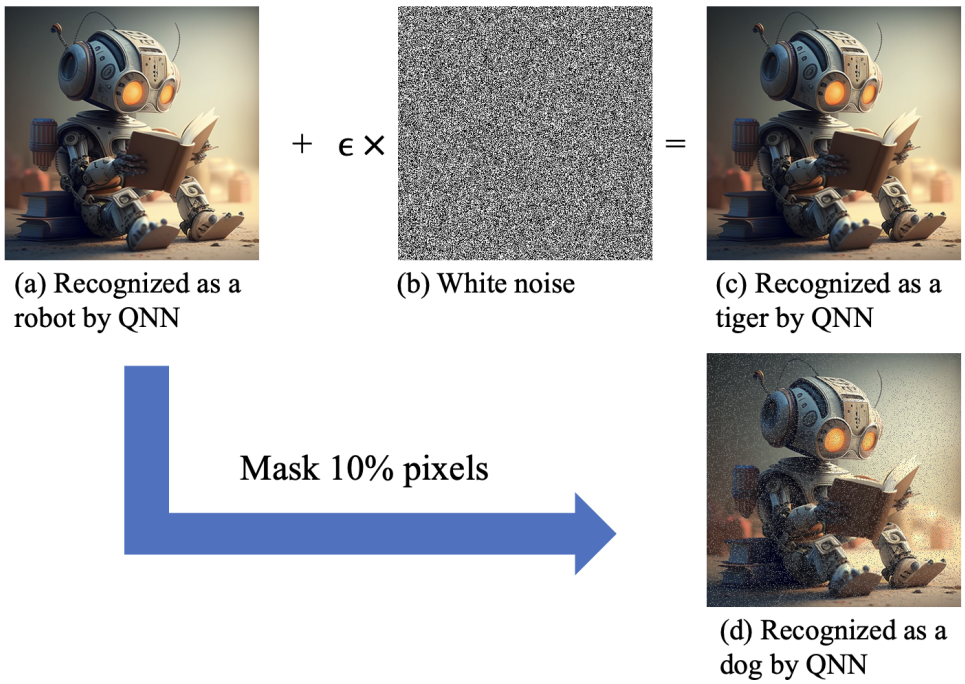


Figure 4. A schematic illustration of generating adversarial examples using adversarial machine learning methods and masking based on the outcomes of feature-based explanation methods. (**a**) The original image I, identified by a QNN as a robot; (**b**) a random noise image I_r; (**c**) the adversarial image $I_a = I + \epsilon I_r$, identified by a QNN as a tiger, where $\epsilon = 0.004$; (**d**) the image after masking 10% of pixels selected by feature-based explanation methods, identified by a QNN as a dog.

This method is mathematically equivalent to previous gradient-based attacks on training sets [53,54] but requires fewer visually noticeable changes. It demonstrates how minimally perturbed adversarial training examples can be generated to manipulate model predictions on targeted test data. Models that rely heavily on a small number of highly influential data points are most vulnerable to such attacks. Evaluating the extent to which subtly perturbing training examples affect loss on targeted test examples helps gauge model robustness against data poisoning attacks.

The relationship between XQAI and adversarial examples highlights several critical aspects of quantum machine learning models. XQAI can expose weaknesses in a model's decision-making process, revealing potential vulnerabilities that adversarial examples might exploit. This understanding of model weaknesses allows researchers to develop more robust models capable of resisting such attacks. Additionally, XQAI methods can be employed to detect and assess the impact of adversarial examples on model predictions by analyzing discrepancies in explanations generated for original versus adversarial inputs. These insights can guide the creation of effective defenses against adversarial attacks. Ultimately, ensuring the model's resilience to adversarial examples while maintaining transparency in its decision-making process is essential for fostering trust in AI systems. XQAI contributes to this trust by offering interpretable insights into the internal workings of models, thereby supporting both robustness and transparency.

4. Experiment

In this section, we report an empirical investigation conducted utilizing XQAI to compare the performance and interpretability of QNNs and DNNs. Specifically, this study integrated explainability into QNNs for multi-class classification tasks using the widely recognized Iris and Digit datasets. The outcomes of the QNNs were juxtaposed with those derived from classical NNs. Models demonstrating exceptional performance were selected, and the results of applying both example-based and feature-based explanation methodologies are presented. To facilitate a quantitative comparison of feature-based explanation techniques, we propose using the probability drop curve and rank correlations as metrics to evaluate the faithfulness of the respective methods.

The QNN architecture used in this experiment comprises multiple layers of single-qubit rotation gates, parameterized by trainable weights, and entangling operators. Entanglement is achieved through two-qubit operations, specifically using CNOT gates. This study compares three layouts of two-qubit operations: nearest-neighbor (NN), circuit-block (CB), and all-to-all (AA) layouts, as illustrated in Figure 3. With CNOT gates fixed as the entangling operator, we investigated the performance of three Pauli rotation gates (R_x, R_y, and R_z). The results indicated that models employing R_y gates outperformed those using R_x gates, while models with R_z gates failed to learn due to the commutation of $\sigma_z \otimes I$ with the CNOT gate. Consequently, R_y was chosen as the default single-qubit rotation gate. The QNN's output is a probability distribution over the classes. The first few qubits are measured, with each category corresponding to a distinct computational basis.

For comparison, we also implemented multilayer perceptron (MLP) and convolutional neural network (CNN) models. To ensure a fair comparison, classical feature selection techniques were not applied to reduce the number of features input into the QNN. Both QNNs and classical NNs were trained using the Adam optimizer, with a learning rate of 0.001 and a batch size of 4, for 100 epochs. Alternative numerical methods, such as genetic algorithms [58], can also be used for optimizing quantum systems.

4.1. Evaluating Explanation Methods on Classical Benchmarks

For the classical datasets, we used the following benchmark datasets:

- Iris Dataset: This dataset contains 150 instances, each with four features representing the length and width of the sepal and petal of Iris flowers. There are three classes, each corresponding to a different species of Iris: setosa, versicolor, and virginica.
- Digit Dataset: This dataset consists of 1797 instances, where each instance is an 8×8 grayscale image of a handwritten digit (0–9). For this experiment, we selected images of {0, 1, 2, 3} to perform a four-class classification task. The images are represented as 64-dimensional feature vectors.

The Iris and Digit datasets are publicly available and can be accessed through the UCI Machine Learning Repository and the Scikit-Learn library, respectively. Both datasets were divided into training (60%) and testing (40%) sets.

Tables 3 and 4 provide comparative evaluations of model performance on the Iris and Digit datasets, ranked by test accuracy. The models evaluated include the Hardware Efficient Ansatz (HE), Re-uploading Hardware Efficient Ansatz (Re), MLP, and CNN. Data for QNNs are gate-encoded using R_y rotation gates. The Hardware Efficient Ansatz is applied to 6-qubit quantum circuits with $L = 30$. The Re-uploading Hardware Efficient Ansatz is applied to 6-qubit quantum circuits with $L = 30$ and $R = 2$. "Layout" refers to the layout of two-qubit operations, while "Cost" indicates the cost function employed: cross-entropy (CE) and fidelity distance (FD). "Train Acc" and "Test Acc" represent training and testing accuracies, respectively. Accuracy is calculated as the ratio of correct predictions to the total number of predictions, evaluated separately for the training and test datasets. This metric provides a straightforward measure of model performance across different architectures and datasets. For the QNN and classical NN models with the highest test accuracy on each dataset, we evaluate the quality of the explanations generated by our XQAI framework.

Table 3. Performance on Iris dataset. The MLP has four layers with $[4, 8, 8, 3]$ neurons.

Model	Layout	Cost	Train Acc	Test Acc
MLP	/	CE	0.947	0.987
Re	NN	FD	0.973	0.973
Re	CB	CE	0.973	0.947
Re	NN	CE	0.973	0.947
Re	CB	FD	0.987	0.947
Re	AA	CE	0.973	0.947
Re	AA	FD	0.973	0.947
HE	CB	CE	0.973	0.920
HE	NN	CE	0.973	0.920
HE	CB	FD	0.933	0.907
HE	AA	CE	0.960	0.907
HE	NN	FD	0.920	0.880
HE	AA	FD	0.933	0.880

Table 4. Performance on Digit dataset. The MLP has three layers with $[64, 6, 4]$ neurons. The CNN model has four layers with $[1, 4, 2, 2]$ channels.

Model	Layout	Cost	Train Acc	Test Acc
MLP	/	CE	1.000	0.994
Re	NN	CE	1.000	0.983
Re	CB	CE	0.997	0.978
HE	AA	CE	0.983	0.972
Re	CB	FD	0.983	0.964
HE	CB	CE	0.981	0.956
Re	NN	FD	0.983	0.953
HE	NN	CE	0.967	0.944
HE	AA	FD	0.953	0.944
HE	CB	FD	0.964	0.936
HE	NN	FD	0.931	0.933
CNN	/	CE	0.992	0.922

4.1.1. Example-Based Explanation Methods

In the analysis of the Digit dataset, as illustrated in Figure 5, the relative influence function appears to provide more consistent and meaningful explanations compared to the standard influence function. This is particularly evident in the Re-uploading Hardware Efficient Ansatz, which outperforms the MLP in providing more reliable and interpretable top-5 examples. The superior performance of the relative influence function can be attributed to its ability to capture local influences in the context of the model's decision-making process, distinguishing between the examples that truly impact the model's predictions and those that might be misleading.

This result highlights the importance of selecting appropriate example-based methods, especially in quantum models like QNNs, where the decision boundaries may differ significantly from classical models. The ability of the Re-uploading Ansatz to better align its predictions with the test data through example-based methods suggests that quantum models might be more sensitive to specific influential data points, offering opportunities for more robust interpretations of quantum-based decisions.

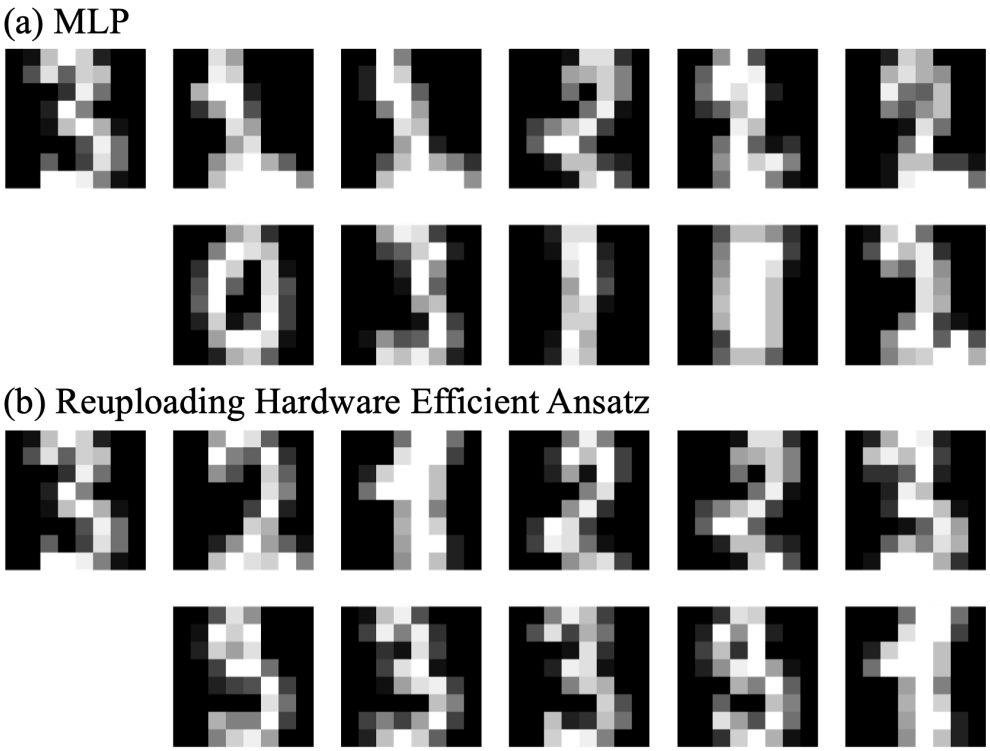


Figure 5. Explanations of a digit figure depicting the number 3. The top row shows the test data and the top-5 examples found by the influence function. The second row displays the top-5 examples found by the relative influence function.

4.1.2. Feature-Based Explanation Methods

The comparison of the five feature-based explanation methods, as shown in Figure 6, reveals key differences in their ability to generate interpretable heatmaps. Occlusion analysis, gradient input multiplication, and integrated gradients produce clearer and more detailed heatmaps, successfully identifying the most important regions of the input that influence the model's predictions. This indicates that these methods are better suited for capturing feature importance in both QNN and classical NN models.

In contrast, the saliency map and SmoothGrad methods generate less interpretable and more diffuse heatmaps, particularly on the Re-uploading Hardware Efficient Ansatz. This observation suggests that gradient-based methods like the saliency map, while computationally efficient, may struggle with noisy or less smooth decision boundaries in QNN models. The use of SmoothGrad, which averages gradients over multiple noisy versions of the input, does not seem to sufficiently address this issue, as the resulting heatmaps remain somewhat unclear.

4.1.3. Probability Drop Curves

The average probability drop curves, as presented in Figure 7, provide a quantitative assessment of the five feature-based explanation methods. The sharp decline in probability when masking the most important features suggests that all five methods effectively identify influential features. Masking approximately 10% of the features leads to a probability drop that approaches random guessing, underscoring the high sensitivity of both QNNs and classical NNs to a small subset of critical features.

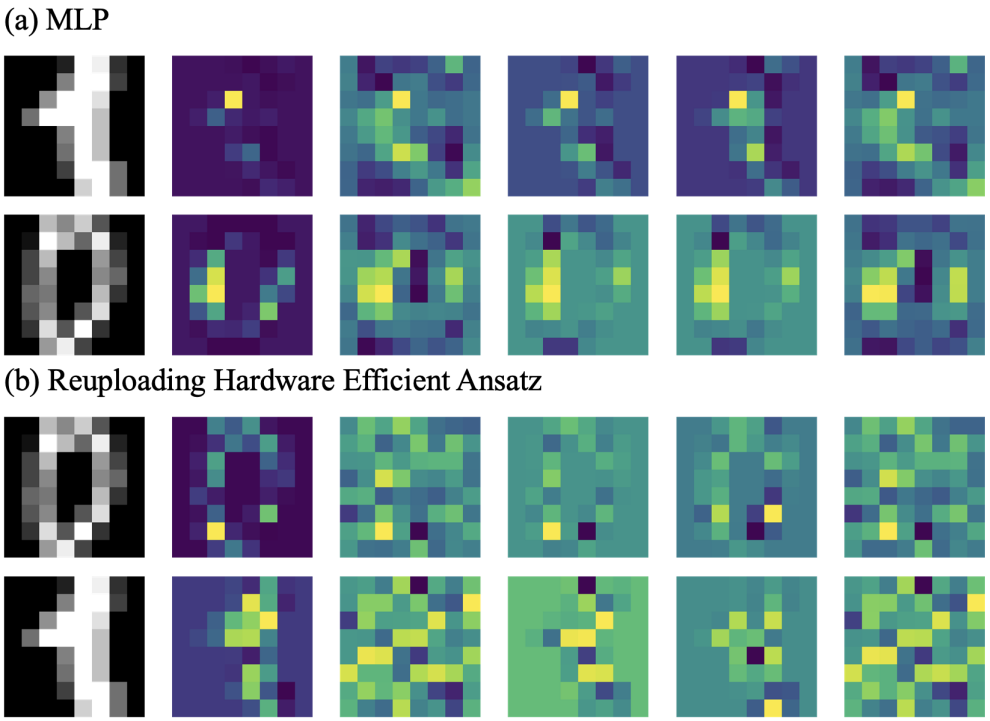
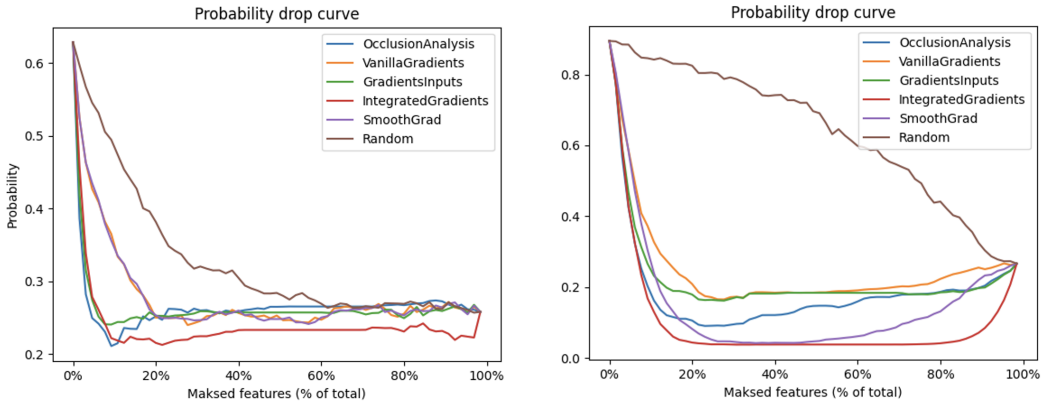


Figure 6. Explanations of two digit figures on Re-uploading Hardware Efficient Ansatz and MLP models. From left to right: original image, occlusion analysis, saliency map, gradient input multiplication, integrated gradients, and SmoothGrad heatmaps. In each heatmap, darker colors represent lower influence, while brighter colors indicate higher influence on the model's decision-making process.



(a) Reuploading Hardware Efficient Ansatz (b) MLP

Figure 7. Average probability drop curves for the five feature-based explanation methods on the Digit dataset for (**a**) Re-uploading Hardware Efficient Ansatz and (**b**) MLP models.

The minimum values of the curves, which are lower than the probability when all features are removed, offer further insights. This phenomenon can be attributed to the fact that the features are ranked by importance, with the most significant ones masked first. Masking these important features leads to a rapid decrease in probability. Conversely, removing the least significant features might sometimes slightly increase the probability, likely due to their negative contribution or noise-like influence. This finding highlights the importance of feature ranking in interpretability and suggests that not all features contribute positively to the model's decision.

Furthermore, the results demonstrate that QNNs rely more heavily on complete data than MLPs, as indicated by the steeper probability drop when features are randomly removed. This steeper decline suggests that QNNs encode information more holistically, meaning the removal of even a few seemingly less significant features can have a more

pronounced impact on the model's output. The heightened sensitivity in QNNs can be explained by their more refined feature interactions. In QNNs, the relationships between features are often encoded in a complex, interdependent manner due to quantum phenomena like entanglement, which allows QNNs to capture more intricate patterns. While this refined feature interaction gives QNNs an advantage in leveraging all input features for prediction, it also makes them more vulnerable to the removal of any feature, as the intricate connections among features can be disrupted. This behavior suggests both a strength—through more sophisticated data encoding—and a potential limitation, as QNNs may be more susceptible to noise or missing data.

This analysis reinforces the need for robust interpretability methods, particularly for QNNs, where understanding the role of each feature is crucial to the model's overall performance. Recognizing how QNNs utilize input features differently from classical models like MLPs can guide strategies to enhance their resilience and applicability in real-world scenarios.

5. Conclusions

In this paper, we present an XQAI framework that integrates explainability into QNNs for multi-class classification tasks. We evaluate the performance of QNNs and classical NNs using the Iris and Digit datasets. Our results demonstrate that certain QNN configurations achieve performance that is comparable to or even exceeds that of classical NNs. Through the application of example-based and feature-based explanation methods, we demonstrate the effectiveness of our XQAI framework in providing insights into both QNN and classical NN models. The probability drop curve, proposed as a metric for evaluating the faithfulness of feature-based explanation methods, offers a meaningful way to assess these techniques.

While this work demonstrates that adapting classical XAI techniques for QNNs provides useful insights, there are inherent limitations in applying these methods directly to quantum systems. Quantum-specific phenomena, such as entanglement and superposition, introduce complexities not encountered in classical models. This complicates the interpretation of certain features and requires further adaptation of classical techniques to account for the unique properties of quantum models. Future work should focus on developing XAI methods specifically designed for quantum neural networks, addressing the challenges posed by quantum phenomena.

Future research should explore additional datasets, model architectures, and explanation methods to further validate the efficacy of our XQAI framework. Moreover, the development of new metrics to evaluate the interpretability and faithfulness of explanation methods would advance the field of XQAI, both in quantum and classical machine learning.

Author Contributions: Conceptualization, J.T. and W.Y.; methodology, J.T. and W.Y.; software, J.T.; validation, J.T. and W.Y.; formal analysis, J.T. and W.Y.; investigation, J.T.; resources, W.Y.; data curation, J.T.; writing—original draft preparation, J.T. and W.Y.; writing—review and editing, J.T. and W.Y.; visualization, J.T.; supervision, W.Y.; project administration, W.Y.; funding acquisition, W.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China (No. 62372459, No. 62376282, No. 91948303).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Related Works

Appendix A.1. XAI Methods

Neural networks, while highly effective, often suffer from a lack of interpretability. This has prompted researchers to develop XAI techniques to provide insights into the inner workings of these models. XAI methods play a crucial role in analyzing the influence of individual training data points on model predictions or parameters, enhancing interpretability and trustworthiness in machine learning models. These methods allow researchers to better understand model behavior, identify influential samples, and address potential issues in the training dataset, ultimately improving the reliability of neural networks across various applications.

Appendix A.1.1. Feature-Based Explanation Methods

Many XAI methods aim to elucidate a model's decision-making process by assigning importance values to input features, thus highlighting their contributions to specific predictions. These methods provide diverse approaches to interpreting and understanding the inner workings of neural networks, enhancing transparency and fostering confidence in their predictions. Interpretability in XAI can be categorized into global interpretability and local interpretability.

Global interpretability pertains to understanding the overall model behavior and the relationships between input features and predictions. For example, linear models, such as linear regression or logistic regression, inherently offer global interpretability, as their coefficients can be directly interpreted. A positive coefficient for a feature signifies a positive relationship between the input and output, while a negative coefficient indicates an inverse relationship. In more complex models, such as random forests [59], global feature importance can be assessed by determining the contributions of each feature to the overall model. Higher importance scores suggest a stronger influence on the model's predictions. Feature importance ranking involves iteratively training the model while removing or shuffling features to evaluate their impact on the model's performance, thus ranking features based on their significance to the model's decision-making process.

Local interpretability aims to explain individual predictions by identifying the input features or regions most responsible for a specific output. In image classification, deep learning models, such as CNNs, can utilize saliency maps [45] for local interpretability. Saliency maps highlight the most influential regions of an input image that contribute to the model's prediction. Class activation mapping [60] is another technique that visualizes the significance of different regions in an image for a specific class by leveraging the activation maps of the last convolutional layer to produce a heatmap, pinpointing regions that contribute most to the predicted class.

Layer-wise relevance propagation [61] is a technique that explains the predictions of deep learning models by attributing relevance scores to each input feature. These scores are computed by propagating information backward from the output layer through the network to the input layer. LRP helps in understanding how the model processes information hierarchically and identifies the most influential features for a given prediction.

Local interpretable model-agnostic explanations (LIME) [61] explains the predictions of any classifier by locally approximating it with an interpretable model around the prediction. This is achieved by perturbing the input data, generating new samples, and training an interpretable model (e.g., linear regression or decision trees) on these samples to explain individual predictions.

Deep Learning Important Features (DeepLIFT) [62] is a feature attribution method that assigns contribution scores by comparing the activation of each neuron to a reference activation and propagating these scores through the network to compute feature importance. SHapley Additive exPlanations (SHAP) [63], based on cooperative game theory, assigns Shapley values to each input feature, providing consistent and locally accurate explanations for any model. SHAP allows for an equitable distribution of contributions from each feature to the model's prediction.

Counterfactual explanations [16] offer insights by generating alternative input instances that would lead to different predictions. This approach helps identify the minimal changes required to the input to alter the model's prediction, highlighting critical features. Activation maximization [64] generates synthetic inputs that maximize the activation of specific neurons or classes, providing insights into the learned features and decision boundaries of the neural network.

Appendix A.1.2. Example-Based Explanation Methods

Example-based explanation methods [65] focus on understanding the effects of individual training data points on model predictions or parameters. These methods provide various approaches to analyzing the impact of each training point on a model's performance, thereby improving the understanding of model behavior, identifying influential data points, and detecting issues within the training dataset.

Deletion diagnostics is a technique in which one training sample is removed at a time, and the model is retrained on the remaining samples. The model's performance is then evaluated to determine the impact of removing each data point, helping identify influential training examples.

Cook's distance [66], commonly used in linear regression, measures the influence of each data point by quantifying the change in regression coefficients when a data point is removed. Although originally developed for linear regression, the concept of assessing the impact of each data point on model parameters can be extended to other models.

The influence functions, from robust statistics [46], have been adapted to measure the influence of individual training examples on model parameters and predictions. [57] demonstrates how influence functions can be used to identify influential training data, understand model behavior, and detect issues such as mislabeled data or biased samples.

Data Shapley [67] is a method that applies Shapley values from cooperative game theory to measure the contribution of each data point to model performance. By attributing Shapley values to each training sample, Data Shapley provides a principled way to quantify the value of each data point for training the model.

TracIn [68] leverages gradient information during training to estimate the importance of each data point based on how its gradient aligns with the final parameter update direction, allowing for a fine-grained assessment of data point importance.

Appendix A.2. XQAI Methods

Despite growing interest in QML, research on explainable quantum AI (XQAI) remains limited. [69] explored explainability in parameterized quantum circuits by adapting Shapley values to the quantum realm. However, their work focused primarily on the substructures of quantum circuits, which is distinct from the core focus of our paper. Furthermore, [70] adapted Shapley values to quantum machine learning by evaluating the marginal contribution of features in quantum models. They proposed a framework where quantum measurements, treated as probabilistic outcomes, are integrated into Shapley value calculations, providing an effective way to quantify the importance of individual features in QNNs. While we do not introduce new experiments related to Shapley values, these recent advancements show the potential of Shapley values in quantum contexts and support the theoretical basis for their application in our future work. [71] demonstrated that the output of QNNs can be represented as truncated Fourier series. Building on this, [72] accelerated model-agnostic explanation methods, such as integrated gradients and SHAP, for QNN models.

Appendix B. Clever Hans Effect

Evaluating an AI system requires not only measuring task performance but also understanding its decision-making process. Traditional metrics, like classification accuracy and rewards in reinforcement learning, show how well a model performs but do not reveal

why it makes certain decisions. This is critical in determining whether the model has learned meaningful, generalizable patterns or is relying on superficial correlations.

The Clever Hans effect [73] exemplifies this issue, where a model's success is based on spurious correlations rather than genuine understanding. As shown in Figure A1, this leads to predictions that, while accurate in limited conditions, lack real value when applied to new, more complex scenarios. Thus, in addition to task performance, it is essential to assess whether the model's judgments are based on meaningful patterns or coincidental, spurious data.

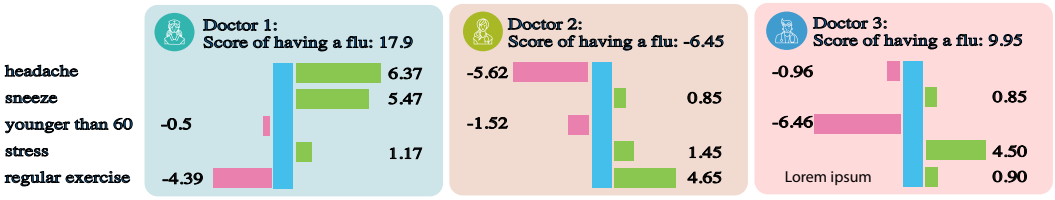


Figure A1. An example of the Clever Hans effect. This figure illustrates the decision-making process employed by three physicians in diagnosing a patient. The patient's symptoms are represented by a feature vector (True, True, False, True, False), corresponding to the presence or absence of a headache, sneezing, being younger than 60 years old, experiencing stress, and engaging in regular exercise, respectively. Doctor 1 exhibits the most rational judgment, ultimately determining that the patient is suffering from the flu. In contrast, Doctor 2 demonstrates a less accurate assessment, arriving at a conclusion that contradicts Doctor 1's diagnosis. Lastly, Doctor 3 reaches the same conclusion as Doctor 1; however, his decision-making process is notably flawed.

AI systems must not only perform tasks but also base their decisions on valid, generalizable relationships. Task performance alone is insufficient; models must demonstrate that their reasoning is grounded in relevant, interpretable patterns rather than temporary associations found in training data.

When decision-making is based on spurious correlations, models fail to exhibit human-like understanding and common sense. Such models lack insight into causal relationships and the invariances that characterize real-world data, limiting their ability to generalize beyond the training environment.

In contrast, models that make valid judgments align with fundamental principles and grasp underlying cause-and-effect relationships. These models can generalize across diverse, novel tasks by focusing on what is truly relevant. This ability to reason flexibly, similar to human expertise, is essential for AI to achieve practical, real-world applicability.

Appendix C. Other Gradient-Based Explanation Methods

Appendix C.1. Gradient Input Multiplication

A commonly used approach in gradient-based explanations is the element-wise multiplication of the saliency map values with the actual input values, represented as

$$s^{GI}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} Q(\boldsymbol{x}; \boldsymbol{\theta}) \odot \boldsymbol{x}. \tag{A1}$$

This method is based on the assumption that the contribution of a feature to the overall output score is proportional to its input value. For example, in a linear system described by $y = \boldsymbol{w}^T \boldsymbol{x}$, the product $w_i x_i$ reflects the contribution of the feature x_i to the final output. This interaction between input features and model parameters provides insights into how different features influence the model's predictions in a probabilistic classification task.

It is important to recognize that saliency maps can sometimes be noisy or less interpretable due to the high sensitivity of gradients to small input feature changes, particularly for complex QNNs. To address this issue, techniques such as integrated gradients and SmoothGrad can be used. These techniques provide smoother and more interpretable visualizations of feature importance by reducing noise or incorporating a baseline for comparison.

Appendix C.2. Integrated Gradients

Integrated gradients offer an interpretable, model-agnostic explanation method that satisfies the axioms of sensitivity and implementation invariance. This technique explains a model's prediction for an input instance $\boldsymbol{x}$ with respect to a selected baseline input $\boldsymbol{x}'$. The core idea is to compute the gradients of the model's output with respect to the input features along a straight-line path from the baseline $\boldsymbol{x}'$ to the input instance $\boldsymbol{x}$, then integrate these gradients over the path.

For each input feature i, the integrated gradient is computed as

$$s_i^{IG}(\boldsymbol{x}) = (x_i - x_i') \int_{\alpha=0}^{1} \frac{\partial}{\partial x_i} Q(\boldsymbol{x}' + \alpha(\boldsymbol{x} - \boldsymbol{x}'), y; \boldsymbol{\theta}) d\alpha, \tag{A2}$$

where α ranges from 0 to 1, x_i and x_i' are the values of the i-th feature in the input instance $\boldsymbol{x}$ and the baseline $\boldsymbol{x}'$, respectively, and $\frac{\partial}{\partial x_i} Q(\boldsymbol{x}' + \alpha(\boldsymbol{x} - \boldsymbol{x}'), y; \boldsymbol{\theta})$ represents the gradient of the model's output with respect to the i-th feature at the intermediate point $\boldsymbol{x}' + \alpha(\boldsymbol{x} - \boldsymbol{x}')$.

In practice, evaluating this integral can be computationally expensive, especially for high-dimensional inputs. To mitigate this, the integral is approximated as a summation over a finite number of steps N:

$$s_i^{IG}(\boldsymbol{x}) \approx \frac{1}{N}(x_i - x_i') \sum_{k=1}^{N} \frac{\partial}{\partial x_i} Q\left(\boldsymbol{x}' + \frac{k}{N}(\boldsymbol{x} - \boldsymbol{x}'), y; \boldsymbol{\theta}\right). \tag{A3}$$

The choice of baseline in the integrated gradients method plays a crucial role in interpreting the results. The baseline is typically chosen to represent a neutral or reference input. For instance, in the case of image data, the baseline might be an all-black image or one with pixel values set to the dataset mean. A well-chosen baseline ensures that the computed feature attributions accurately reflect each feature's contribution to the model's output.

Appendix C.3. SmoothGrad

SmoothGrad [74] seeks to reduce the noise in gradient-based explanations by averaging the gradients of several noisy versions of the input. This technique leads to smoother and more interpretable visualizations of feature importance.

For each input feature i, the SmoothGrad explanation is computed as

$$s_i^{SG}(\boldsymbol{x}) = \int \frac{\partial}{\partial x_i} Q(\boldsymbol{x} + \boldsymbol{\epsilon}, y; \boldsymbol{\theta}) p(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon}, \tag{A4}$$

where $\boldsymbol{x}$ is the input, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$ represents Gaussian noise with a mean of 0 and covariance matrix $\sigma^2 \boldsymbol{I}$ (where $\boldsymbol{I}$ is the identity matrix), and $p(\boldsymbol{\epsilon})$ is the probability density function of the Gaussian noise distribution.

By adding Gaussian noise to the input $\boldsymbol{x}$ and computing gradients for each noisy version, multiple gradient-based explanations are generated. These explanations capture the local variations in the model's behavior. Averaging the gradients reduces noise and yields a smoother, more stable explanation of each feature's importance.

SmoothGrad can be combined with other gradient-based methods, such as saliency maps and integrated gradients, to further enhance the clarity and interpretability of feature importance visualizations in deep learning models.

Appendix D. Comparison of Feature-Based Methods

Occlusion analysis and gradient-based explanation methods are two distinct approaches used to determine the importance of input features for a given prediction. While both aim to provide insight into a model's decision-making process, they differ significantly in methodology, computational complexity, and robustness.

Appendix D.1. Methodology

Occlusion analysis systematically occludes portions of the input using a sliding window (mask) and evaluates the resulting changes in the model's output. This process helps identify the critical regions in the input that contribute most to the model's prediction. In contrast, gradient-based methods compute the gradients of the model's output with respect to the input features. These gradients reflect the sensitivity of the model's output to small variations in the input features, highlighting the importance of each feature in the model's decision.

Appendix D.2. Computational Complexity

Occlusion analysis involves performing multiple forward passes through the model for each occluded version of the input. This can be computationally expensive, especially for large-scale input or deep models. Gradient-based methods, on the other hand, are generally more efficient, as they require only one backward pass to compute the gradients. However, techniques such as integrated gradients and SmoothGrad, which involve multiple iterations or integration steps, increase computational demands. Therefore, a balance must be struck between the computational cost and the desired accuracy of the explanations.

Appendix D.3. Robustness

Occlusion analysis tends to generate stable and consistent explanations because it directly perturbs the input, making it less sensitive to noise in the model's output. On the other hand, gradient-based methods can be more susceptible to noise and may produce noisy or less interpretable explanations. Techniques like SmoothGrad address this issue by averaging gradients over multiple noisy versions of the input, resulting in smoother and more reliable explanations.

In summary, both occlusion analysis and gradient-based methods are valuable tools for interpreting deep learning models. The choice between the two approaches depends on the specific use case, the available computational resources, and the required level of interpretability.

References

1. Biamonte, J.; Wittek, P.; Pancotti, N.; Rebentrost, P.; Wiebe, N.; Lloyd, S. Quantum Machine Learning. *Nature* **2017**, *549*, 195–202. [CrossRef] [PubMed]
2. Rebentrost, P.; Mohseni, M.; Lloyd, S. Quantum Support Vector Machine for Big Data Classification. *Phys. Rev. Lett.* **2014**, *113*, 130503. [CrossRef] [PubMed]
3. Havlíček, V.; Córcoles, A.D.; Temme, K.; Harrow, A.W.; Kandala, A.; Chow, J.M.; Gambetta, J.M. Supervised Learning with Quantum-Enhanced Feature Spaces. *Nature* **2019**, *567*, 209–212. [CrossRef]
4. Abbas, A.; Sutter, D.; Zoufal, C.; Lucchi, A.; Figalli, A.; Woerner, S. The Power of Quantum Neural Networks. *Nat. Comput. Sci.* **2021**, *1*, 403–409. [CrossRef]
5. Schuld, M.; Bocharov, A.; Svore, K.M.; Wiebe, N. Circuit-Centric Quantum Classifiers. *Phys. Rev. A* **2020**, *101*, 032308. [CrossRef]
6. Beer, K.; Bondarenko, D.; Farrelly, T.; Osborne, T.J.; Salzmann, R.; Scheiermann, D.; Wolf, R. Training Deep Quantum Neural Networks. *Nat. Commun.* **2020**, *11*, 808. [CrossRef]
7. Schuld, M.; Sinayskiy, I.; Petruccione, F. The Quest for a Quantum Neural Network. *Quantum Inf. Process.* **2014**, *13*, 2567–2586. [CrossRef]
8. Tian, J.; Sun, X.; Du, Y.; Zhao, S.; Liu, Q.; Zhang, K.; Yi, W.; Huang, W.; Wang, C.; Wu, X.; et al. Recent Advances for Quantum Neural Networks in Generative Learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 12321–12340. [CrossRef]
9. Ciliberto, C.; Herbster, M.; Ialongo, A.D.; Pontil, M.; Rocchetto, A.; Severini, S.; Wossnig, L. Quantum Machine Learning: A Classical Perspective. *Proc. R. Soc. A Math. Phys. Eng. Sci.* **2018**, *474*, 20170551. [CrossRef]
10. Wang, X.; Du, Y.; Luo, Y.; Tao, D. Towards Understanding the Power of Quantum Kernels in the NISQ Era. *Quantum* **2021**, *5*, 531. [CrossRef]
11. Qian, Y.; Wang, X.; Du, Y.; Wu, X.; Tao, D. The Dilemma of Quantum Neural Networks. *arXiv* **2021**, arXiv:2106.04975. [CrossRef] [PubMed]
12. Hastie, T.; Tibshirani, R.; Friedman, J.H.; Friedman, J.H. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*; Springer: Berlin/Heidelberg, Germany, 2009; Volume 2.
13. Doshi-Velez, F.; Kim, B. Towards A Rigorous Science of Interpretable Machine Learning. *arXiv* **2017**, arXiv:1702.08608.
14. Lipton, Z.C. The Mythos of Model Interpretability. *Commun. ACM* **2018**, *61*, 36–43. [CrossRef]

15. Rudin, C. Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Nat. Mach. Intell.* **2019**, *1*, 206–215. [CrossRef]
16. Wachter, S.; Mittelstadt, B.; Russell, C. Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR. *SSRN Electron. J.* **2017**, *31*, *841*. [CrossRef]
17. Adadi, A.; Berrada, M. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access* **2018**, *6*, 52138–52160. [CrossRef]
18. Arrieta, A.B.; Díaz-Rodríguez, N.; Del Ser, J.; Bennetot, A.; Tabik, S.; Barbado, A.; García, S.; Gil-López, S.; Molina, D.; Benjamins, R.; et al. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI. *arXiv* **2019**, arXiv:1910.10045.
19. Molnar, C. *Interpretable Machine Learning*; Lulu.com: Morrisville, NC, USA, 2020.
20. Samek, W.; Montavon, G.; Lapuschkin, S.; Anders, C.J.; Müller, K.R. Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications. *Proc. IEEE* **2021**, *109*, 247–278. [CrossRef]
21. Tim, M. Explanation in Artificial Intelligence: Insights from the Social Sciences | Elsevier Enhanced Reader. *Artif. Intell.* **2019**, *267*, 1–38. [CrossRef]
22. Otgonbaatar, S.; Datcu, M. Classification of Remote Sensing Images with Parameterized Quantum Gates. *IEEE Geosci. Remote. Sens. Lett.* **2021**, *19*, 8020105. [CrossRef]
23. Riedel, M.; Cavallaro, G.; Benediktsson, J.A. Practice and Experience in Using Parallel and Scalable Machine Learning in Remote Sensing from HPC over Cloud to Quantum Computing. In Proceedings of the 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS, Brussels, Belgium, 11–16 July 2021; pp. 1571–1574. [CrossRef]
24. Sebastianelli, A.; Zaidenberg, D.A.; Spiller, D.; Le Saux, B.; Ullo, S.L. On Circuit-Based Hybrid Quantum Neural Networks for Remote Sensing Imagery Classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens.* **2021**, *15*, 565–580. [CrossRef]
25. Zaidenberg, D.A.; Sebastianelli, A.; Spiller, D.; Le Saux, B.; Ullo, S.L. Advantages and Bottlenecks of Quantum Machine Learning for Remote Sensing. In Proceedings of the 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS, Brussels, Belgium, 11–16 July 2021; pp. 5680–5683. [CrossRef]
26. McClean, J.R.; Boixo, S.; Smelyanskiy, V.N.; Babbush, R.; Neven, H. Barren Plateaus in Quantum Neural Network Training Landscapes. *Nat. Commun.* **2018**, *9*, 4812. [CrossRef] [PubMed]
27. Sharma, K.; Cerezo, M.; Cincio, L.; Coles, P.J. Trainability of Dissipative Perceptron-Based Quantum Neural Networks. *Phys. Rev. Lett.* **2022**, *128*, 180505. [CrossRef] [PubMed]
28. Pesah, A.; Cerezo, M.; Wang, S.; Volkoff, T.; Sornborger, A.T.; Coles, P.J. Absence of Barren Plateaus in Quantum Convolutional Neural Networks. *Phys. Rev. X* **2021**, *11*, 041011. [CrossRef]
29. Cerezo, M.; Sone, A.; Volkoff, T.; Cincio, L.; Coles, P.J. Cost Function Dependent Barren Plateaus in Shallow Parametrized Quantum Circuits. *Nat. Commun.* **2021**, *12*, 1791. [CrossRef]
30. Kjaergaard, M.; Schwartz, M.E.; Braumüller, J.; Krantz, P.; Wang, J.I.J.; Gustavsson, S.; Oliver, W.D. Superconducting Qubits: Current State of Play. *Annu. Rev. Condens. Matter Phys.* **2020**, *11*, 369–395. [CrossRef]
31. Cirac, J.I.; Zoller, P. Quantum Computations with Cold Trapped Ions. *Phys. Rev. Lett.* **1995**, *74*, 4091. [CrossRef]
32. Deutsch, D.; Jozsat, R. Rapid Solution of Problems by Quantum Computation. *Proc. R. Soc. Lond. Ser. A Math. Phys. Sci.* **1992**, *439*, 553–558.
33. Shor, P.W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* **1997**, *26*, 1484–1509. [CrossRef]
34. Grover, L.K. A Fast Quantum Mechanical Algorithm for Database Search. In Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing-STOC '96, Philadelphia, PA, USA, 22–24 May 1996; pp. 212–219. [CrossRef]
35. Harrow, A.W.; Hassidim, A.; Lloyd, S. Quantum Algorithm for Linear Systems of Equations. *Phys. Rev. Lett.* **2009**, *103*, 150502. [CrossRef]
36. Lloyd, S.; Mohseni, M.; Rebentrost, P. Quantum Principal Component Analysis. *Nat. Phys.* **2014**, *10*, 631–633. [CrossRef]
37. Hubregtsen, T.; Pichlmeier, J.; Stecher, P.; Bertels, K. Evaluation of Parameterized Quantum Circuits: On the Relation between Classification Accuracy, Expressibility, and Entangling Capability. *Quantum Mach. Intell.* **2021**, *3*, 9. [CrossRef]
38. Sim, S.; Johnson, P.D.; Aspuru-Guzik, A. Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms. *Adv. Quantum Technol.* **2019**, *2*, 1900070. [CrossRef]
39. Pérez-Salinas, A.; Cervera-Lierta, A.; Gil-Fuster, E.; Latorre, J.I. Data Re-Uploading for a Universal Quantum Classifier. *Quantum* **2020**, *4*, 226. [CrossRef]
40. Qpic/Qpic: Creating Quantum Circuit Diagrams in TikZ. Available online: https://github.com/qpic/qpic (accessed on 1 January 2024).
41. Schuld, M.; Bergholm, V.; Gogolin, C.; Izaac, J.; Killoran, N. Evaluating Analytic Gradients on Quantum Hardware. *Phys. Rev. A* **2019**, *99*, 032331. [CrossRef]
42. Mitarai, K.; Negoro, M.; Kitagawa, M.; Fujii, K. Quantum Circuit Learning. *Phys. Rev. A* **2018**, *98*, 32309. [CrossRef]
43. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2017**, arXiv:1412.6980.
44. Preskill, J. Quantum Computing in the NISQ Era and Beyond. *Quantum* **2018**, *2*, 79. [CrossRef]
45. Simonyan, K.; Vedaldi, A.; Zisserman, A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *arXiv* **2014**, arXiv:1312.6034.

46. Hampel, F.R. The Influence Curve and Its Role in Robust Estimation. *J. Am. Stat. Assoc.* **1974**, *69*, 383–393. [CrossRef]
47. Barshan, E.; Brunet, M.E.; Dziugaite, G.K. RelatIF: Identifying Explanatory Training Examples via Relative Influence. *arXiv* **2020**, arXiv:2003.11630.
48. Liu, J.; Yuan, H.; Lu, X.M.; Wang, X. Quantum Fisher Information Matrix and Multiparameter Estimation. *J. Phys. A Math. Theor.* **2020**, *53*, 023001. [CrossRef]
49. Meyer, J.J. Fisher Information in Noisy Intermediate-Scale Quantum Applications. *Quantum* **2021**, *5*, 539. [CrossRef]
50. Samek, W.; Binder, A.; Montavon, G.; Bach, S. Evaluating the Visualization of What a Deep Neural Network Has Learned. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *28*, 2660–2673. [CrossRef] [PubMed]
51. Lu, S.; Duan, L.M.; Deng, D.L. Quantum Adversarial Machine Learning. *Phys. Rev. Res.* **2020**, *2*, 033212. [CrossRef]
52. Carlini, N.; Wagner, D. Towards Evaluating the Robustness of Neural Networks. *arXiv* **2017**, arXiv:1608.04644.
53. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. *arXiv* **2015**, arXiv:1412.6572.
54. Moosavi-Dezfooli, S.M.; Fawzi, A.; Frossard, P. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 2574–2582. [CrossRef]
55. Papernot, N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z.B.; Swami, A. The Limitations of Deep Learning in Adversarial Settings. *arXiv* **2015**, arXiv:1511.07528.
56. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; Fergus, R. Intriguing Properties of Neural Networks. *arXiv* **2014**, arXiv:1312.6199.
57. Koh, P.W.; Liang, P. Understanding Black-box Predictions via Influence Functions. *arXiv* **2020**, arXiv:1703.04730.
58. Tsoulos, I.G.; Stavrou, V.; Mastorakis, N.E.; Tsalikakis, D. GenConstraint: A Programming Tool for Constraint Optimization Problems. *SoftwareX* **2019**, *10*, 100355. [CrossRef]
59. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
60. Zhou, B.; Khosla, A.; Lapedriza, A.; Oliva, A.; Torralba, A. Learning Deep Features for Discriminative Localization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2921–2929.
61. Montavon, G.; Binder, A.; Lapuschkin, S.; Samek, W.; Müller, K.R. Layer-Wise Relevance Propagation: An Overview. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*; Samek, W., Montavon, G., Vedaldi, A., Hansen, L.K., Müller, K.R., Eds.; Springer International Publishing: Cham, Switzerland, 2019; Volume 11700, pp. 193–209. [CrossRef]
62. Shrikumar, A.; Greenside, P.; Kundaje, A. Learning Important Features through Propagating Activation Differences. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 3145–3153.
63. Lundberg, S.; Lee, S.I. A Unified Approach to Interpreting Model Predictions. *arXiv* **2017**, arXiv:1705.07874. [CrossRef]
64. Erhan, D.; Bengio, Y.; Courville, A.; Vincent, P. Visualizing Higher-Layer Features of a Deep Network. *Tech. Rep. Univ. Montr.* **2009**, *1341*, 1.
65. Hammoudeh, Z.; Lowd, D. Training Data Influence Analysis and Estimation: A Survey. *arXiv* **2022**, arXiv:2212.04612. [CrossRef]
66. Cook, R.D. Detection of Influential Observation in Linear Regression. *Technometrics* **2000**, *42*, 65–68. [CrossRef]
67. Ghorbani, A.; Zou, J. Data Shapley: Equitable Valuation of Data for Machine Learning. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 2242–2251.
68. Pruthi, G.; Liu, F.; Kale, S.; Sundararajan, M. Estimating Training Data Influence by Tracing Gradient Descent. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 19920–19930.
69. Heese, R.; Gerlach, T.; Mücke, S.; Müller, S.; Jakobs, M.; Piatkowski, N. Explainable Quantum Machine Learning. *arXiv* **2023**, arXiv:2301.09138.
70. Burge, I.; Barbeau, M.; Garcia-Alfaro, J. A Quantum Algorithm for Shapley Value Estimation. *arXiv* **2023**, arXiv:2301.04727.
71. Schuld, M.; Sweke, R.; Meyer, J.J. Effect of Data Encoding on the Expressive Power of Variational Quantum-Machine-Learning Models. *Phys. Rev. A* **2021**, *103*, 032430. [CrossRef]
72. Steinmüller, P.; Schulz, T.; Graf, F.; Herr, D. eXplainable AI for Quantum Machine Learning. *arXiv* **2022**, arXiv:2211.01441.
73. Lapuschkin, S.; Wäldchen, S.; Binder, A.; Montavon, G.; Samek, W.; Müller, K.R. Unmasking Clever Hans Predictors and Assessing What Machines Really Learn. *Nat. Commun.* **2019**, *10*, 1096. [CrossRef] [PubMed]
74. Smilkov, D.; Thorat, N.; Kim, B.; Viégas, F.; Wattenberg, M. SmoothGrad: Removing Noise by Adding Noise. *arXiv* **2017**, arXiv:1706.03825.

eXplainable AI for Quantum Machine Learning

Patrick Steinmüller,[1,*] Tobias Schulz,[1] Ferdinand Graf,[1,†] and Daniel Herr[2,‡]

[1] *d-fine GmbH*
[2] *d-fine AG*

(Dated: October 2022)

Parametrized Quantum Circuits (PQCs) enable a novel method for machine learning (ML). However, from a computational point of view they present a challenge to existing eXplainable AI (xAI) methods. On the one hand, measurements on quantum circuits introduce probabilistic errors which impact the convergence of these methods. On the other hand, the phase space of a quantum circuit expands exponentially with the number of qubits, complicating efforts to execute xAI methods in polynomial time. In this paper we will discuss the performance of established xAI methods, such as Baseline SHAP and Integrated Gradients. Using the internal mechanics of PQCs we study ways to speed up their computation.

I. INTRODUCTION

Since most ML models are too complex for humans to properly interpret, methods have been devised to study these models and to provide the ability to understand how models arrive at certain predictions. This explanation is valuable for model developers, who need to understand e.g. the limitations from a model on a global perspective to adjust the model architecture or the training data. It is also valuable for model users, who are interested in the factors that drive specific model predictions. Hence, those methods can improve the overall model quality as well as model acceptance. In addition, governmental initiatives like the European Union's 'Artificial Intelligence Act' [1] require that models applied in high-risk areas (e.g. access to education and essential private services) provide a minimum level of transparency for users, which highlights the importance of xAI methods.

In this paper, we will study models based on so-called Parameterized Quantum Circuits (PQCs) [2], which are also known as variational quantum circuits or quantum neural networks [3], and how xAI methods can be adjusted to these type of models.

Definition 1 (Parametrized Quantum Circuit (PQC))**.** A PQC with N Qubits and $n+1$ features consists of $n+1$ single-qubit rotation gates $R_\cdot^{(j)}(\varphi_i)$, as well as two-qubit entanglement gates. Here, $j = 1, \ldots, N$ denotes on which qubit the gate acts on, $\cdot = x, y, z$ is a placeholder for the specific axis of rotation, and $i = 1, \ldots, n+1$ indexes the parameters. For our purposes measurements in this circuit are always performed at the end and are in the computational basis.

Remark 2. In some circuits features are re-uploaded. This means that the feature is encoded by some rotation several times at different parts of the circuit. In the following, we will continue the analysis without re-uploading. However, the re-uploading case is easily realised by just equalizing some features $x_i = x_j = ...$

Theorem 3 (Output and Learning Behavior)**.** *For a quantum circuit as defined in Definition 1 the following holds [4]: The output is given as a truncated Fourier series with $\Omega = \{-1, 0, 1\}^{n+1}$ as frequency spectrum*

$$f(\varphi) = \sum_{\omega \in \Omega} a_\omega \sin\langle \omega, \varphi \rangle + b_\omega \cos\langle \omega, \varphi \rangle. \tag{1}$$

This means that the result of PQCs can be viewed as truncated Fourier Series. In our case the parameters a_ω, b_ω are real numbers and a $\mathbb{Z} + i\mathbb{Z}$ multiple of $(1/\sqrt{2})^l$ for some exponent of l. If the set of parameters is divided up into features φ and controls θ, and the feature parameters are allowed to be present repeatedly [5], then a_ω, b_ω are trigonometric polynomials in θ [4]. The structure of the circuit dictates the amount of control available in training the circuit.

II. LITERATURE REVIEW

With the general availability of early quantum computers in the cloud [6–8] a new field developed in finding useful problems to tackle with the current generation of noisy quantum computers [9, 10]. At the core of this effort are the aforementioned PQCs, as they have some inherent resistance against systematic errors of the devices. QAOA [11] and VQE [12] are the most famous algorithms that employ this parameterized approach. Later on, it has been repurposed for various quantum machine learning models [13, 14]. There are still a lot of open questions regarding the effectiveness of training [15, 16] and their expressivity [17] as machine learning models.

Nevertheless, there is some interesting research into the behaviour of PQCs as machine learning models [4]. We try to build upon this work by introducing explainable AI (xAI) to the field of quantum machine learning. Standard xAI methods might help elucidate the behavior of current PQC-based machine learning models.

* patrick.steinmueller@d-fine.de
† ferdinand.graf@d-fine.de
‡ daniel.herr@d-fine.ch

xAI for quantum machine-learning (QML) also seems to have a large overhead with simulation of quantum circuits. Recent advances were demonstrated by beating Google's quantum advantage experiment [18] using tensor network-based simulations [19]. There are other approaches such as stabilizer simulations [20, 21], which can simulate large numbers of qubits but scale unfavourably in the number of non-Clifford gates. We use the aforementioned relationship between PQCs and Fourier series in this paper [4], but expect that other xAI methods can be devised from other simulation approaches.

III. REVIEW OF MODEL-AGNOSTIC EXPLAINABILITY METHODS

There are many model-agnostic explainability methods currently available. Since our main concern is with PQC based models, we are going to assume that our models are at least continuously differentiable. The methods under consideration in our paper are KernelSHAP [22] and Integrated Gradients[23].

In the following, f denotes the model, x the input value for which an explanation is sought and b a base value.

A. Integrated Gradients (IG)

IG [23] is a fast method relying on evaluating the gradient ∇f at equidistant points. Denote by $\gamma_{i;N} = i/Nx + (1 - i/N)b; i = 0, \dots, N$ an N mesh. For the purposes of computing IG efficiently we need to approximate the partial derivative. To do this, we move $\gamma_{N;i}$ along the e axis by a shift δ_e. Using the trapezoid rule and a simple approximation for the partial derivative, we get:

$$\begin{aligned}
\mathrm{IG}(e) &= \int_0^1 \frac{\partial f}{\partial x_e}(\gamma(s))\frac{\mathrm{d}\gamma_e}{\mathrm{d}s}(s)\mathrm{d}s \\
&\approx \frac{x_e - b_e}{2N}\sum_{i=0}^{N-1}\frac{\partial f}{\partial x_e}(\gamma_{N;i+1}) + \frac{\partial f}{\partial x_e}(\gamma_{N;i}) \\
&= \frac{x_e - b_e}{N}\sum_{i=1}^{N-1}\frac{\partial f}{\partial x_e}(\gamma_{N;i}) \\
&\quad + \frac{x_e - b_e}{2N}\left(\frac{\partial f}{\partial x_e}(x_e) + \frac{\partial f}{\partial x_e}(b_e)\right) \\
&\approx \frac{x_e - b_e}{2N\delta_e}\sum_{i=1}^{N-1} f(\gamma_{N;i} + \delta_e) - f(\gamma_{N;i} - \delta_e)
\end{aligned} \tag{2}$$

In the above derivation the end terms were neglected since for large N they will vanish to zero.

IG is a very fast method for differentiable models. It follows a path in a straight line from start b to end x. This makes IG a good choice for large, high-dimensional models, especially in computer vision.

In equation 2 we present two ways of estimating IG values. One with gradients and in the last line with a specific gradient approximation. In settings where differentiable models have good gradient approximations readily available, the first version can be used. If not, the latter can be used as well.

B. SHAP

We are restricting our discussion of SHAP to Baseline SHAP (BS) which is a sub-variant of KernelSHAP. BS uses a single base value $b \in \mathbb{R}^{n+1}$ and an input value $x \in \mathbb{R}^{n+1}$. To get KernelSHAP from BS we can use BS with several base vectors $b_1, \dots$ and average the results over those.

Let $F = [n+1]$ denote the set of features. Let $e \in F$ the feature for which an explanation is desired. Let $S \subseteq F \setminus \{e\}$ be a set of features. Denote by $\overline{S}$ the opposite set in $F \setminus \{e\}$ such that $F = S \cup \overline{S} \cup \{e\}$. Next we define the intermediate vectors

$$(g_S)_i = \begin{cases} x_i, & i \in S; \\ b_i, & \text{otherwise} \end{cases}. \tag{3}$$

Then the BS value for e is defined as

$$\begin{aligned}
&\mathrm{Sh}_f(e) = \\
&\quad \frac{1}{n+1}\sum_{S \subseteq F\setminus\{e\}} \frac{|S|!(n-|S|)!}{n!}(f(g_{S\cup\{e\}}) - f(g_S)) \\
&\quad = \frac{1}{(n+1)!}\sum_{P \in \mathrm{Perm}(F)} (f(g_{P_e\cup\{e\}}) - f(g_{P_e}))
\end{aligned} \tag{4}$$

In the last line $\mathrm{Perm}(F)$ denotes the set of permutations of F and P a specific permutation of F. P_e denotes the elements of the permutation that occurred before feature e. I.e. Let $P = (1, 3, 2)$ then $P_2 = \{1, 3\}$.

C. Comparison of IG and BS

Both IG and BS are black-box methods. They do not generally pose many requirements for explainability. The strongest assumption is the requirement of models to be differentiable. This might exclude tree models and neural networks using step-functions as activation function. However, this can be alleviated by using a regularization procedure (i.e. folding against an appropriate derivative of a $\mathcal{C}_c^\infty$ function).

Both IG and BS are path-dependent feature-perturbation methods. IG samples points along the line spanned by (b, x) while BS samples its points from the set of vertices of the axes-aligned orthotope spanned by (x, b). Here we can see their structural similarity.

BS and IG are model-centric explainability methods. Let $(X, \mathcal{F}, \mathbb{P})$ denote the probability space from which data is drawn to train the model f. Both IG and BS will ignore the underlying correlations of the data. BS treats features as independent. In case of linear Models this can be overcome by a correlation correction if the data follows a multivariate Gaussian distribution. IG will assume a correlation of features along the direction of $x - b$.

BS - SHAP in general - guarantees that features not contributing to the overall model output receive no attribution. The contribution is measured in terms of their marginal impact $f(g_{S\cup\{e\}}) - f(g_S)$. If that difference is always 0, e.g. in a linear model which applies a weight 0 to that feature, the SHAP value will be zero. IG on the other hand will not guarantee this. Specifically, if $x_e - b_e$ increase alongside another feature $x_{e'} - b_{e'}$, and the gradients in both e and e' are similar, then both features will have similar contributions.

Both IG and BS are linear in f. This means that computing values for a linear combination of models f_i, it is enough to compute the value for each f_i and then perform the linear combination.

Both IG and BS feature the concept of a base value. In computer vision tasks this value is often set to 0. However, other suitable choices are possible and depend on the problem at hand.

D. Stability of IG and BS

If a function f is executed on quantum hardware, the imperfect nature of the device will introduce a significant number of errors. Let f denote the true *mathematical* function that is approximated by a parametrized quantum circuit. The approximation can be denoted by $g \approx f + \epsilon$.

For a set of input values $X = \{x_1, \ldots, x_S\}$, we have a sequence of outputs $g_i = f(x_i) + \epsilon_i$. We assume that the individual errors ϵ are identically distributed with zero-mean $\mu = 0$ and variance $\sigma^2 > 0$. In equation 2 we need $2(N-1)$ function evaluations for computing the IG-value along a single axis. We need $2(N-1)n$ to compute IG-values for all features.

Let S be the set of evaluation points: $S = \{\gamma_{N;i} - \delta_e, \gamma_{N;i} + \delta_e\}$ and $s = |S|$ its size. Using 2 and plugging in our definition for g:

$$\mathrm{IG}_e(g) = \mathrm{IG}_e(f) + \frac{x_e - b_e}{2N\delta_e} \sum_{i=1}^{N-1} \epsilon_{\gamma_{N;i}+\delta_e} - \epsilon_{\gamma_{N;i}-\delta_e}$$

$$\mathrm{IG}_e(g) = \mathrm{IG}_e(f) + \underbrace{\frac{x_e - b_e}{2N\delta_e} \sum_{i\in S} \epsilon_i}_{=:X_N}$$

Using the central limit theorem (CLT), the error X_N converges in distribution to $\mathcal{N}\left(0, \frac{\sigma^2}{N}\right)$.

We employ the same approach in analysing the stability of BS. In equation 4 two definitions of BS were given. The summation over all permutations has duplicate function evaluations over single points, while the definition using the powerset of $F \setminus \{e\}$ has the individual evaluations scaled by a term, weighing extremal points stronger than others.

Let us first consider the (inefficient) implementation via the permutation sum. If the function is evaluated at the same point for every permutation anew, then each error for the sum is independent. Using linearity of the SHAP values we have: $\mathrm{Sh}_g(e) = \mathrm{Sh}_f(e) + \mathrm{Sh}_\epsilon(e)$ and further:

$$\mathrm{Sh}_\epsilon(e) = \frac{1}{(n+1)!} \sum_{P\in\mathrm{Perm}(F)} \epsilon_F - \epsilon'_F.$$

Using the CLT we get $X_n = \mathrm{Sh}_\epsilon(e) \sim \mathcal{N}\left(0, \frac{2\sigma^2}{(n+1)!}\right)$. The inefficient algorithm reduces the impact of any error, while the number of function evaluations explodes significantly. When evaluating the error with the standard formulation we face a different scenario:

$$\mathrm{Sh}_\epsilon(e) = \frac{1}{(n+1)!} \sum_{S\subseteq F\setminus\{e\}} (|S|!(n-|S|)!)(\epsilon_S - \epsilon'_S) \quad (5)$$

$$= \frac{1}{n+1} \sum_{s=0}^{n} \sum_{S\subseteq F\setminus\{e\};|S|=s} \binom{n}{s}^{-1} (\epsilon_S - \epsilon'_S). \quad (6)$$

The inner sum, sums over $\binom{n}{s}$ many sets. For $s = 0$ or $s = n$ the number of summations is 1. This exactly fits our intuition that errors at the extremal points of the spanned box have the strongest impact on the outcome. However, even in this case we can expect some convergence against a normal distribution. As the number of features increases we expect $X_n = \mathrm{Sh}_\epsilon(e) \sim \mathcal{N}\left(0, \frac{\sigma^2}{n}\right)$.

IV. EXTENDING SHAP TO PQCS

A. Mathematical Results

We are going to employ the result from Theorem 3 to find a (faster) extension of BS for PQCs. First we recall some definitions.

Definition 4 (Multivariate Polynomials)**.** Let V be a K vector-space, where K is a field. We define a polynomial p as an element of $\bigoplus_{i\geq 0} S^i(V)$ where $S^i(V)$ denotes the space of symmetric multivariate forms. Elements of $S^i(V)$ are also called monomials of order i. Each monomial of order i can be represented by a totally symmetric tensor Λ_i. Totally symmetric tensors allow for a rank-1 decomposition as follows

$$\Lambda_i = \sum_{j=1}^{p} \lambda_j v_j^{\otimes i}; \alpha_j \in K, v_j \in V. \tag{7}$$

This form is easily stored on a computer. To show the relationship between rank one compositions and Fourier Series note the simple Taylor expansion.

$$\exp(-i\langle\omega,\varphi\rangle) = \sum_{k\geq 0} \frac{(-1)^k i^k \langle\omega,\varphi\rangle^k}{k!} \approx \sum_{k=0}^{K} \frac{(-1)^k i^k \langle\omega,\varphi\rangle^k}{k!} \tag{8}$$

Expanding all terms in a Fourier Series shows that the wave vectors $\omega_i^{\otimes k}$ represent a natural choice for the rank-1 decomposition. Note that Taylor expansions are not the only choice here. Other polynomial approximation algorithms can be used as well. Chebyshev approximations will reduce the required polynomial order by about half.

Proposition 5 (PolynomialSHAP)**.** *Let Λ be a totally symmetric tensor of order r with a rank-one decomposition as in 7. Let $x, b \in \mathbb{R}^m$ be input and baseline, respectively. Then the BS values are given by:*

$$\begin{aligned}\mathrm{Sh}(e) = 2 \sum_{\substack{j+m+k=r \\ 0\leq j,m,k\leq r \\ m\ odd; k\ even}} \binom{r}{j;m;k} \sum_{\substack{\gamma\in\mathbb{N}^n \\ |\gamma|=k}} \frac{1}{l(\gamma)+1}\binom{k}{\gamma} \\ \sum_{i=1}^{p} \langle v_i, M\rangle^j \left\langle v_i, \frac{\Delta_e}{2}\right\rangle^m \\ \lambda_i \prod_{h=1}^{n} (v_{ih}\alpha_h)^{\gamma_h}.\end{aligned} \tag{9}$$

Here the following definitions are made:

1. *$\mathbb{N}$ denotes the natural numbers with 0;*
2. *$M = (x+b)/2$; $\Delta = x - b$; Δ_e is the vector $\Delta_i = 0$ for $i \neq e$ and Δ_e otherwise; $\alpha_h = ((x+b)/2)_h$;*
3. *$l(\gamma)$ denotes the number of odd indices in γ.*

The proof of this statement can be found in the supplement. The formula above extends the concept of Linear SHAP to Multivariate polynomial functions and runs in $\mathcal{O}(m^r)$, where m denotes the number of features and r the degree of the polynomial.

B. Algorithmic Description and Runtime

Algorithm 1 describes the qSHAP routine.

Algorithm 1 Compute SHAP Values for PQCs f

Require: f function describing the PQC, n number of features, N_i number of times feature $i \in \{1,\dots,n\}$ is encoded. S set of samples in data-space, k order of polynomial approximation.

Set of admissible wave-vectors $\Omega = \prod_{i=1}^{n} [-N_i, N_i]_{\mathbb{Z}}$.
for $\omega \in \Omega$ **do**
    Compute Fourier Coefficients a_ω, b_ω with sample points S
    Using taylors theorem, compute polynomial extension up to order k.
end for
Set $\mathrm{Sh}(e) \leftarrow 0$ for all features $1 \leq e \leq n$.
for $\omega \in \Omega$ **do**
    for (k=1) to K: **do**
        Use PolynomialSHAP to compute contributions: $\mathrm{Sh}_{\omega,k}(e)$.
        $\mathrm{Sh}(e) \leftarrow \mathrm{Sh}(e) + \mathrm{Sh}_{\omega,k}(e)$.
    end for
end for

Essentially the algorithm has three steps:

1. Evaluate the PQC on random sample points.
2. Compute the Fourier decomposition. And form a multivariate polynomial approximation in rank-one form.
3. Use the PolynomialSHAP Algorithm to compute the contributions from each term.

C. Open Issues and Further Direction

Assuming access to quantum hardware the limiting issue for this approach is to obtain a reasonable rank-1 approximation. Currently, having access to the full set of Fourier Modes is required.

The algorithm does not scale exponentially in the number of qubits used for the PQC, but rather in the number of features present in the algorithm. At present no efficient algorithm for Fast Fourier transformation is known to the authors that can take advantage of the way PQCs activate their Fourier terms. The Fourier terms seem to be concentrated close to the border of Ω.

Subsequent steps of the algorithm run at most in polynomial time.

V. EXPERIMENTS

Our experiments are conducted with the 2 by 2 *Bars and Stripes* problem. All possible images are shown in

FIG. 1. For our examples we use the bars and stripes data set. All possibilities of two by two pixel images are shown in this figure. The two images on the left are examples of stripes and the two images on the right are examples of bars. Our xAI models are applied to classifiers trained to distinguish bars from stripes.

Figure 1. The classifiers were trained to distinguish between the images of bars and images of stripes. The three classifiers each used a different parameterized quantum circuit shown in figures 5, 6 and 7. We are going to compare the KernelSHAP implementation of BS, IG, and our approximate qSHAP algorithm both with noise and without noise on these trained PQC classifiers.

For all xAI algorithms and backends we use the same model with the same parameters. That means that the training for each model was performed on classical hardware. The simulations in this case had no noise. Then we compared the results of the different xAI methods across a range of different hardware: classical simulation with no noise, classical simulation with shot-noise, classical simulation with shot-noise and a noise model and execution on a real QPU. To reach optimal model performance in each scenario it is generally recommended to perform the training on the same hardware, on which the model is used later on. However, in this case each model would have different model parameters and the results of the xAI methods would be harder to compare. For this reason we deployed the classically trained model across all hardware options.

Other techniques used for noise reduction on NISQ machines are mentioned in [24]. We did not train the models for each of the noise models further. Instead we can use the different performance of the explainers as a check of robustness against noise.

A. Single-qubit classifier

The single qubit classifier uses the same qubit to upload the top two pixels of the 2 by 2 Bars and Stripes images. The circuit for this classifier is given in Figure 5 and allows it to effectively learn an XOR operation of the two inputs. These inputs are the two top pixels of a bars and stripes image. If they are the same, the image shows stripes and if they are different, the image shows bars.

Looking at the xAI models in the noiseless case, we see that the two upper pixel have an effect towards the model prediction. The bottom pixel roughly have a predictive value of zero. This is the expected behavior as only the upper pixels are used for the model.

With the inclusion of the first noise model this effect does not seem to change with KernelSHAP and our qSHAP method. IG, on the other hand, already seems to struggle due to the noise of this model. With increasing noise this discrepancy becomes more and more pronounced. The noise model on the real hardware seems to be the most severe, causing a lot of artifacts for IG and to a lesser extent for KernelSHAP. The qSHAP method seems to be the most robust against this noise. The noise even leads to negative contributions for a pixel. This is most pronounced in the bottom left pixel for IG, yet this pixel is not used in the classifier.

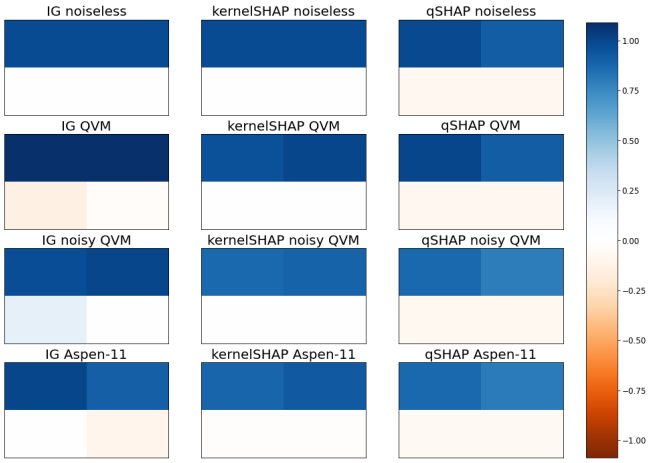


FIG. 2. A comparison of the different xAI methods applied to the simple *single-qubit classifier*. The circuit for the classifier is given in Figure 5. The higher the value for a pixel the more influence it has on the classification. On the horizontal axis the different explainability methods are plotted (Integrated Gradients, KernelSHAP, and qSHAP). Vertically we run the model on different backends. The topmost row uses a state-vector simulator (no noise except for shot noise). The two following rows use different settings for the Rigetti simulator. One with a generic noise model (second row), and one with the noise model of the device (third row). The bottom row shows the results from a run on Rigetti's Aspen-11 quantum computer.

B. Two-qubit classifier

The two-qubit model is slightly more complicated compared to the one-qubit model. It uses a circuit with two qubits, where each qubit is initialized with a rotation dependent on one of the upper two pixels of the bars and stripes image. Again the bottom two pixel are not used in the classifier.

The different xAI models fare similarly for this classifier. This can potentially be explained by the fact that noise does not have as large an influence as for the one-qubit classifier.

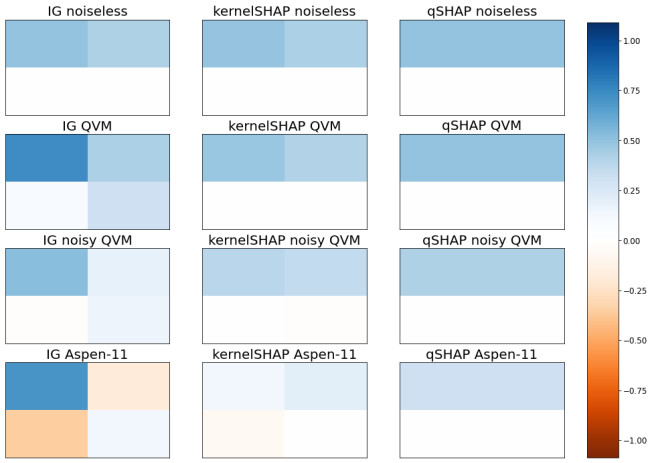


FIG. 3. A comparison of the different xAI methods applied to the *two-qubit classifier*. The circuit for the classifier is given in Figure 6. The two upper pixels are used in the classifier. For more information on the different rows and columns of this Figure refer to the description in 5. Any explainability attributions in the lower two pixels are likely due to noise effects (e.g. shot noise).

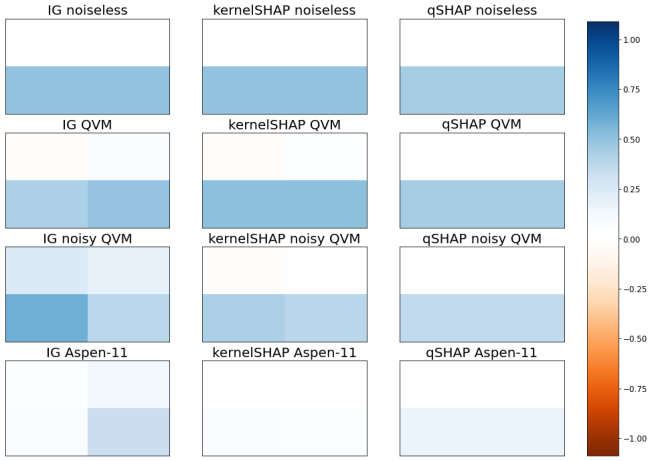


FIG. 4. A comparison of the different xAI methods applied to the *four-qubit classifier*. The circuit for the classifier is given in Figure 7. All pixel are used as input to the classifier and only the measurement result of the last qubit is used for the classification. For more information on the different rows and columns of this Figure refer to the description in 5.

C. Four-qubit classifier

Each pixel of a bars and stripes image is encoded in a different qubit in the four-qubit classifier. After two cycles of parameterized rotations and entanglement operations the fourth qubit is then measured to obtain the classification.

This is the first classifier where all qubits could affect the classification. This means that we cannot interpret a nonzero influence of all pixels as instabilities of the xAI models due to e.g. noise. Nevertheless, in the noiseless case, all xAI models show that only the lower two qubits have an effect on the classification.

D. Classification Performance vs. Explainability

In xAI research one is often faced between achieving a high classification performance or achieving fast and "good" explanations. To a limited degree we see this issue appear here as well. The four-qubit model has the highest number of trainable parameters, meaning that in theory, the space of functions it can encode is much larger. The classification problem presented here is very simple and can be solved in various ways by only taking a subset of the features. In our four-qubit circuit, the qubits that are furthest away from the measurement qubit experience a lesser weighing than the other qubits. This behaviour is similar to classical neural networks where it is harder for the training algorithms to correctly adjust the weights. Furthermore, the quantum nature of PQCs introduces noise, whose impact increases with the size and complexity of the circuit.

E. Runtime for qSHAP

When discussing the runtime of this algorithm we should discuss three parts here:

1. The Sampling Step.

2. The Fourier Step.

3. The Polynomial Step.

During the *Sampling Step* the quantum circuit is sampled S times. The amount of samples required depends on the dimension of the feature space, the circuit noise, etc. This step is linear in the number of samples used. The necessary number of samples should be drawn from convergence criteria from the stochastic integration methods used. However, in practice the size of the circuit is also an issue. There are many steps involved in compiling the logical circuit to the circuit that is evidently used by the hardware. A more complex circuit will introduce a higher overhead, resulting in less evaluations per minute.

Assuming a probabilistic rate of convergence of $\mathcal{O}((n+1)^{-\frac{1}{2}})$ the number of circuit evaluations should scale like $\mathcal{O}(\epsilon^{-1}(n+1)^2)$.

In our runs we used between 250 and 300 circuit evaluations and the below table lists the total time it took to gather the data:

Algorithm	Samples	Time [s]
Single Qubit	250	399.526641
Two Qubit	250	497.247195
Four Qubit	300	682.531422

TABLE I. Table comparing the runtimes recorded for our experimentes of qSHAP on the Rigetti-M-11 hardware.

Starting with the *Fourier Step* we do not use the QPU anylonger. This step is used to estimate which Fourier modes are activated. Importantly, the search space scales exponentially with the number of features and not the number of qubits (as the phase space does). Thus the limiting factor is the number of features in the model. Let N_ω denote the number of fourier modes found.

Lastly, the *Polynomial Step* runs in polynomial time in the number of features. Importantly, this step is also depending on N_ω.

F. Runtime for IG

IG was executed with 20 nodes on the line from base value to the input value. For each point we had to evaluate the circuit eight times to approximate the partial derivatives. The runtimes are shown below.

Algorithm	Samples	Time [sec]
Single Qubit	20	240.778704
Two Qubit	20	261.569914
Four Qubit	20	343.405340

TABLE II. Table comparing the runtimes recorded for our experimentes of qSHAP on the Rigetti-M-11 hardware.

VI. CONCLUSION

In this paper we have applied two black-box explainers to nascent quantum machine learning models. The drawback of these standard black-box xAI models is their exponential scaling and their behaviour under the noise of NISQ devices. As a first step in resolving these issues we propose qSHAP, which is an xAI method specifically designed for PQCs.

Please note that this algorithm does not resolve issues related to the exponential scaling. However, the algorithm does not scale exponentially with respect to the number of qubits, but with respect to the number of features.

For qSHAP, the adverse effect of noise is considerably reduced compared to the other explainers.

While there is a large overlap between xAI for PQCs and the general simulation of circuits, we expect some interesting new approaches to arrive from the recent breakthroughs in simulating circuits used to show quantum advantage [13].

Future work can also be focused on finding efficient algorithms for approximating PQCs with multivariate polynomials in rank-one decomposition. Together with the PolynomialSHAP subroutine, this would result in an efficient algorithm for computing SHAP values for PQCs.

VII. ACKNOWLEDGEMENTS

This work was supported by the German Federal Ministry for Economic Affairs and Climate Action through project PlanQK (01MK20005D).

Special thanks goes to Till Duesberg who provided welcome vetting of many aspects of the initial drafts and Dr. Cristian Grozea who provided constructive criticism throughout. We also want to thank the d-fine internal reviewers Dr. Daniel Ohl de Mello and Dr. Ulf Menzler.

[1] Council of European Union, Proposal - laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts (2021).
[2] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, Parameterized quantum circuits as machine learning models, Quantum Science and Technology **4**, 043001 (2019).
[3] E. Farhi and H. Neven, Classification with quantum neural networks on near term processors, arXiv preprint 10.48550/arXiv.1802.06002 (2018).
[4] M. Schuld, R. Sweke, and J. J. Meyer, Effect of data encoding on the expressive power of variational quantum-machine-learning models, Phys. Rev. A **103**, 032430 (2021).
[5] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, Data re-uploading for a universal quantum classifier, Quantum **4**, 226 (2020).
[6] IBM, Ibm quantum experience (2016).
[7] Amazon, Aws bracket (2020).
[8] Microsoft, Azure quantum (2021).
[9] S. J. Devitt, Performing quantum computing experiments in the cloud, Phys. Rev. A **94**, 032329 (2016).
[10] W. Zeng, B. Johnson, R. Smith, N. Rubin, M. Reagor, C. Ryan, and C. Rigetti, First quantum computers need smart software, Nature **549**, 149 (2017).
[11] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm, arXiv preprint 10.48550/arXiv.1411.4028 (2014).

[12] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, A variational eigenvalue solver on a photonic quantum processor, Nature Communications **5**, 4213 (2014).
[13] H.-Y. Huang, M. Broughton, J. Cotler, S. Chen, J. Li, M. Mohseni, H. Neven, R. Babbush, R. Kueng, J. Preskill, and J. R. McClean, Quantum advantage in learning from experiments, Science **376**, 1182 (2022), https://www.science.org/doi/pdf/10.1126/science.abn7293.
[14] D. Herr, B. Obert, and M. Rosenkranz, Anomaly detection with variational quantum generative adversarial networks, Quantum Science and Technology **6**, 045004 (2021).
[15] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Barren plateaus in quantum neural network training landscapes, Nature Communications **9**, 4812 (2018).
[16] R. Sweke, F. Wilde, J. Meyer, M. Schuld, P. K. Faehrmann, B. Meynard-Piganeau, and J. Eisert, Stochastic gradient descent for hybrid quantum-classical optimization, Quantum **4**, 314 (2020).
[17] A. Abbas, D. Sutter, C. Zoufal, A. Lucchi, A. Figalli, and S. Woerner, The power of quantum neural networks, Nature Computational Science **1**, 403 (2021).
[18] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, Quantum supremacy using a programmable superconducting processor, Nature **574**, 505 (2019).
[19] F. Pan, K. Chen, and P. Zhang, Solving the sampling problem of the sycamore quantum supremacy circuits, arXiv preprint 10.48550/arXiv.2111.03011 (2021).
[20] S. Aaronson and D. Gottesman, Improved simulation of stabilizer circuits, Phys. Rev. A **70**, 052328 (2004).
[21] S. Bravyi, D. Browne, P. Calpin, E. Campbell, D. Gosset, and M. Howard, Simulation of quantum circuits by low-rank stabilizer decompositions, Quantum **3**, 181 (2019).
[22] S. M. Lundberg and S.-I. Lee, A unified approach to interpreting model predictions, in *Advances in Neural Information Processing Systems*, Vol. 30, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Inc., 2017).
[23] M. Sundararajan, A. Taly, and Q. Yan, Axiomatic attribution for deep networks, in *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 70, edited by D. Precup and Y. W. Teh (PMLR, 2017) pp. 3319–3328.
[24] L. Mineh and A. Montanaro, Accelerating the variational quantum eigensolver using parallelism (2022).

Appendix A: Diagrams of the Circuits

This section contains the circuits that we used for our experiments.

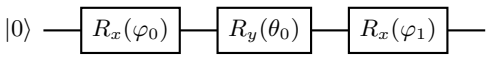


FIG. 5. Single-Qubit circuit that can learn the 2 by 2 Bars and Stripes by using two pixels in a row or in a column. This equates to learning XOR.

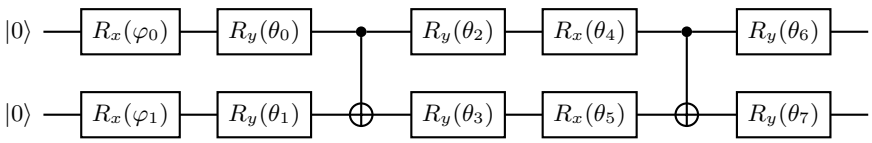


FIG. 6. Two-Qubit circuit that can learn the 2 by 2 Bars and Stripes by using two pixels in a row or in a column. This equates to learning XOR.

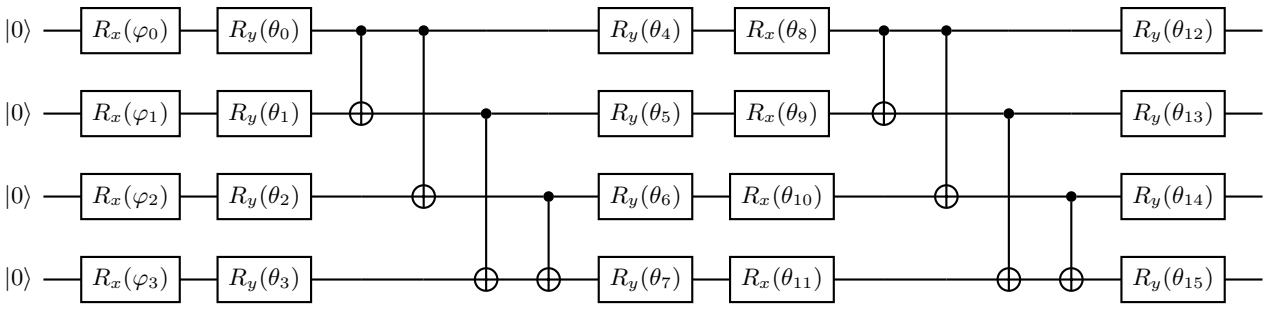


FIG. 7. Four-Qubit circuit that can learn the 2 by 2 Bars and Stripes by using all pixels in the image.

Appendix B: Proof of Proposition 5

In this section we detail the proof for Proposition 5. The following Lemma demonstrates our approach for the quadratic model and motivates our general approach.

Lemma 6. *Consider the BS problem with input x and baseline b for a quadratic form model $f : \mathbb{R}^{n+1} \to \mathbb{R}, x \mapsto x^T C x$ where $C \in \mathbb{R}^{(n+1)\times(n+1)}$ denotes a symmetric matrix. With mean $M = \frac{1}{2}(x+b)$ and difference $\Delta = (x-b)$. The shap values can be computed as:*

$$\mathrm{Sh}(e) = 2M^T C \Delta_e \tag{B1}$$

Proof. Using $b^T C a = a^T C b$ for all $a, b \in \mathbb{R}^{n+1}$:

$$\begin{aligned}\mathrm{Sh}(e) &= \sum_{S\subseteq[n+1]\setminus\{e\}} \frac{|S|!|\overline{S}|!}{(n+1)!}[(x_S + x_e + b_{\overline{S}})^T C(x_S + x_e + b_{\overline{S}}) - (x_S + b_e + b_{\overline{S}})^T C(x_S + b_e + b_{\overline{S}})] \\ &= \sum_{S\subseteq[n+1]\setminus\{e\}} \frac{|S|!|\overline{S}|!}{(n+1)!} 2\left(x_S + b_{\overline{S}} + \frac{1}{2}(x_e + b_e)\right)^T C(x_e - b_e)\end{aligned}$$

Evaluating the left side of the form:

$$\begin{aligned}&2\left(x_S + b_{\overline{S}} + \frac{1}{2}(x_e + b_e)\right) \\ &= 2\left(\frac{x_S + x_e + x_{\overline{S}} + b_S + b_e + b_{\overline{S}}}{2} + \frac{x_S - b_S - (x_{\overline{S}} - b_{\overline{S}})}{2}\right) \\ &= 2\left(M + \frac{1}{2}(\Delta_S - \Delta_{\overline{S}})\right)\end{aligned}$$

Thus:

$$\mathrm{Sh}(e) = \sum_{S\subseteq[n+1]\setminus\{e\}} \frac{|S|!|\overline{S}|!}{(n+1)!}[2M^T C\Delta_e + (\Delta_S - \Delta_{\overline{S}})^T C\Delta_e]$$

Since we are summing over all subsets of $[n+1]\setminus\{e\}$ we have an even number of summands. The transform $S \mapsto \overline{S}$ is bijective and maps the summand:

$$\frac{|S|!|\overline{S}|!}{(n+1)!}(\Delta_S - \Delta_{\overline{S}})^T C\Delta_e \mapsto -\frac{|S|!|\overline{S}|!}{(n+1)!}(\Delta_S - \Delta_{\overline{S}})^T C\Delta_e.$$

Thus the second sum cancels out and we are left with:

$$\mathrm{Sh}(e) = 2M^T C\Delta_e \sum_{S\subseteq[n+1]\setminus\{e\}} \frac{|S|!|\overline{S}|!}{(n+1)!}$$

We evaluate the last sum. Instead of directly summing over all subsets S we sum over the subsets of given size s and then over all sizes. Assume we have subset of S of size s. There are $\binom{n}{s}$ ways to select s elements out of the set $[n+1]\setminus\{e\}$. Thus we can transform the sum:

$$\begin{aligned}\sum_{S\subseteq[n+1]\setminus\{e\}}\frac{|S|!|\overline{S}|!}{(n+1)!} &= \sum_{s=0}^{n}\sum_{\substack{S\subseteq[n+1]\setminus\{e\}\\|S|=s}}\frac{s!(n-s)!}{(n+1)!}\\ &= \sum_{s=0}^{n}\frac{s!(n-s)!}{(n+1)!}\binom{n}{s}\\ &= \sum_{s=0}^{n}\frac{1}{n+1}=1\end{aligned}$$

□

For two vector spaces V,W and bases $a_1,\ldots,a_n$ and $b_1,\ldots,b_m$ we can represent any r-multilinear map Λ by the values of $\Lambda(a_{i_1},\ldots,a_{i_r})$ in the base of W. These values are denoted by $\Lambda_{i_1,\ldots,i_r;j}$ for $1\le j\le m; 1\le i_1,\ldots,i_r\le n$. If W is one-dimensional. This representation is thus naturally in the tensor product written as:

$$\begin{aligned}\Lambda(v_1,\ldots,v_r) &= \sum_{1\le j_1\ldots j_r\le n}\alpha_{1;j_1}\cdots\alpha_{r;j_r}\lambda(a_{i_1}\otimes\cdots\otimes a_{j_r})\\ &= \sum_{j=1}^{m}\sum_{1\le j_1\ldots j_r\le n}\Lambda_{i_1,\ldots,i_r;j}\alpha_{1;j_1}\cdots\alpha_{r;j_r}\beta_j b_j\end{aligned}$$

If furthermore Λ is symmetric then for any permutation π we have $\Lambda_{i_1,\ldots,i_r;j}=\Lambda_{i_{\pi(1)},\ldots,i_{\pi(r)};j}$ and that:

$$\begin{aligned}\Lambda(v_1,\ldots,v_r) &= \sum_{1\le j_1\ldots j_r\le n}\alpha_{1;j_1}\cdots\alpha_{r;j_r}\lambda(a_{i_1}\otimes\cdots\otimes a_{j_r})\\ &= \sum_{j=1}^{m}\sum_{1\le j_1\le\ldots\le j_r\le n}\sum_{\substack{\beta\in\mathbb{N}^r\\|\beta|=r}}\binom{n}{\beta}\Lambda_{i_1,\ldots,i_r;j}\alpha_{1;j_1}\cdots\alpha_{r;j_r}\lambda(a_{i_1}\vee\cdots\vee a_{j_r})\\ &= \sum_{j=1}^{m}\sum_{1\le j_1\le\ldots\le j_r\le n}\sum_{\substack{\beta\in\mathbb{N}^r\\|\beta|=r}}\binom{n}{\beta}\Lambda_{i_1,\ldots,i_r;j}\alpha_{1;j_1}\cdots\alpha_{r;j_r}\beta_j b_j\end{aligned}$$

r-multilinear maps are our generalisation for monomials. Below we introduce some non-standard notations for convenience:

1. Let $x\in\mathbb{R}^n$, then for any $l\ge 0$ we let $x^{\otimes l}$ denote the l-fold self-tensorproduct of x;
2. For $\beta\in\mathbb{N}^n$ and $|\beta|=\beta_1+\ldots+\beta_n=r$ we define $\binom{r}{\beta}=\frac{r!}{\beta_1!\cdots\beta_k!}$;
3. Let $\beta\in\mathbb{N}^n$ be an n-dimensional index vector of rank $r=|\beta|$. For $\alpha\in\mathbb{R}^n$ we have
$$\alpha^{\vee r}=\sum_{|\beta|=r}\underbrace{\binom{r}{\beta}\alpha_1^{\beta_1}\cdots\alpha_n^{\beta_n}}_{=:(\alpha^{\vee r})_\beta}e_1^{\vee\beta_1}\vee\cdots\vee e_n^{\vee\beta_n};$$
4. We introduce a generalization of the scalar product we call $\odot$: $\Lambda\odot M^{\otimes l}=\sum_{i_{r-l+1}=1,\ldots,i_r=1}^{n}\Lambda_{\ldots i_{r-l+1},\ldots,i_r}M_{i_{r-l+1},\ldots,i_s}$ collapses the last dimension of the tensor Λ. Since Λ is symmetric multilinear the exact sequence of collapse is irrelevant.

With the above notation we can rewrite the multinomial expansion as:

$$(x_1 + \ldots + x_n)^k = \sum_{|\beta|=k} (x^{\vee k})_\beta = \sum_{|\beta|=k} \binom{k}{\beta} x_1^{\beta_1} \cdots x_n^{\beta_n}$$
$$= \sum_{1 \le \beta_1 \ldots \beta_n \le k} \underbrace{x_{\beta_1} \cdots x_{\beta_n}}_{=:(x^{\otimes k})_\beta} = \sum_{\beta \in [1,n]^k} (x^{\otimes k})_\beta.$$

Furthermore for symmetric multilinear forms we have:

$$(\Lambda \odot M^{\otimes r}) \odot x^{\otimes s} = \Lambda \odot (M^{\otimes r} \otimes x^{\otimes s}) = \Lambda \odot (x^{\otimes s} \otimes M^{\otimes r})$$

For convenience we drop the paranthesis.

Proposition 7. *Let V be a K vector space and $\Lambda : V^r \mapsto K$ be a symmetric r-multilinear form. Let $a_1, \ldots, a_r \in V$. A rank one decomposition of Λ is given by a natural number p, vectors $v_1, \ldots, v_p \in V$ and coefficients $\lambda_1, \ldots, \lambda_p \in V$ such that:*

$$\Lambda = \sum_{i=1}^{p} \lambda_i v_i^{\otimes r}. \tag{B2}$$

Such a decomposition is not unique. The minimum number p for such a decomposition is called the (generalized) rank of Λ. The evaluation of Λ is given by scalar products as follows:

$$\Lambda(a_1, \ldots, a_r) = \sum_{i=1}^{p} \lambda_i \prod_{j=1}^{r} \langle v_i, a_j \rangle \tag{B3}$$

Proof. With that in mind we can start evaluating BS values for a symmetric r-form:

$$\mathrm{Sh}(e) = \sum_{S \subseteq [n+1] \setminus \{e\}} \frac{|S|!|\overline{S}|!}{(n+1)!} (\Lambda \odot (\underbrace{x_S + b_{\overline{S}} + x_e}_{=:A})^{\otimes r} - \Lambda \odot (\underbrace{x_S + b_{\overline{S}} + b_e}_{=:A'})^{\otimes r})$$

The terms in the parenthesis can be rearranged as follows:

$$A = x_S + b_{\overline{S}} + x_e = \frac{x_S + x_e + x_{\overline{S}} + b_S + b_e + b_{\overline{S}} + x_S - b_S + x_e - b_e + b_{\overline{S}} - x_{\overline{S}}}{2}$$
$$= M + \frac{\Delta_e}{2} + \frac{\Delta_S - \Delta_{\overline{S}}}{2}$$
$$A' = M - \frac{\Delta_e}{2} + \frac{\Delta_S - \Delta_{\overline{S}}}{2}$$

Thus yielding:

$$\mathrm{Sh}(e)$$
$$= \sum_{S \subseteq [n+1] \setminus \{e\}} \frac{|S|!|\overline{S}|!}{(n+1)!} \sum_{l,k} \binom{r}{l;k} (1 - (-1)^l) \Lambda \odot M^{\otimes (r-m-k)} \otimes \left(\frac{\Delta_e}{2}\right)^{\otimes m} \otimes \left(\frac{\Delta_S - \Delta_{\overline{S}}}{2}\right)^{\otimes k}$$
$$= \sum_{m,k} \binom{r}{m;k} (1 - (-1)^l) \Lambda \odot M^{\otimes (r-m-k)} \otimes \left(\frac{\Delta_e}{2}\right)^{\otimes m} \otimes \left(\sum_{S \subseteq [n+1] \setminus \{e\}} \frac{|S|!|\overline{S}|!}{(n+1)!} \left(\frac{\Delta_S - \Delta_{\overline{S}}}{2}\right)^{\otimes k} \right)$$

The above summands are zero if l is even or k is odd.
The preceeding discussion motivates us to evaluate the the term

$$\sum_{S \subseteq [n+1] \setminus \{e\}} \frac{|S|!|\overline{S}|!}{(n+1)!} \left(\frac{\Delta_S - \Delta_{\overline{S}}}{2}\right)^{\otimes k}$$
$$= \sum_{S \subseteq [n+1] \setminus \{e\}} \frac{|S|!|\overline{S}|!}{(n+1)!} \sum_{\beta \in [1,n]^k} \epsilon(S) (\alpha^{\otimes k})_\beta e^{\otimes \beta}.$$

Where $\alpha_i = \frac{1}{2}(\Delta_S - \Delta_{\overline{S}})_i$ if $i \in S$ and $\alpha_i = -\frac{1}{2}(\Delta_S - \Delta_{\overline{S}})_i$ if $i \in \overline{S}$. Let ϵ_i denote a sign that is $+1$ if $i \in S$ and -1 otherwise. Thus

$$\sum_{S\subseteq[n+1]\setminus\{e\}} \frac{|S|!|\overline{S}|!}{(n+1)!} \sum_{\beta\in[1,n]^k} \epsilon(S)(\alpha^{\otimes k})_\beta e^{\otimes\beta}$$
$$= \sum_{\beta\in[1,n]^k} (\alpha^{\otimes k})_\beta e^{\otimes\beta} \sum_{S\subseteq[n+1]\setminus\{e\}} \frac{|S|!|\overline{S}|!}{(n+1)!} \prod_{\beta_i\in\overline{S}} (-1).$$

In the following we denote by $L, 2l = |L|$ the set indices with odd exponents for a given monomial, whose number must be even. Let $o = 2l - \overline{o} = 2l - |\overline{S} \cap L|$:

$$\sum_{\beta\in[1,n]^k} (\alpha^{\otimes k})_\beta e^{\otimes\beta} \sum_{S\subseteq[n+1]\setminus\{e\}} \frac{|S|!|\overline{S}|!}{(n+1)!} \prod_{\beta_i\in\overline{S}} (-1)$$
$$= \sum_{\beta\in[1,n]^k} (\alpha^{\otimes k})_\beta e^{\otimes\beta} \sum_{o=0}^{2l} \sum_{\substack{S\subseteq[n+1]\setminus\{e\} \\ |\overline{S}\cap L|=2l-o}} \frac{|S|!|\overline{S}|!}{(n+1)!} \prod_{\beta_i\in\overline{S}} (-1)$$

The indices of odd exponents distribute over S and $\overline{S}$, where o denotes the number of hits in S and $\overline{o}$ in $\overline{S}$, respectively. The signature $(o, \overline{o}), o + \overline{o} = 2l$ completely determines the sign of the ϵ product. If o is even, then it is positive, otherwise negative. Thus the sum becomes:

$$\prod_{\beta_i\in\overline{S}} (-1) = (-1)^{2l-o}$$

Since only those terms of the monomial matter, that have odd exponents, we get a combinatorial problem as follows:

Given a signature of $(o, \overline{o})$, how many ways are there to arrange S and $\overline{S}$ to hit that signature?

There need to be at least o elements in S, but also $\overline{o} = 2l - o$ in $\overline{S}$. The rest of the elements can be chosen freely.

$$\binom{2l}{o}\binom{n-2l}{s-o}.$$

This can be used to evaluate the inner sum now:

$$\sum_{\substack{S\subseteq[n+1]\setminus\{e\} \\ |S\cap L|=o}} \frac{|S|!|\overline{S}|!}{(n+1)!}(-1)^{2l-o}$$
$$= \sum_{s=o}^{n-(2l-o)} \frac{s!(n-s)!}{(n+1)!}\binom{2l}{o}\binom{n-2l}{s-o}(-1)^{2l-o}$$
$$= \binom{2l}{o}\frac{(n-2l)!}{(n+1)!}(-1)^{2l-o} \sum_{s=o}^{n-(2l-o)} \frac{s!}{(s-o)!}\frac{(n-s)!}{(n-s-(2l-o))!}$$
$$= \binom{2l}{o}\frac{(n-2l)!}{(n+1)!}(-1)^{2l-o} \sum_{s=o}^{n-(2l-o)} s\cdots(s-o+1)\cdot(n-s)\cdots(n-(2l-o)+1-s)$$

We notice the polynomial in the sum would also be zero in the case of $0 \le s \le o-1$ and $n-(2l-o)+1 \le s \le n$ and so we can extend the range of the sum:

$$\binom{2l}{o}\frac{(n-2l)!}{(n+1)!}(-1)^{2l-o} \sum_{s=0}^{n} s\cdots(s-o+1)\cdot(n-s)\cdots(n-(2l-o)+1-s).$$

We claim now that the sum over the polynomial can always be solved and will yield a similar result depending on the parameters $o, 2l, s, k$.

Definition 8 (Derived Sequences)**.** Let $v : \mathbb{Z} \to \mathbb{R}$ be a sequence. Then the following are derived sequences:

- The (first) difference sequence: $\Delta v : \mathbb{Z} \to \mathbb{R}, s \mapsto (\Delta v)(s) := v(s) - v(s-1)$;
- The n-th difference sequence is recursively defined: $(\Delta^{n+1} v)(s) := (\Delta(\Delta^n v))(s)$;

With

$$\begin{aligned}(\Delta u)(s) &= s \cdots (s - o + 1)\\ v(s) &= (n - s) \cdots (n - (2l - o) + 1 - s)\end{aligned}$$

We have:

1. $(\Delta u)(s)$ has roots at $0, \ldots, o - 1$,
2. $v(s)$ has roots at $n - (2l - o) + 1, \ldots, n$,
3. u could have the form $u(s) = \frac{1}{o+1}(s + 1) \cdots (s - o + 1)$ and has roots at $-1, \ldots o - 1$,
4. $\Delta v(s) = -(2l - o) \cdot (n - 1 - s) \cdots (n - (2l - o) + 1 - s)$ and has roots at $n - (2l - o) + 1, \ldots, n - 1$.

With the above convention we can formulate:

$$\begin{aligned}\sum_{s=0}^{n} (\Delta u)(s)v(s) &= \sum_{s=0}^{n} u(s)v(s) - \sum_{s=0}^{n} u(s-1)v(s)\\ &= \sum_{s=0}^{n} u(s)v(s) - \sum_{s=0}^{n} u(s-1)(v(s) - v(s-1)) - \sum_{s=0}^{n} u(s-1)v(s-1)\\ &= \sum_{s=0}^{n} u(s)v(s) - \sum_{s=-1}^{n-1} u(s)v(s) - \sum_{s=0}^{n} u(s-1)\Delta v(s)\\ &= u(n)\underbrace{v(n)}_{=0} - \underbrace{u(-1)}_{=0} v(-1) - \sum_{s=-1}^{n-1} u(s)\Delta v(s)\\ &= -\sum_{s=-1}^{n-1} u(s)\Delta v(s+1).\end{aligned}$$

More generally we can formulate for $1 \le j \le 2l - o$:

$$\begin{aligned}\sum_{s=0}^{n} (\Delta^j u)(s)v(s) &= -\sum_{s=-1}^{n-1} (\Delta^{j-1} u)(s)\Delta v(s)\\ &= (-1)^j \sum_{s=-j}^{n-j} u(s)(\Delta^j v)(s+j).\end{aligned}$$

So to evaluate the sum:

$$\sum_{s=0}^{n} s \cdots (s - o + 1) \cdot (n - s) \cdots (n - (2l - o) + 1 - s)$$

we use the slightly different settings:

$$\begin{aligned}(\Delta^{2l-o} u)(s) &= s \cdots (s - o + 1)\\ v(s) &= (n - s) \cdots (n - (2l - o) + 1 - s).\end{aligned}$$

The i-th integral of $\Delta^{2l-o} u$ has the form:

$$(\Delta^{2l-o+i} u)(s) = \frac{1}{(o+1) \cdots (o+i)}(s + i) \cdots (s - o + 1)$$

having roots at $-i, \dots, o-1$. The $i-th$ derivate of v has the form:

$$(\Delta^i v)(s) = (-1)^i (2l-o)\cdots(2l-o-i+1)(n-i-s)\cdots(n-(2l-o)+1-s)$$

Using this we can show that:

$$\begin{aligned}
&\sum_{s=0}^{n} (\Delta^{2l-o}u)(s)v(s+2l-o) \\
&= (-1)^{2l-o} \sum_{s=-(2l-o)}^{n-(2l-o)} u(s)(\Delta^{2l-o}v)(s) \\
&= (-1)^{2l-o} \sum_{s=-(2l-o)}^{n-(2l-o)} \frac{o!}{(2l)!}(s+(2l-o))\cdots(s-o+1)(-1)^{2l-o}(2l-o)! \\
&= \binom{2l}{o}^{-1} \sum_{s=-(2l-o)}^{n-(2l-o)} (s+(2l-o))\cdots(s-o+1) \\
&= \binom{2l}{o}^{-1} \left[\frac{1}{2l+1}(s+(2l-o)+1)\cdots(s-o+1)\right]_{-(2l-o)-1}^{n-(2l-o)} \\
&= \frac{1}{2l+1}\binom{2l}{o}^{-1}(n+1)\cdots(n-2l+1)
\end{aligned}$$

Which means that

$$\begin{aligned}
&\binom{2l}{o}\frac{(n-2l)!}{(n+1)!}(-1)^{2l-o}\sum_{s=0}^{n} s\cdots(s-o+1)\cdot(n-s)\cdots(n-(2l-o)+1-s) \\
=&\binom{2l}{o}\frac{(n-2l)!}{(n+1)!}(-1)^{2l-o}\frac{1}{2l+1}\binom{2l}{o}^{-1}(n+1)\cdots(n-2l+1) \\
=&\frac{(n-2l)!}{(2l+1)(n+1)!}(n+1)\cdots(n-2l+1)(-1)^{2l-o} \\
=&\frac{(-1)^{2l-o}}{2l+1}
\end{aligned}$$

To evaluate the rest of the sums:

$$\sum_{\beta\in[1,n]^r} (\alpha^{\otimes k})_\beta e^{\otimes\beta} \sum_{o=0}^{2l} \frac{(-1)^{2l-o}}{2l+1} = \sum_{\beta\in[1,n]^r} \frac{1}{2l+1}(\alpha^{\otimes k})_\beta e^{\otimes\beta} \tag{B4}$$

For an efficient evaluation of the remaining sum we have to know how l depends on β.

It is easy to see, that given such a decomposition, storage and evaluation of such multilinear forms is much faster. Combining all of our results leeds to the following expression:

$$\begin{aligned}
&\mathrm{Sh}(e) \\
=&2 \sum_{\substack{j+m+k=r \\ 0\le j,l,k\le r \\ l \text{ odd}; k \text{ even}}} \binom{r}{j;m;k} \sum_{\substack{\gamma\in\mathbb{N}^n \\ |\gamma|=k}} \frac{1}{2l+1}\binom{k}{\gamma} \sum_{i=1}^{p} \langle v_i, M\rangle^j \left\langle v_i, \frac{\Delta_e}{2}\right\rangle^m \lambda_i \prod_{h=1}^{n} (v_{ih}\alpha_h)^{\gamma_h}
\end{aligned}$$

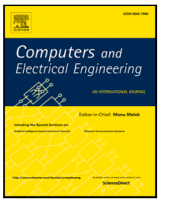
□

Computers and Electrical Engineering 118 (2024) 109302

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng



Explainable hybrid quantum neural networks for analyzing the influence of tweets on stock price prediction

Manoranjan Gandhudi [a], Alphonse P.J.A. [a], Ugo Fiore [b], Gangadharan G.R. [a,*]

[a] National Institute of Technology Tiruchirappalli, India
[b] University of Salerno, Fisciano, Italy

ARTICLE INFO

Keywords:
Explainable artificial intelligence
Quantum neural networks
Stock price prediction
Sentiment analysis
Genetic algorithm
Transformer model

ABSTRACT

Stock price prediction is a complex and challenging activity for organizations and investors to predict future returns. While machine learning and deep learning methods are widely used for stock closing price prediction, these methods have some drawbacks, including high scalability, slow convergence, and poor generalization performance. Furthermore, because those models are inherently black-box, it is challenging to comprehend the logic underlying their forecasts. This study presents an explainable hybrid quantum neural network to investigate the influence of tweets on a stock price prediction. The datasets used in this analysis include the stock prices of six different organizations as well as the 4 million+ tweets written on X (previously Twitter). The proposed methodology finds the average sentiment score of daily tweets using a transformer model which is combined with historical stock data. The proposed hybrid genetic algorithm based quantum neural network predicts the future stock closing price more accurately and uses explainable artificial intelligence methods to investigate the influence of average sentiment score of tweets and compute each attribute contribution towards the outcome. The proposed hybrid quantum neural network outperforms the other existing classical machine learning and quantum inspired machine learning algorithms by achieving a model accuracy (R^2) greater than 99% in the prediction of stock prices for the six different organizations. Further, based on the explainability analysis, we observe that the tweets do not influence stock price to a greater extent.

1. Introduction

Stock market prediction is the process of predicting the future direction of stock prices and overall market trends. Traders, analysts, and investors utilize various methods, instruments, and strategies to understand market patterns, economic indicators, historical data, and other relevant factors [1]. Quantitative, technical, and fundamental approaches are commonly used for stock market prediction [2–5]. However, the complexity arises from factors like geopolitical events, investor sentiment, organizational performance, and unforeseen circumstances, making accurate predictions challenging (Ding et al. 2020).

Stock price prediction with sentiment analysis of tweets analyzes the sentiment of tweets related to a specific stock of a company and uses that sentiment to arrive at predictions about its future stock price movements [6–9]. By understanding the prevailing sentiment, investors could assess the potential for price volatility and market downturns and could achieve insights into how the market might react to specific news or events. While negative tweet sentiments might result in price drops and selling pressure, positive tweet sentiments might enhance buying activity and drive up stock prices. It plays an important role in analyzing market

* Corresponding author.
E-mail address: ganga@nitt.edu (Gangadharan G.R.).

https://doi.org/10.1016/j.compeleceng.2024.109302
Received 19 December 2023; Received in revised form 24 April 2024; Accepted 11 May 2024
Available online 27 May 2024

dynamics and making informed investment decisions with identified trends, by hedging strategies, adjusting portfolio allocations, or implementing risk mitigation measures [10].

The ability of machine learning (ML) and deep learning (DL) approaches to comprehend large volumes of data, recognize intricate patterns, and make predictions based on correlations that have been trained makes them valuable tools for stock prediction [11]. At the core of ML and DL for stock prediction is the concept of feature extraction. Financial data contains various variables such as stock prices, economic and technical indicators, trading volumes, news sentiment etc. These variables are utilized to establish meaningful features that capture relevant stock information. DL techniques, such as artificial neural networks (ANNs), recurrent neural networks (RNNs), and convolutional neural networks (CNNs) are implemented to process time-series and sequential data, allowing them to be suitable for stock market analysis. Large amounts of data may be handled by these models, which can also extract non-linear correlations and reveal complex patterns that conventional statistical techniques might find difficult to identify.

However, predicting stock closing price using ML/DL techniques is computationally prohibitive in case of huge volume of stock price dataset and requires more computation time. Alternatively, quantum inspired machine learning and deep learning techniques are capable of modeling stock price prediction in effective way. As the quantum process works based on qubits, which are represented as superposition states $|0\rangle$ and $|1\rangle$, quantum algorithms perform superior than the classical algorithms in various applications. Quantum approaches generate the distribution by using less number of gate operations, which is not attained by the classical computers [12]. Hence, in this paper, we propose a hybrid genetic algorithm based quantum neural network to predict the variations of stock prices based on their corresponding tweet dataset. Quantum neural networks extract the patterns for making the trading prediction from historical data by modeling as a quantum process. As compared to machine learning and deep learning models, hybrid quantum neural networks with the optimized parameter tuning by genetic algorithm have potential to solve complex problems more accurately and be able to perform the multiple operations simultaneously.

Explainable artificial intelligence (XAI) provides clear explanations for the actions and decisions taken by machine learning/deep learning algorithms as those are black box typed. XAI is a crucial field of study that helps determine the relative relevance of each feature by equitably allocating its contribution. As the predictions arrived by the hybrid quantum neural networks are black box typed, understanding and trusting the predictions are central challenges [13,14]. To overcome this limitation, we use explainable artificial intelligence models [15] to analyze the predictions of hybrid quantum neural networks.

This paper explores the impact of tweets on stock price prediction using an explainable hybrid quantum neural network. The proposed methodology uses a transformer model to determine the daily average sentiment score, which is then fused with historical stock information of an organization and fed into hybrid quantum neural networks. The hybrid quantum neural networks optimized by genetic algorithm predict the future stock closing price more accurately. The predictions arrived from the hybrid quantum neural networks are analyzed by the explainable artificial intelligence models. These XAI models visualize the importance of each feature to arrive at the decision. The proposed explainable hybrid quantum neural network is transparent with respect to fair distribution of each feature importance, adoptable to predict the future stock closing price, and minimize the number of computations to predict the outcomes by performing qubit operations. The salient contributions of the proposed methodology are as follows:

- Fusion of tweets with the historical stock price data of an organization by calculating the average sentiment score for each day using a transformer model.
- A novel explainable hybrid quantum neural network predicting the accurate stock closing price and explaining the significance of each feature influencing the stock closing price.

The identified research questions, research objectives, and research outcomes are presented as follows.

Research Questions

- RQ1: Does the sentiment score of tweets influence the stock price?
- RQ2: Can hybrid quantum neural networks predict the stock closing price accurately?
- RQ3: Are the outcomes generated by the hybrid quantum neural networks explainable?

Research Objectives

- RO1: To compute the average sentiment score of tweets by fusing the stock data using a transformer based sentiment analyzer.
- RO2: To propose a novel hybrid quantum neural network approach to investigate the factors that influence the closing price of a stock.
- RO3: To interpret and analyze the predictions arrived by the hybrid quantum neural networks using explainable artificial intelligence techniques.

Research Outcomes

- ROT1: Hybrid quantum neural networks achieves more accuracy compared to other state of the art quantum inspired ML/DL algorithms (RQ2).
- ROT2: The stock closing price is not influenced by the average sentiment score of tweets to a greater extent based on the results of the proposed method on six different stock data of organizations (RQ1).
- ROT3: Applying the explainable artificial intelligence models on the outcomes of the proposed method clearly demonstrates the non-obvious impact of the average sentiment score in predicting the stock price (RQ3).

The remainder of this paper is organized as follows: Section 2 describes the existing methods for analyzing and predicting stock market prices, particularly with social media sentiment analysis. Section 3 proposes an explainable hybrid quantum neural network for predicting the future stock closing prices accurately. Section 4 discusses the results obtained by the proposed model including explainability analysis, statistical analysis, and ablation studies. Conclusion and future work are discussed in Section 5.

2. Related work

Li and Pan [16] proposed a novel deep learning technique to forecast the stock movement, using a blended ensemble approach to combine two recurrent neural networks with a fully connected neural network. This ensemble blended model captures the patterns of the stock data to perform future predictions. However, the model fails to identify the complex patterns of the stock data. Carta et al. [17] proposed a multi-ensemble, multi-layer stock trading model with hundreds of deep neural networks for preprocessing. The meta model integrates with the deep learning, deep reinforcement learning approaches. Various metalearner trading judgments are combined to create a more robust trading strategy, with multiple trading agents involved in the decision-making process. However, the model is affected by the effects of overfitting. Ni et al. [7] proposed a tweet node model to anticipate the stock price movements based on the historical stock price data and tweets embeddings. The model is designed to interact visually to display the prediction outcomes. However, the constructed neural network seems biased and the fails to analyze the patterns of stock data to analyze the impact of tweets.

Swathi et al. [18] investigated the correlation among the tweets information and historical stock price with the help of learning and teaching based optimization and LSTM approaches. The correlation of tweets are used to identify the impact of sentiment of the users on the stock market. The impact of the sentiment patterns used to investigate the abnormal changes in the stock prices. However, the identified features may be extracted in an effective manner to train the model to enhance the accurate predictions. Li et al. [19] compared a LSTM model performance with the support vector machine and multiple kernel learning models by incorporating news sentiments and stock price to predict the stock price movements. The numerical stock data is modeled as a technical indicator based on technical analysis and the combined data is fed to the deep learning model to perform stock price predictions. However, the model fails to provide the explainability of the predictions arrived by the deep learning model. Chaudhari and Thakkar [9] proposed a quantization-based data fusion approach combining LSTM, back propagation neural network, and deep neural network to predict the stock trend movements. With the fusion of data using quantization, this model achieves the reduction of the amount of data to be transmitted while maintaining as well as improving the stock trend prediction accuracy. However, the model lacks to provide the explainability of the proposed network.

Rezaei et al. [20] proposed multiple hybrid approaches, such as EMD-CNN-LSTM and CEEMD-CNN-LSTM to extricate the time series sequences and the features. These features were fed to the trained model for stock price prediction. The collaboration of multiple models may enhance the analytical power of the model by partitioning the time series data into different frequencies. However, the complexity of the model needs to be optimized for handling the bias in the future predictions. Long et al. [21] described a deep neural network model using the desensitized transaction records and public market information to predict stock price trend by incorporating the knowledge graphs and DL techniques for the financial decision making. The features used for trading are clustered to ensure the robustness of the model and the relevant features are identified based on the knowledge graphs to enhance the model effectiveness. However, the model fails to improve the efficiency of the model in identifying the patterns of the data. Leow et al. [22] proposed a hybrid model by combining BERT model with the genetic algorithm to capture the market conditions through robo advisor to analyze the twitter sentiments for predicting the price movements. The parameters of the model may be updated in an effective manner to analyze the stock patterns in an effective manner. Xu et al. [6] combined a generative adversarial network and cooperative network to propose a self generated adversarial network model to anticipate the stock movements. This model effectively optimizes the overfitting and stochasticity issues simultaneously from the financial text data and stock price historical data. The model lacks to optimize the parameters in an effective manner and the outcomes of the model are not interpreted.

Fiok et al. [23] proposed a transformer based model on the semeval-2017 dataset with the help of NLP techniques to perform sentiment analysis based on twitter tweets and used XAI tools to elaborate the prediction outcomes. However, the accuracy of prediction may be enhanced by using the advanced state of the art models. Xu et al. [24] incorporated the attention model and semantic embedding approach to predict the stock movements by combining local and contextual attention mechanisms. This reduces the noises in the high level fabricated representation to enhance the performance. However, the model does not provide the interpretation for the outcomes. Liu et al. [25] introduced an industry–stock Pearson correlation matrix and uses a matrix factorization approach to extract a vector that fully describes the industry attributes of stocks. This allows the industry attributes to be used to the stock prediction. Additionally, to create a dynamic correlation between stocks and market preference, the past market preference is modeled based on the industry attribute of the companies. This correlation is then merged with the historical price features that the temporal convolutional network retrieved to predict company ranking. However, the model fails to analyze the daily frequencies of stock data and bridge balance between the short and long sides of trading.

Shi et al. [26] proposed a hybrid approach by combining ensemble model and deep learning approach to identify the rich feature extraction using several hidden layers. The weights used in this network are assigned with the help of closed form solution. However, the parameter updation in the model needs to be performed in an effective manner.

Paquet and Soleymani [27] proposed a hybrid approach to perform quantization and measurement of financial trajectories. The approach uses encoders to transform the partitions of the financial data and quantum neural network to predict the density matrix. The model lacks the ability to update the parameters effectively and the outcomes of the model are not interpreted. Liu and Ma [1]

Table 1
Summary of existing models.

Reference	Model(s)	Objectives	Remarks
Ni et al. [7]	Tweet node algorithm	Price prediction based on tweets embeddings and historical data.	The tweets considered from social networks may be biased.
Swathi et al. [18]	Learning and teaching based optimization and long short term memory	Investigate the correlation between tweets and stock price to make predictions.	Feature selection may be improved.
Chaudhari and Thakkar [9]	DNN, LSTM, and BPNN models	Quantization based data fusion method to anticipate the stock price data.	The predictions are block box typed.
Leow et al. [22]	BERT model and genetic algorithm	Robo advisor to analyze the tweets and optimize the portfolio.	The model will not perform the short sellings.
Xu et al. [6]	Self-regulated generative adversarial network	Stock prices prediction based on historical information and tweets.	The predictions are not interpreted.
Fiok et al. [23]	Transformer models	Twitter sentiment analysis with explainable artificial intelligence to analyze the predictions.	Adoption of transformer models can be improved.
Liu et al. [25]	Matrix factorization algorithm	To anticipate the stock price movements with construction of correlation matrix.	The feature correlation is not influenced.
Shi et al. [26]	Deep random vector functional link.	To investigate the performance of the deep learning approach based on RVFL networks.	The parameter can be updated in an effective manner.
Paquet and Soleymani [27]	Quantum neural networks.	To perform quantization and measurement of financial trajectories.	The parameters can be updated in an appropriate way.
Liu and Long [28]	Elman quantum neural network.	To perform the stock market price analysis.	The model may fail to provide the transparency of outcomes to enhance the reliability of predictions
Proposed methodology	Hybrid genetic algorithm based quantum neural network	Influence of tweets on stock price prediction.	Considers the average sentiment scores of the tweets for fusing the data.

Table 2
Symbols and representations.

Symbol	Representation
η	Step size
L	Number of hidden layers
m_l	Number of units in layer l
$\dagger$	Hermitian conjugation
τ_{max}	Strides and number of networks
Δ	Window size
n	Number of qubits
$K_j^{(l)}$	Parameter matrix
Y	Resulting quantum circuit or unitary transformation
p	Number of tweets in a day

proposed an approach combining Elman neural network and quantum physics to predict the stock prices. The model fails to provide the transparency of outcomes to enhance the reliability of predictions.

The shortcomings of the current literature include biased feature selection, non-linear interactions among the features, inappropriate feature correlation, and non-explainability (see Table 1). Further, the researchers have applied the classical machine or deep learning approaches for anticipating the stock prices based on the sentiment of the tweets. These classical machine learning/deep learning models may not have the desired speed, accuracy, and scalability. Further, the predictions are block box typed (non-explainable) in most of these research works. To overcome these limitations, in this paper, we present an explainable hybrid quantum neural network to analyze the stock trends from the data to forecast the accurate stock closing price, by applying the explainable artificial intelligence models for evaluating the importance of each feature on the prediction outcomes.

3. Proposed methodology

In this section, we present a methodology for analyzing the sentiments of the tweets and predicting the stock prices using explainable hybrid quantum neural networks, combining the power of quantum computing, transformer model based sentiment analysis and neural networks in an explainable manner (see Fig. 1). This proposed methodology comprises (i) Transformer model to perform computation of average sentiment score and fusion with stock data, (ii) Designing of hybrid genetic algorithm based quantum neural networks (iii) Explainability analysis over hybrid quantum neural networks. Table 2 lists the symbol table of the important notations used in the paper.

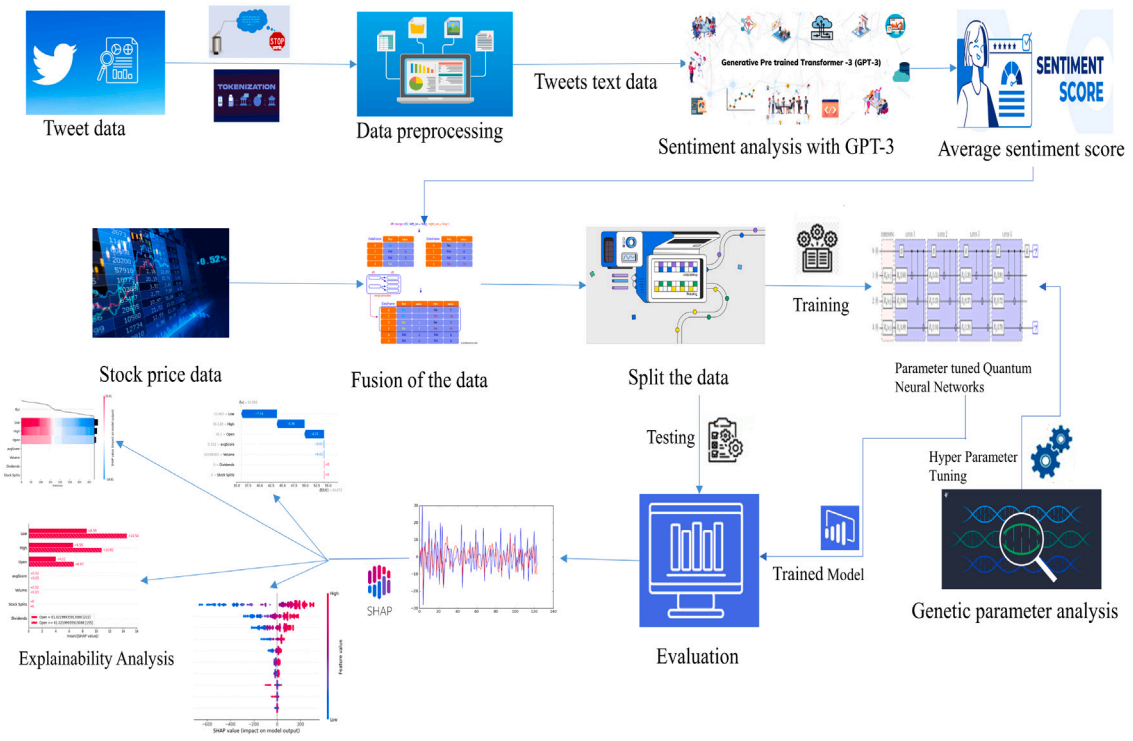


Fig. 1. Framework for explainable hybrid quantum neural network for stock prediction.

3.1. Transformer model for computing sentiment score and fusion with stock data

In the context of tweets and stock data, null values could occur if certain fields are not filled or if data is not available for certain dates or tweets. Removing null values make sure that the data used for analysis is accurate and tweets can be used for further data sentiment analysis. Sentiment analysis aims to discern the emotional polarity for a piece of text or sentiment, such as a tweet. Transformer based GPT-3 is a sentiment analysis model that works based on unlabeled large data to assign sentiment scores to the corresponding sentence [29]. The steps involved in the transformer based sentiment analysis model are listed as follows.
(i) Input text: A tweet is provided in the form of sentence or text or tokens, for which we wish to calculate the sentiment score.
(ii) Design prompt: The input text will be converted to prompt form to instruct the model to calculate sentiment score.
(iii) API Call: The text is given as a input to the GPT-3 model using Open API to analyze the input based on the prompt framework. Then the API will helps to pass the input text as a parameter to the model.
(iv) Response generation: The model analyzes and generates a sentiment score for the text in terms of a numerical score representing the sentiment intensity.

The transformer based sentiment analysis model applies the sentiment analysis to each tweet in the dataframe and generates the sentiment scores for each tweet. Table 3 presents a few sample tweets for Apple Inc. (AAPL) with its corresponding sentiment score for a random day.

From the sentiment score of each tweet, we calculate the average sentiment score of each day. For instance, if there are p number of tweets related to a particular company in a day, then the sentiment scores of each tweet are $[x_1, x_2, \ldots .., x_p]$. The average sentiment score is calculated as the ratio of the sum of the sentiment scores of each tweet to the total number of tweets. Calculation of p number of daily tweets average is presented as follows (see Eq. (1)).

$$Avgscore = \frac{(x_1 + x_2 + \cdots + x_p)}{p} = \frac{\sum_{i=1}^{p} x_i}{p} \tag{1}$$

To analyze the relationship among sentiment scores from the twitter data and the stock data of an organization, we perform fusion of one dataset into another dataset based on the date of tweet/date of stock price as a common feature. The date column is utilized to fuse the data, and the combined data frame that is produced will allow for additional analysis and insights into the potential influence of emotion on stock market activity. The fusion of two different multimodal datasets helps to efficiently analyze the impact of different textual opinions on the numerical data [30]. Thus, data preprocessing followed by fusion ensures the data quality and calculates the sentiment scores using transformers to assess the sentiment of tweets. Fusing the data facilitates the exploration of relationships between sentiment and stock market behavior.

Table 3
Sample tweets with its sentiment score.

S. No	Date	Tweet	Sentiment score
1	01/01/25	My biggest winner in 2014: Inverse volatility ETF $XIVMy biggest loser in 2014: Apple $AAPL	0.10
2	01/01/25	Prediction: $AAPL makes a huge acquisition (Similar to Beats deal; maybe $GPRO?) and begins to tap into its cash hoard	0.31
3	01/01/25	Swing trading: Up to 8.91% return in 14 days http://ow.ly/GDks0 #swingtrading #forecast #techstock $MWW $AAPL $TSLA	0.0
4	01/01/25	Had a down day of −.66%. Worst performer was $AAPL down −1.9% and best was $SBUX up 32%. #Performance #Transparency	0.02
5	01/01/25	$UNP $ORCL $QCOM $MSFT $AAPL Top scoring mega caps right now at the end of 2014 on http://GetAOM.com	0.20
6	01/01/25	RT CNBC: Earlier this month, a mysterious glitch caused $AAPL to suddenly drop 6% ≫ http://cnb.cx/1wafKRS (vi... http://ift.tt/1tpseAE)	−0.272
7	01/01/25	Investment themes: The american consumer is confident again $AAPL	0.49
8	01/01/25	Can't update my iPad because of this... Apple sued for shrinking storage space on 16 GB devices thanks to iOS 8 $AAPL	0.44
9	01/01/25	Prediction: PayPal post-spinoff and $PAYarenolongerindependentcompanies.Theyareboughtby$GOOGL VorMA to compete with $AAPL pay	−0.38
10	01/01/25	The week's winners and losers on Wall Street: Apple gels, Marriott jams http://aol.it/1tkWug8 $AAPL $MAR	−0.07

3.2. Designing of hybrid genetic algorithm based quantum neural networks

Quantum computing leverages quantum bits, known as qubits, which can represent both 0 and 1 simultaneously based on the principle of superposition [1,31]. This property allows quantum computers to perform complex computations in parallel, potentially providing exponential speedup for certain problems compared to classical computers. The data is presented in the form of classical bits. To train the quantum models, we need to convert the classical bits to qubits which transforms the classical information processing to quantum information processing. The classical bits of information are binary digits. Conversely, qubits are quantum counterparts that have the ability to reside in superpositions of states, simultaneously representing 0 and 1, and they can entangle with other qubits. Classical information can be encoded into the quantum state of a qubit. For instance, the classical bit 0 map to the quantum state $|0\rangle$ and 1 to the quantum state $|1\rangle$. Since qubits can exist in a superposition of states, superpositions of $|0\rangle$ and $|1\rangle$ could also be used to encode classical information. A qubit is represented as $|\phi\rangle = \sum_i a_i|i\rangle (i = 1, 2, \dots .n)$, where i represents the computation space for n-dimensional space $\sum_i |a_i|^2 = 1$. If there is even the slightest disturbance happens, $|\phi\rangle$ collapses into possible states of $|i\rangle$ with a probability of $|a_i|^2$. For example, $(|0\rangle + |1\rangle)/\sqrt{(2)}$ represents a superposition with equal probabilities of measuring 0 or 1. The quantum simulation involving the conversion of classical bits to qubits is performed by the qiskit learn and pennylane libraries by providing the circuit size as a parameter value.

The architecture of the quantum neural network contains multiple quantum perceptrons with the optimized parameters [27]. Each perceptron contains b input and c output qubits, which result in $4^{b+c} - 1$ of complex parameters. These parameters are similar to classical weight matrices and the states are denoted by density matrices. The density matrices of input and output are considered as pure states, and hence they can be represented in terms of vector states $|\psi^{\text{in}}\rangle$, $|\psi^{\text{out}}\rangle$ and their Hermitian conjugates $\langle\psi^{\text{in}}|$, $\langle\psi^{\text{out}}|$ as follows (see Eq. (2)).

$$\rho^{\text{in}} = |\psi^{\text{in}}\rangle\langle\psi^{\text{in}}|, \quad \rho^{\text{out}} = |\psi^{\text{out}}\rangle\langle\psi^{\text{out}}| \tag{2}$$

The input layer of a quantum neural network is represented as the density input matrix, L unitary operators or hidden layers and the output layer is denoted by the density output matrix. The deep quantum neural network is presented as follows (see Eq. (3)).

$$\rho^{\text{out}} = \text{tr}_{l\in\{1,\dots,L\}}\left[Y\left(\rho^{\text{in}} \otimes |0_{\text{out}}, 0_{\text{L}}, \dots, 0_1\rangle\langle 0_{\text{in}}, 0_1, \dots, 0_{\text{L}}, 0_{\text{out}}|\right) Y^{\dagger}\right] \tag{3}$$

where † represents Hermitian conjugation. Apart from the output layer, the trace is considered for every layer and $|0, \dots, 0\rangle$ is considered as the ground state and Y is considered as the resulting quantum circuit or unitary transformation (see Eq. (4)).

$$Y = \mathbf{U}^{\text{out}}\mathbf{U}^{\text{L}}\mathbf{U}^{\text{L-1}} \dots \mathbf{U}^{1} \tag{4}$$

where $\{\mathbf{U}^{l}\}_{l=1}^{L}$ are unitary operators, i.e. they satisfy Eq. (5).

$$\mathbf{U}^{l}\mathbf{U}^{l^{\dagger}} = \mathbf{U}^{l^{\dagger}}\mathbf{U}^{l} = \mathbf{I}, \quad \forall l \tag{5}$$

Each unitary operator can be factored as the product of non-commuting unitary operators

$$\mathbf{U}^{(l)} = \prod_{j=1}^{m_l} \mathbf{U}_j^{(l)} \tag{6}$$

Each corresponding to a single unit in the lth layer. The computed network is represented as a constitution of sequence of completely positive transition maps [32].

$$\rho^{\text{out}} = \epsilon^{\text{out}}\left(\epsilon^{L}\left(\ldots \epsilon^{2}\left(\epsilon^{1}\left(\rho^{\text{in}}\right)\right)\ldots\right)\right) \tag{7}$$

The transition map acting on layer $l-1$ is

$$\epsilon^{l}(X^{l-1}) = tr_{l-1}\left[\prod_{j=m_l}^{1}\mathbf{U}_j^{l}(\mathbf{X}^{l-1}\otimes|0_l,\ldots,0_1,0_{\text{in}}\rangle\langle 0_{\text{in}},0_1,\ldots,0_{\mathbf{L}},0_{\text{out}}|)\prod_{j=1}^{m_l}\mathbf{U}_j^{l\dagger}\right] \tag{8}$$

where m_l is the number of units in layer l. A Genetic Algorithm (GA) is a heuristic optimization algorithm inspired by the way of natural selection and genetics, to identify the approximate solutions to optimization and search problems by mimicking the process of evolution. The representation of jth transformed quantum bit based on genetic algorithm is presented as follows Acampora et al. [33] (see Eqs. (9) and (10))

$$K_i^{js} = \frac{1}{2}(b_j(1+\beta_j^i) + a_j(1-\beta_j^i)) \tag{9}$$

$$K_i^{jc} = \frac{1}{2}(b_j(1+\alpha_j^i) + a_j(1-\alpha_j^i)) \tag{10}$$

The rotation of probability amplitudes of qubits of matrix and rotation angle size are represented as follows Singh et al. [34] (see Eqs. (11) and (12))

$$U\begin{bmatrix}sin(gene_{ij})\\cos(gene_{ij})\end{bmatrix} = \begin{bmatrix}sin(\Delta\theta) & -cos(\Delta\theta)\\cos(\Delta\theta) & sin(\Delta\theta)\end{bmatrix}\begin{bmatrix}sin(gene_{ij})\\cos(gene_{ij})\end{bmatrix} = \begin{bmatrix}sin(gene_{ij}+\Delta\theta)\\cos(gene_{ij}+\Delta\theta)\end{bmatrix} \tag{11}$$

$$\Delta\theta_{ij} = -\text{sgn}(A)\,\delta\theta_o \exp\left(-\frac{\Delta f(x_j^i - \Delta f_j min)}{\Delta f_j^{max} - \Delta f_j min}\right) \tag{12}$$

The transformation of jth quantum bit is updated based on Eqs. (9) and (10), the iteration will continue until the iteration number reaches to ceiling value or satisfy the convergence condition. Transition map can be integrated to the activation function and is represented using Eq. (7). As a cost function that quantifies how close the network output ρ^{out} is to the true output $|\psi_t^{\text{out}}\rangle$, the fidelity will be used. It is defined as

$$\mathcal{L} = \frac{1}{t_f - t_i + 1}\sum_{t=t_i}^{t_f}\langle\psi_t^{\text{out}}|\rho_t^{\text{out}}|\psi_t^{\text{out}}\rangle, \quad \mathcal{L}\in[0,1] \tag{13}$$

where t denotes a sample from the training set. The cost function ranges from 0 to 1. As the hybrid quantum neural network reaches the maximum cost function then it may equal to true output. While training, with step size η, the unitary operators are updated as follows (see Eq. (14)).

$$\mathbf{U}_j^l \to \exp\left[i\eta\mathbf{K}_j^l\right]\mathbf{U}_j^l, \quad i\equiv\sqrt{-1} \tag{14}$$

This ensures that the outcome may remain unitary. The variation of the cost function after a training iteration is represented as follows (see Eq. (15)).

$$\Delta C = \frac{\eta}{t_f - t_i + 1}\sum_{t=t_i}^{t_f}\sum_{l=1}^{L+1}\text{tr}[\sigma_t^l\Delta\epsilon^l(\rho_t^{l-1})] \tag{15}$$

where

$$\rho_t^l = \epsilon^l(\ldots\epsilon^2(\epsilon^1(\rho_t^{\text{in}})\ldots)) \tag{16}$$

For the layer l, the density matrix is represented as follows (see Eq. (17))

$$\sigma_t^l = \mathcal{F}^{l+1}\left(\ldots\mathcal{F}^{L}\left(\mathcal{F}^{\text{out}}\left(|\psi_t^{\text{out}}\rangle\langle\psi_t^{\text{out}}|\right)\right)\ldots\right) \tag{17}$$

For a completely positive map ϵ, the adjoint channel is represented as $\mathcal{F}$ (see Eq. (18)).

$$\mathcal{F}^l(X^{l-1}) = \text{tr}_{l-1}\left[\sum_{j=m_l}^{1}\mathbf{U}_j^{l\dagger}\left(|0_{\text{out}},0_L,\ldots,0_1,0_{\text{in}}\rangle\langle 0_{\text{in}},0_1,\ldots,0_l|\otimes X^{l-1}\right)\prod_{j=1}^{m_l}\mathbf{U}_j^l\right] \tag{18}$$

The training phase consists of the following steps: (i) Selection of an initial unitary operator in random way (ii) Evaluation of the density matrix repeatedly (layer by layer) (iii) Computation of the parameter matrix $\mathbf{K}_j^{(l)}$ for the qubits which are unaffected by the step size η and $\mathbf{U}_j^{(l)}$ considering the trace (iv) Updation of the unitary matrix. Steps (ii) and (iii) are repeated till the cost function reaches to maximum [32]. As an outcome, for every layer l, the parameter matrix is computed. Two layers are considered at each time, which considerably optimizes the memory usage. Additionally, to ensure that quantum neural networks are computationally tractable, the matrix dimensions are scaled with the overall number of layers. We considered four layers with the optimized genetic

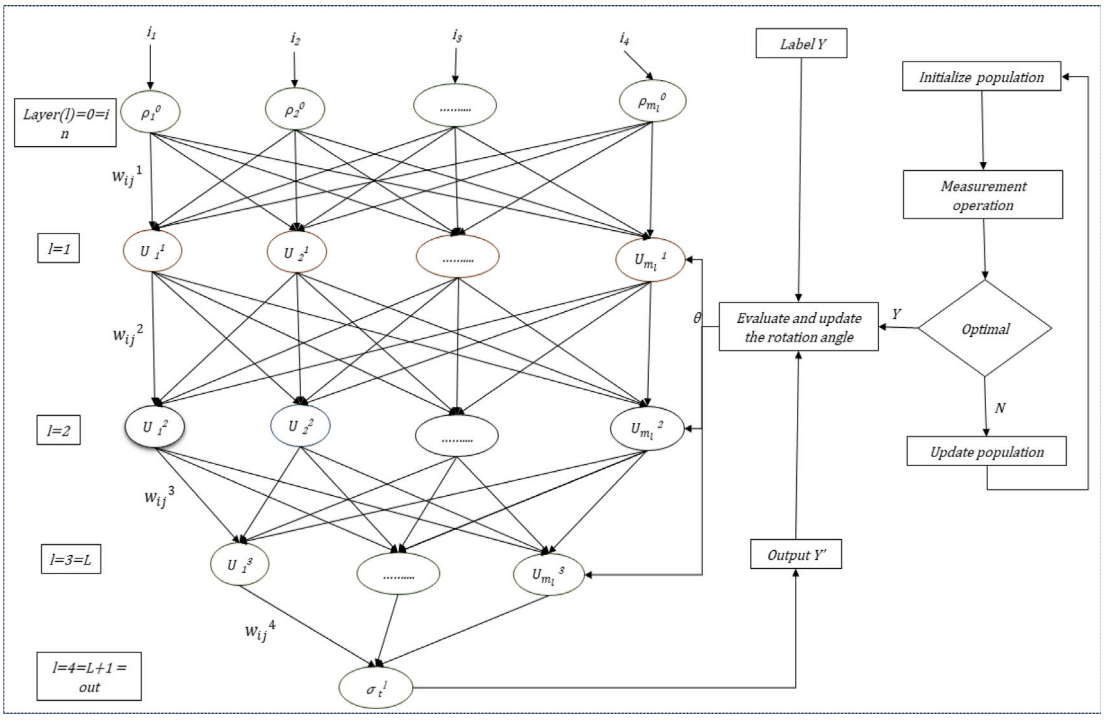


Fig. 2. Architecture of hybrid quantum neural network.

algorithm based hyperparameters to build the novel hybrid quantum neural network. The architecture of the proposed method is represented as follows (see Fig. 2).

$$\mathbf{U}_j^l, \quad j \sim p(j), \quad l \sim p(l) \tag{19}$$

$$\forall \left[\left(|\psi_t^{\text{in}}\rangle, |\psi_t^{\text{out}}\rangle \right) \in \left\{ \left(|\psi_t^{\text{in}}\rangle, |\psi_t^{\text{out}}\rangle \right) \right\}_{t=t_i}^{t_f} \right] \wedge \forall [l] \Rightarrow \rho_t^l = \text{tr}_{l-1} \epsilon^l \left(\rho_t^{l-1} \right) \tag{20}$$

$$\text{tr}_{l-1} \epsilon^l \left(\rho_t^{l-1} \right) = \text{tr}_{l-1} \left[\prod_{j=m_l}^{1} \mathbf{U}_j^l \left(\rho_t^{l-1} \otimes |0_{\text{out}}, 0_L, \ldots, 0_1\rangle\langle 0_{\text{in}}, 0_1, \ldots, 0_L, 0_{\text{out}}| \right) \prod_{j=1}^{m_l} \mathbf{U}_j^{l\dagger} \right] \tag{21}$$

$$\mathbf{U}_j^l \rightarrow \exp \left[\epsilon i \mathbf{K}_j^l \right] \mathbf{U}_j^l, \tag{22}$$

$$\mathbf{K}_j^l = \eta * \frac{2_{l-1}^m}{(t_f - t_i + 1)} * \sum_{t=t_i}^{t_f} \text{tr}_{\not\subset \mathbf{U}_j^l} \circ \mathbf{M}_j^l \tag{23}$$

where

$$\mathbf{M}_j^l = \left[\prod_{\alpha=j}^{1} \mathbf{U}_\alpha^l (\rho_t^{(l-1;l)}) \prod_{\alpha=1}^{j} \mathbf{U}_\alpha^{l\dagger} \cdot \prod_{\alpha=j+1}^{m_l} \mathbf{U}_\alpha^{l\dagger} (I_{l-1} \otimes \sigma_t^l) \prod_{\alpha=m_l}^{j+1} \mathbf{U}_\alpha^l \right] \tag{24}$$

$$\rho_t^{(l-1;l)} \rho_t^{l-1} \otimes |0_{\text{out}}, 0_L, \ldots, 0_1\rangle\langle 0_{\text{in}}, 0_1, \ldots, 0_L, 0_{\text{out}}| \tag{25}$$

$$\boldsymbol{\sigma}_t^l = \mathbf{F}^{l+1} \left(\ldots \mathbf{F}^{\text{out}} (|\psi^{\text{out}}(t)\rangle\langle\psi^{\text{out}}(t)|) \ldots \right) \tag{26}$$

3.3. Explainability analysis over hybrid quantum neural networks

We present the explainability analysis of hybrid quantum neural networks for predicting the stock closing price as follows. The quantum component of the proposed method is often used in the feature mapping stage, where input data is transformed to a high-dimensional quantum space. In the proposed method, the input data, such as historical stock data, is mapped to a quantum state using quantum gates and circuits. This mapping is designed to apprehend the intricate relationships and patterns within the data in a quantum representation as described in Section 3.2. Three quantum gates are considered in our work, so that the possible

Table 4
Number of tweets considered for each stock after data preprocessing.

Stock_Name	Number of tweets after preprocessing
Apple (AAPL)	1 417 557
Amazon (AMZN)	715 527
Google (GOOG)	389 998
Google-L (GOOG-L)	326 384
Microsoft (MSFT)	373 222
Tesla (TSLA)	1 094 031

number of quantum circuits can be computed using $X^3 * Y^3 * Z$, where X is the number of unique single qubit gates, Y is the number of different kinds of two qubit gates, and Z is the number of different kinds of three qubit gates. This expression takes into consideration of the permutations that can be obtained by combining single qubit, two qubit, and three qubit gates within a three qubit quantum system. Each type of single qubit gate can be independently applied to each qubit, resulting in X^3 possible number of configurations. In a similar way, each of these configurations can then be combined with each type of two qubit gate, yielding Y^3 number of possibilities. Furthermore, the addition of three qubit gates introduces a new layer of complexity, with Z number of potential configurations. When combined, these parameters yield $X^3 * Y^3 * Z$ number of unique quantum circuits, illustrating the wide variety of circuit possibilities for design available in a three-qubit quantum system.

Quantum feature mapping techniques, such as quantum embeddings or quantum kernels, are employed to encrypt the data into a quantum state. The quantum representation obtained from the feature mapping stage is then fed into a classical neural network model. This model typically contains multiple layers of classical neurons, such as convolutional or dense layers, which can process the quantum encoded data. The classical network learns from the relationships among the target variable and the quantum-encoded features. The hybrid quantum neural network is trained with labeled data, where the historical stock market price data, tweets sentiments data and the corresponding stock closing price values are considered as training examples. In order to lower the specified loss function, the genetic algorithm is used during training to optimize the parameters of both the classical neural network layers and the quantum feature mapping.

The outcomes of hybrid quantum neural networks are analyzed with Shapley analysis for providing explanations over the predictions of the proposed method and analyzing the interpretability of features. Utilizing Shapley analysis methods such as Shapley Heatmap Analysis, Waterfall Analysis, Cohort Plot Analysis, and Mean Value Analysis enhances the interpretability of outcomes produced by the proposed hybrid quantum neural network. Shapley Heatmap Analysis visually represents the quantum and neural network-driven contributions of individual features to prediction outcomes, offering insights into feature importance. Waterfall Analysis dissects the cumulative impact of features on predictions, unveiling the hierarchical structure of feature importance within the proposed method. Cohort Plot Analysis segments data into cohorts based on feature characteristics, revealing patterns and anomalies in feature interactions across different subsets of observations. Mean Value Analysis computes average Shapley values for each feature, providing an overview of feature importance and aiding in the identification of key drivers within the feature space. Collectively, these Shapley analysis methods facilitate comprehensive explanations of hybrid quantum neural network predictions and insightful analyses of feature interpretability, empowering practitioners to understand and trust the decision-making processes of hybrid quantum neural networks.

4. Results and validation

The proposed methodology is executed on the Google Colab environment using NVIDIA-SMI version 535.104.05, CUDA version 12.2 and hardware accelerator T4 GPU. In our work, we used two different datasets to anticipate the stock closing price. The dataset 1 that we used for our experiments contains more than 4 million unique tweets on six different organizations (Apple, Amazon, Google, Google-L, Microsoft, and Tesla). The dataset 2 that we used in our experiments contains the historical stock prices of the six organizations which are represented in the dataset 1. These datasets are publicly available at https://www.kaggle.com/datasets/omermetinn/tweets-about-the-top-companies-from-2015-to-2020.

The features of the dataset 1 are *tweet id*, *author details of tweet posted*, *post date*, *text of tweet and comments*, *total number of likes for tweet*, and *number of retweets*. The dataset 2 contains historical stock price data from the year 2015 to 2020. The features of the dataset 2 are stock prices of *open*, *high*, *low*, *close*, *volume*, *dividend*, and *stock splits*. In our work, we fused the average daily tweets sentiment score (*avgScore*) from the tweet dataset into the stock price dataset.

As part of data preprocessing, we removed the words which are not useful to find the emotion of the tweet, such as alphanumeric characters, punctuation, numbers, stop words, hashtags, URLs and also removed the words whose length is equal to one. After removing these words, we performed stemming and lemmatization on the words. The number of tweets considered for each stock after preprocessing is given as follows (see Table 4).

We input the data to the transformer sentiment analyzer after preprocessing the data. The analyzer is able to find the sentiment score for each tweet. From the tweets score, we computed the average sentiment score for everyday and fused this average sentiment score data with the stock price data.

In our work, we use a comprehensive simulation setup designed to explore how changes in the key parameters influence the performance of the model. By systematically adjusting these parameters and analyzing their effects, we aim to gain valuable

Table 5
Hybrid quantum neural network parameters.

Parameter	Parameter value
L (number of hidden layers)	4
η (Step size)	0.5
τ_{max}(Strides and number of networks)	20
Number of epochs	50
Δ (Window size)	2
Population size	50
Mutation probability	0.1
Quantum bits (n-dimensional space)	3
Initial value of step length	0.01π

insights into the underlying dynamics of the system and extract optimal operating conditions. Using Google Colab environment, we implemented our hybrid quantum neural network model to perform the future stock price prediction. Table 5 lists the parameters used to implement the hybrid quantum neural network.

This preprocessed data is split into 70% for training and 30% for testing the proposed model. The training data is trained with the proposed hybrid quantum neural networks to analyze the variations and stock movements accurately. The trained hybrid quantum neural network is evaluated on the test dataset. The actual stock closing price and the predicted stock closing price values for the six different organizations are illustrated as follows (see Fig. 3).

The proposed hybrid quantum neural network is evaluated with the performance metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R^2) (see Table 6). These are commonly used metrics in evaluating the performance of regression models, including time series forecasting.

Mean Squared Error (MSE) measures the average squared difference between the actual and predicted values. It penalizes large errors more heavily than small errors. A lower MSE indicates better performance, as it signifies that the model's predictions are closer to the actual values on average. Thus, minimizing MSE is often a primary goal in regression model optimization.

Root Mean Squared Error (RMSE) is the square root of MSE, providing a measure of the average magnitude of errors in the same units as the target variable. Like MSE, lower RMSE values indicate better model performance, with the added benefit of being easily interpretable in the same units as the target variable. RMSE is particularly useful for understanding the typical size of prediction errors.

Mean Absolute Error (MAE) measures the average absolute difference between the actual and predicted values. It provides a more robust measure of error compared to MSE, as it is less sensitive to outliers. Similarly to MSE and RMSE, lower MAE values indicate better model performance. MAE is especially useful when outliers are present or when errors of a consistent size are undesirable.

R-squared (R^2) represents the proportion of variance in the dependent variable that is explained by the independent variables in the model. A higher R^2 value suggests that the model provides a better explanation of the variability in the target variable. R^2 is valuable for assessing the overall goodness-of-fit of the model and comparing different models' performance. The mathematical notations of these metrics are presented as follows (see Table 6).

MSE, RMSE, MAE, and R^2 are chosen as evaluation metrics because they provide comprehensive insights into different aspects of model performance, including accuracy, precision, robustness to outliers, and overall explanatory power. By considering these metrics together, analysts can assess the effectiveness of the proposed solution in accurately forecasting time series data and make informed decisions about model selection and optimization.

In all these metrics, y_i represents the actual value, $\bar{y}_i$ represents the predicted value, and n represents the number of data points. The selection of evaluation metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R^2 (coefficient of determination) for assessing the performance of hybrid quantum neural network offers tailored advantages within this unique framework. MSE, RMSE, and MAE provide a detailed view of prediction errors, focusing on both large errors and average magnitude, essential for a comprehensive analysis of hybrid quantum neural network performance across various datasets. Additionally, RMSE and MAE, being less influenced by outliers, adapt well to the probabilistic nature of quantum operations, accurately reflecting performance in quantum computing environments. R^2 serves as a valuable indicator of how well hybrid quantum neural networks align their predictions with actual data, offering insights into the model's ability to leverage complex quantum relationships and providing a quantum-specific measure of fitness and predictive power. Moreover, RMSE and MAE's interpretability within the quantum domain enhances generalization, making them applicable to a broad spectrum of quantum computing tasks. By using established metrics, hybrid quantum neural network performance assessment aligns with quantum computing standards, ensuring consistency, comparability across studies, and enhancing credibility and reproducibility within the quantum machine learning field. In summary, leveraging these evaluation metrics enables a nuanced assessment of prediction accuracy, robustness, interpretability, and alignment with quantum computing standards, facilitating a comprehensive evaluation of hybrid quantum neural network performance in quantum computing applications.

The proposed method is evaluated and compared with the existing models across the six different organizations (see Table 7). The existing algorithms including Deep Long Short Term Memory [35], Ensemble Deep RVFL [26], Ensemble Deep ESN [36], Quantum Decision Tree [37], Quantum Support Vector Machine [38], Quantum Deep Artificial Neural Networks [1], Quantum Leap [27] are compared with the proposed methodology.

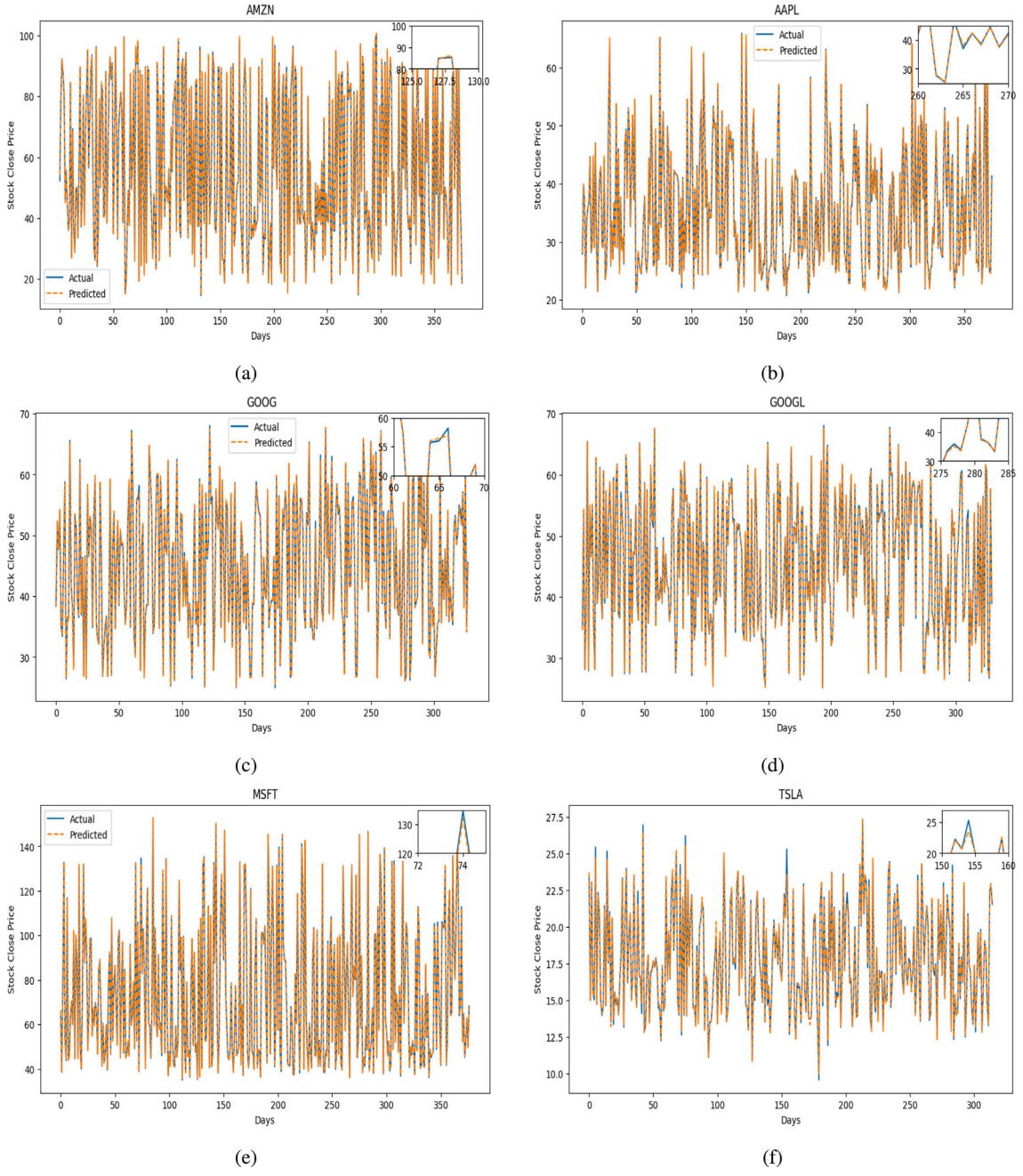


Fig. 3. Comparison of actual vs. predicted values by hybrid quantum neural networks. (a) Amazon. (b) Apple. (c) Google. (d) Google-L. (e) Microsoft. (f) Tesla.

Table 6
Evaluation metrics.

Metric	Equation
MSE	$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
RMSE	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$
MAE	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$
R^2	$1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$

Table 7
Comparison of proposed model outcomes with existing models.

Company	Model	MSE	RMSE	MAE	R^2
Apple (AAPL)	Deep LSTM	8.24	2.87	3.01	91.44
	Ensemble Deep RVFL	7.48	2.73	2.42	91.98
	Ensemble Deep ESN	5.26	2.29	1.77	93.16
	Quantum Decision Tree	2.84	1.68	1.42	95.84
	Quantum SVM	3.71	1.92	1.94	93.71
	Quantum Deep ANN	1.44	1.20	1.52	94.92
	Quantum Leap	0.98	0.99	1.02	96.18
	Proposed model	**0.42**	**0.64**	**0.48**	**99.27**
Amazon (AMZN)	Deep LSTM	3.04	1.74	1.44	91.96
	Ensemble Deep RVFL	2.91	1.70	1.32	94.18
	Ensemble Deep ESN	2.44	1.56	1.20	95.14
	Quantum Decision Tree	1.08	1.03	0.89	96.46
	Quantum SVM	1.87	1.36	1.12	97.29
	Quantum Deep ANN	0.96	1.16	0.92	97.64
	Quantum Leap	0.67	0.81	0.59	98.21
	Proposed model	**0.31**	**0.55**	**0.35**	**99.95**
Google (GOOG)	Deep LSTM	5.48	2.34	1.31	96.14
	Ensemble Deep RVFL	6.01	2.45	1.38	95.97
	Ensemble Deep ESN	5.52	2.35	1.30	96.38
	Quantum Decision Tree	0.87	0.93	0.78	99.18
	Quantum SVM	0.58	0.76	0.69	99.47
	Quantum Deep ANN	0.36	0.60	0.52	99.52
	Quantum Leap	0.34	0.58	0.51	99.49
	Proposed model	**0.09**	**0.30**	**0.23**	**99.93**
Google_L (GOOG_L)	Deep LSTM	15.92	3.98	3.45	92.41
	Ensemble Deep RVFL	12.12	3.48	3.11	92.10
	Ensemble Deep ESN	13.28	3.64	3.25	93.28
	Quantum Decision Tree	5.68	2.38	2.61	94.57
	Quantum SVM	9.47	3.07	3.27	93.48
	Quantum Deep ANN	2.56	1.61	1.21	93.96
	Quantum Leap	1.16	1.07	0.94	97.48
	Proposed model	**0.11**	**0.33**	**0.23**	**99.89**
Microsoft (MSFT)	Deep LSTM	15.01	3.87	3.30	91.98
	Ensemble Deep RVFL	12.78	3.57	3.11	93.65
	Ensemble Deep ESN	9.49	3.08	2.92	94.12
	Quantum Decision Tree	4.27	2.06	1.24	97.59
	Quantum SVM	2.31	1.51	1.19	99.12
	Quantum Deep ANN	1.96	1.42	0.79	99.27
	Quantum Leap	1.80	1.34	0.72	99.29
	Proposed model	**0.32**	**0.56**	**0.39**	**99.96**
Tesla (TSLA)	Deep LSTM	8.12	2.84	1.86	91.42
	Ensemble Deep RVFL	5.48	2.34	1.64	93.91
	Ensemble Deep ESN	6.24	2.49	1.51	94.85
	Quantum Decision Tree	2.27	1.50	1.04	97.21
	Quantum SVM	3.19	1.78	1.59	96.89
	Quantum Deep ANN	2.89	1.72	1.51	97.12
	Quantum Leap	2.12	1.45	1.39	97.93
	Proposed model	**0.38**	**0.61**	**0.41**	**99.15**

The results illustrate that the proposed methodology accurately predicts the future closing values of the stocks across all datasets with an accuracy rate of better than 99 percent (see Table 7). The proposed method outperforms SOTA models due to its ability of effective feature representation, adaptability and flexibility, parallel processing and complex decision boundaries. The proposed approach integrates the advantages of multiple computational paradigms (quantum computing, neural networks, and genetic algorithms) while minimizing the drawbacks of each one separately. The proposed method accurately identifies the complex relationships and patterns in data through the combination of neural networks and quantum computing principles. Quantum computing principles, such as superposition and entanglement, offer computational advantages over classical approaches by enabling parallel processing of information. The proposed methodology leverages the quantum principles to explore a larger solution space more efficiently and to analyze and process data more quickly. This parallelism can significantly speed up inference and training, leading to faster convergence. The neural network component of the proposed approach is able to efficiently intricate the complex relationships in tasks where the decision boundary is highly nonlinear or complex. This capability makes them well-suited for tasks where traditional approaches may fail to capture the underlying relations or patterns accurately. Genetic algorithms are known for their ability to optimize and adapt solutions over multiple iterations. By incorporating genetic algorithms into the training process, the proposed methodology updates their parameters and structure, leading to improved performance. This adaptability gives them an edge over fixed-structure approaches like Quantum Support Vector Machines or Quantum Decision Trees. Overall,

Table 8
Feature importance from Shap mean value analysis.

Company/Attributes	High	Low	Open	Volume	Dividends	Stock splits	Avg score
APPLE (AAPL)	*All/Pos_High*	*All/Pos_High*	*All/Pos_High*	*Neu_High*	*Neu_Low*	*Neu_Low*	*Neu_High*
AMAZON (AMZN)	*All*	*All*	*All*	*Neu_Low*	*Neu_Low*	*Neu_Low*	*Neu_Med*
GOOGLE (GOOG)	*All*	*All*	*All/Neg_Low*	*Neu_High*	*Neu_Low*	*Neu_Low*	*Neu_Med*
GOOGLE-L (GOOG-L)	*All/Pos_High*	*All/Pos_High*	*Pos & Neg*	*Neu_Low*	*Neu_Low*	*Neu_Low*	*Neu_Med*
MICROSOFT (MSFT)	*Pos&Neg/Pos_High*	*Pos&Neg/Pos_High*	*All/Pos_High*	*Neu_Low*	*Neu_Low*	*Neu_Low*	*Neu_Low*
TESLA (TSLA)	*All/Pos_High*	*All*	*Neu/Neg_High*	*Neu_Low*	*Neu_Low*	*Neu_Low*	*Neu_Med*

the integration of quantum computing, neural networks, and genetic algorithms in the proposed methodology gives a versatile and powerful framework for tackling a wide range of machine learning tasks, leading to improved accuracy compared to deep LSTM, deep ensemble methods, or various other quantum machine learning models.

4.1. Explainability analysis for stock prediction

A crucial graphical depiction of Shapley values for model interpretability is the Shapley heat map. We take the Shapley values — a value of prediction among the supplied features — from the Shapley analysis. The value represents the difference between the expected outcome and predicted outcome. Features which are contributing high will have higher Shapley values whereas features which are contributing less will have less Shapley values. The Shapley heat map for the six organizations are shown as follows (see Fig. 4). In Fig. 4(a), we have grouped the instances of similar importance values. The features *Low*, *High* and *Open* could influence the predictions in positive (red colored) or negative (blue colored) or neutral (white colored) way. Based on their intensity levels, the thickness of the feature is presented. For every instance contained in the test data, the *avgScore* is depicted in white, indicating that the feature would not affect the outcome of the prediction. For each of the six groups, we can find the same observation. These plots allow us to draw the conclusion that the sentiment score of tweets has no bearing on the forecast of stock price.

Waterfall plots are the models of explainability analysis, which is used to represent the explanations for the individual outcome. Here, we considered a data sample to explain the influence of features towards the outcomes. The waterfall plot begins by depicting expected value as a result. In this graph, the color red denotes the contribution of positive features to the anticipated outcome, while the color blue denotes the contribution of negative features. The value of each feature for the sample under consideration is shown by the gray text before the feature name. The Waterfall plots based on the proposed hybrid quantum neural networks for the six different organizations are presented as follows (see Fig. 5). For instance, in Fig. 5(a), the average predicted closing price of a stock E[f(x)] is 54.072 and the predicted stock closing price f(x) is 35.998. The features *Low*, *High*, and *Open* influence the average predicted value by −7.74, −6.06, and −4.25 respectively. The feature *avgScore* influence the average predicted value by 0.02 negatively. The remaining features *volume*, *Stock Splits*, and *Dividends* influence the average predicted value by −0.01, +0 and +0 respectively. Thus, we observe that the average predicted value is only little impacted by the *avgScore* among all of these factors for Amazon. We can see a comparable influence on all other organizations regarding the *avgScore*.

The cohort Shapley plots display the Shapley values of explainable analysis for each feature present in the cohort. The plots are represented as a horizontal bar chart or heatmap, where each feature is represented as a cell or bar. The length of the cell represents the impact or magnitude of its Shapley value. Features that have a greater impact on the cohort's outcomes will have longer bars or more intense colors (see Fig. 6). The feature *Low* is having significant contribution for Amazon, Apple and Google-L and the feature *High* is having the significant contribution for Google, Microsoft and Tesla. The plot represents the importance of features in a sequence manner. For Amazon, *Low* is the most significant feature and *Dividends* is the least significant feature. From the plots, we observe that the *avgScore* is having the feature importance ranging from +0.01 to +0.05 for all organizations. Thus, the feature *avgScore* is not contributing to predict the closing price of stock.

Shap mean value analysis interprets to find how the variables influence the predicted outcomes. We use the Shap mean value analysis to visualize the influence of the features on the stock price prediction (see Fig. 7). For the stock price prediction of these six distinct organizations, the feature importance is represented in Table 8 and we observe how each feature is contributed towards the outcomes. In Table 8, the variable *All* represents that the feature is contributed positively, negatively and neutral way. *Pos* represents that the feature is contributed positively and *Neg* represents that the feature is contributed negatively. *Pos_High* represents that the feature is contributed positively with higher coefficients and *Pos_Low* represents that the feature is contributed positively with lower coefficients. Similarly, features with *Neg_High* and *Neg_Low* indicate that these features contributed negatively with high and low coefficients respectively. *Neu_High* represents that the feature contributed with high coefficients with no impact and *Neu_Low* represents that the feature contributed with low coefficients with no impact and *Neu_med* represents that the feature contributed with both positive and negative coefficients with no impact. From the Shap mean value analysis, we observe that the features *Low*, *High* and *Open* are contributing to a greater extent in the prediction of the closing price of a stock more accurately. The remaining features *avgScore*, *Dividends*, *Stock Splits* and *Volume* are not contributed towards the outcomes. Thus, we observe that the tweets do not influence stock price to a greater extent in this case study.

In our research, we applied different models within the realm of explainable artificial intelligence (XAI) to determine the role of individual attributes to our outcomes. In particular, we employed approaches like Shapely heat map analysis (see Fig. 4), waterfall modeling (see Fig. 5), cohort analysis (see Fig. 6), and Shapley mean value analysis (see Fig. 7). Our major focus lays

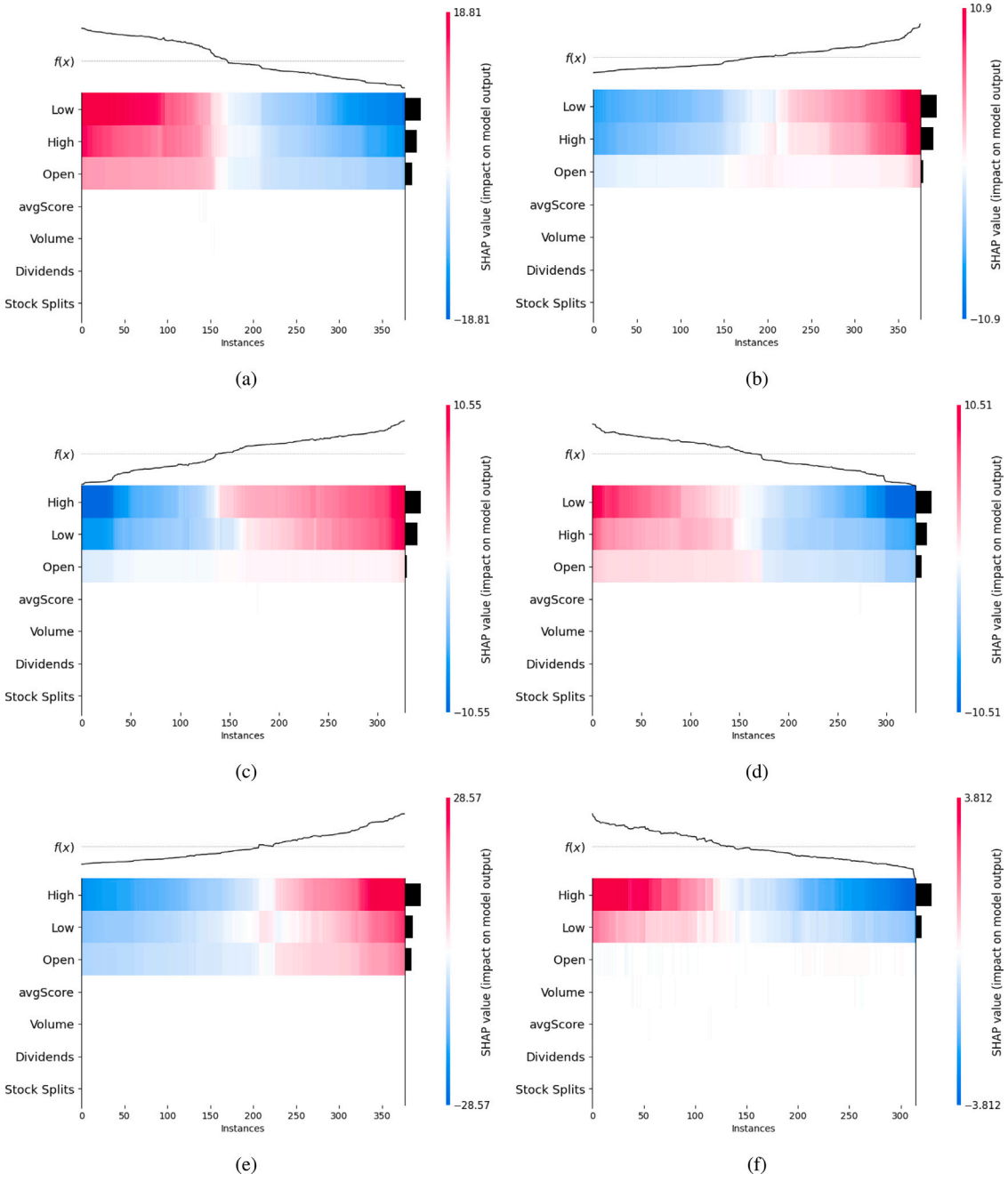


Fig. 4. Shapely heat matrix for six different organization (a) Amazon. (b) Apple. (c) Google. (d) Google-L. (e) Microsoft. (f) Tesla.

in understanding the influence of average sentiment scores on these stock price prediction outcomes. Remarkably, in contrast to our initial expectations, XAI methods illustrate that the sentiment scores taken from the tweets had no obvious impact on the predictions of stock price. This observation underscores the nuanced nature of attribute contributions within our model, suggesting that factors beyond sentiment play a more significant role in determining the outcomes of the stock price prediction.

4.2. Statistical analysis

Statistical analysis provides valuable insights into the relationships between variables, aiding in understanding complex phenomena such as stock market dynamics. A widely used measure for quantifying the strength and direction of the linear relationship

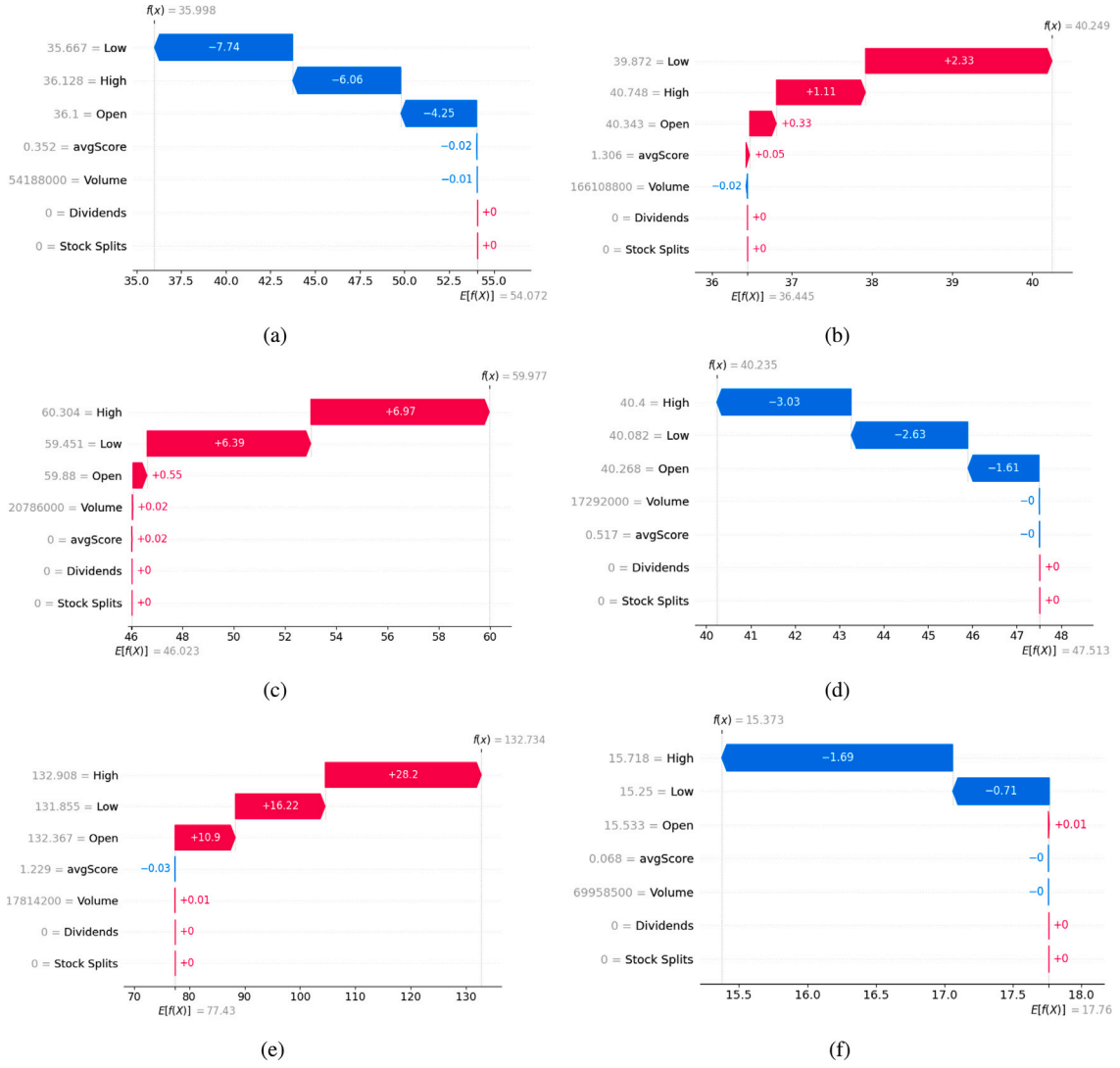


Fig. 5. Shapley waterfall plots of hybrid quantum neural networks predictions (a) Amazon. (b) Apple. (c) Google. (d) Google-L. (e) Microsoft. (f) Tesla.

between two variables is the Pearson correlation coefficient (PCC). The PCC ranges from −1 to 1, where −1 represents a perfect negative linear relationship, 1 represents a perfect positive linear relationship, and 0 represents no linear relationship. In our study, we examined the potential connection between average tweet sentiment scores and stock prices, yielding a PCC coefficient of 0.003. This result suggests a negligible linear relationship between the two variables. Additionally, the associated p-value, a measure of the probability of obtaining the observed correlation coefficient if the true correlation in the population is zero, was found to be 0.12. With a p-value above the conventional significance level of 0.05, the analysis indicates that the observed correlation is not statistically significant. Consequently, based on this analysis, it appears that there is no meaningful association between average tweet sentiment scores and stock prices.

Spearman's Rank Correlation (SRC), also known as Spearman's rho, is considered as a non-parametric measure for the linear association between the variables. We estimate the SRC between the trade volume and the volume of tweets (see Table 9). The SRC and its associated p-value is computed for all organizations. From these computed values, we observe that the p-value for Microsoft exceeds the predetermined threshold value of 0.05, indicating that the considered features (trade volume and tweet volume) are not statistically significant. The remaining all dataset attributes are statistically significant.

Further, we conducted Friedman and Nemenyi tests [39] to assess the performance of the models. The Friedman test checks if there are statistically significant differences among multiple related groups and the Nemenyi test is a pairwise comparison test to determine which groups differ significantly from each other.

To perform the Friedman test we need to (i) Compute the average rank for each model across all datasets. (ii) Calculate the Friedman statistic. (iii) Determine if the Friedman statistic is significant. The average rank of each model across all datasets are listed in Table 10.

Fig. 6. Shap cohort plots of hybrid quantum neural networks predictions. (a) Amazon. (b) Apple. (c) Google. (d) Google-L. (e) Microsoft. (f) Tesla.

The Friedman statistic follows a chi-squared distribution with k-1 degrees of freedom. For $k = 8$ models and $\alpha = 0.05$, the critical value of the Friedman statistic is approximately 14.06 (from the chi-squared distribution table). Since our computed Friedman statistic (23.451) is greater than the critical value, we can reject the null hypothesis and conclude that there are significant differences among the models. Further, we proceed with the Nemenyi test to determine which models differ significantly from each other.

To conduct the Nemenyi test, we compute the critical difference (CD) value, which represents the minimum mean rank difference required for two models to be considered significantly different. Then, we compare the mean ranks of each pair of models to the CD value. For 95% confidence level and $k = 8$, the critical value from the Studentized range distribution table is approximately 2.56 and the computed CD value is 2.40. Hence, we compare the mean ranks of each pair of models to the CD value. The comparison of mean rank difference and CD value significance is presented in Table 11.

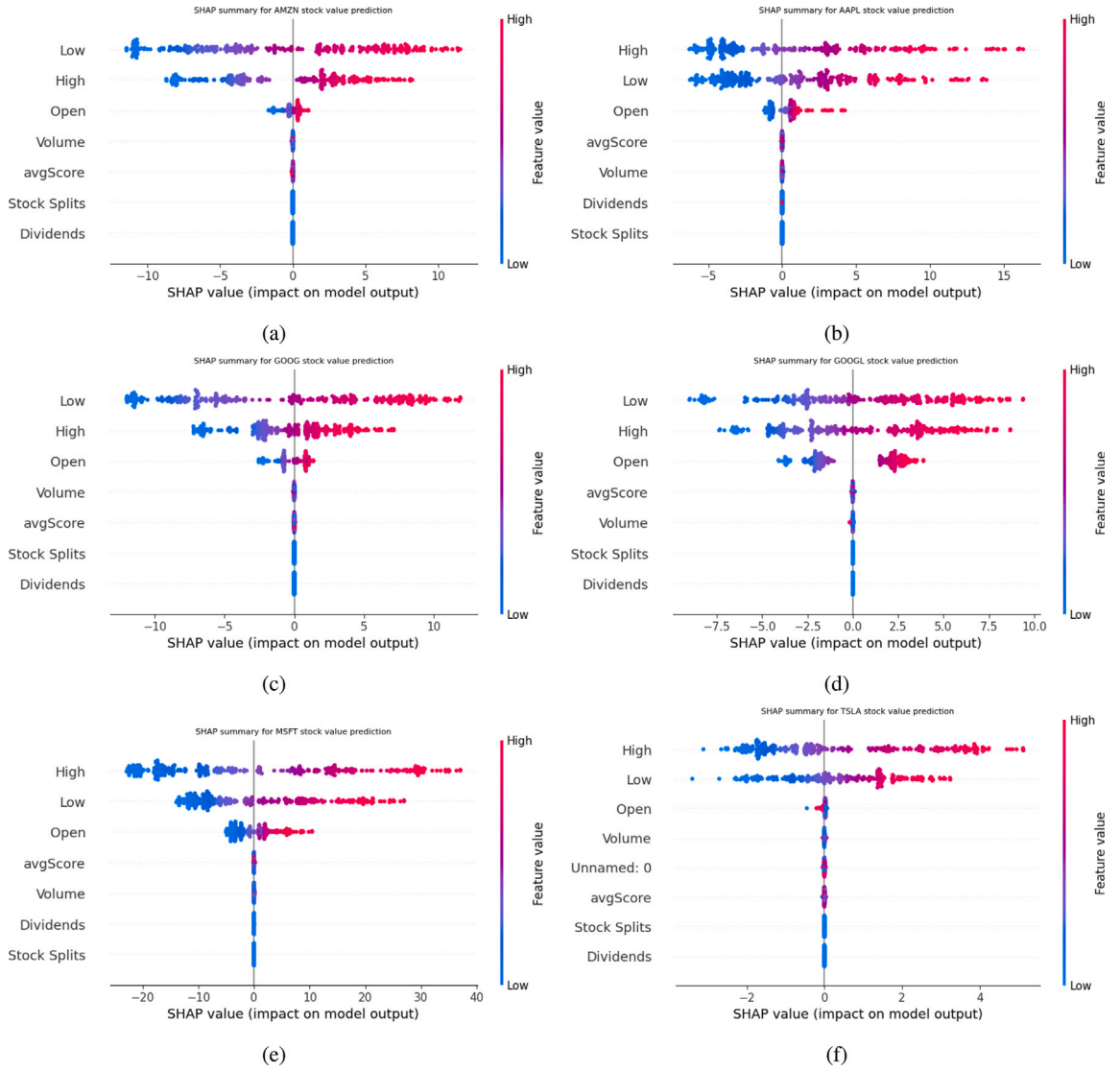


Fig. 7. Shap mean value analysis of hybrid quantum neural networks predictions. (a) Amazon. (b) Apple. (c) Google. (d) Google-L. (e) Microsoft. (f) Tesla.

Table 9
SRC and p-values.

Company name	SRC	Relation type	p-value	Significant
Apple (AAPL)	0.39	Medium correlation	$0.26e^{-5}$	Y
Amazon (AMZN)	0.21	Small correlation	$0.72e^{-5}$	Y
Google (GOOG)	0.23	Small correlation	$0.07e^{-3}$	Y
Google-L (GOOG-L)	0.33	Medium correlation	$0.24e^{-4}$	Y
Microsoft (MSFT)	0.03	No correlation	0.29	N
Tesla (TSLA)	0.59	Strong correlation	$0.39e^{-8}$	Y

From the comparison, we observe that Deep LSTM, Ensemble Deep RVFL, Ensemble Deep ESN, Quantum Decision Tree, Quantum SVM, Quantum Deep ANN, and Quantum Leap exhibit significantly higher mean rank differences compared to the proposed method. The significant differences suggest that, on average, these models perform less accurate than the proposed method across the datasets and the proposed method consistently outperforms these models in terms of mean rank across multiple datasets. Therefore, based on the analysis, the proposed method stands out as a robust and effective approach compared to the other models. The comparison of CD diagram for all the models is presented in Fig. 8.

Table 10
Average rank of each model using Friedman test.

Model	Average rank
Deep LSTM	7.66
Ensemble Deep RVFL	6.83
Ensemble Deep ESN	6.5
Quantum Decision Tree	4.16
Quantum SVM	4.66
Quantum Deep ANN	3.16
Quantum Leap	2.0
Proposed Method	1.0

Table 11
Significance of every model with the proposed method using Friedman and Nemenyi test.

Model	Mean rank difference	Significant (CD < Mean rank difference)
Deep LSTM	5.67	Yes
Ensemble Deep RVFL	4.82	Yes
Ensemble Deep ESN	3.00	Yes
Quantum Decision Tree	4.33	Yes
Quantum SVM	3.83	Yes
Quantum Deep ANN	3.00	Yes
Quantum Leap	2.67	Yes

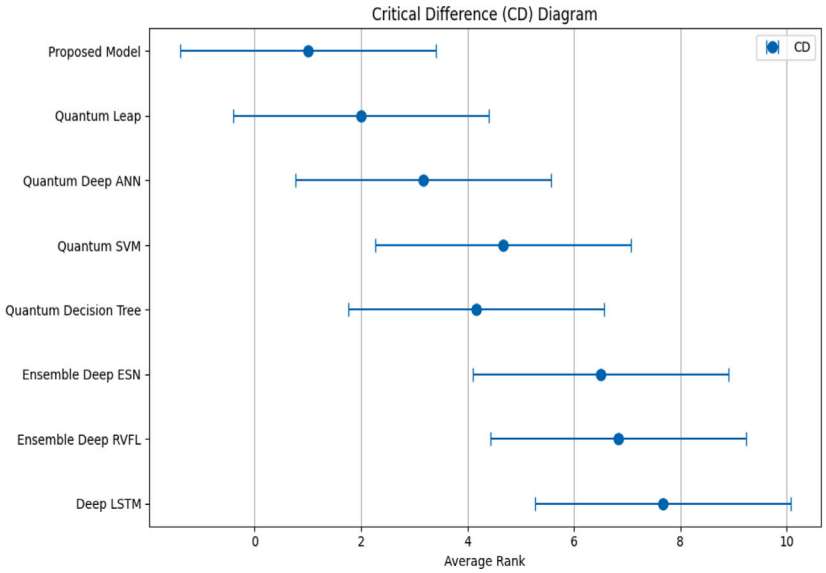


Fig. 8. Comparison of critical difference.

4.3. Ablation studies

We performed ablation studies to demonstrate the significance of various components in the proposed methodology. As part of the ablation study, we remove the specific components from the proposed methodology and we evaluate performance of the model. The outcomes are represented as follows (see Table 12). In Table 12, *w/o sentiment score* represents that we have not introduced the average sentiment score into technical indicators, *w/o hyperparameters embedding* represents that we have not included the tuned hyperparameters, *w/o QNN layer 1* represents the model without the first hidden layer, *w/o QNN layer 2* represents the model without the second hidden layer, and *w/o QNN layer 3* represents the model without the third hidden layer.

Based on the outcomes of the ablation study, we observe that the proposed methodology outperforms compared to the ablation models. Furthermore, we conclude that the tweets did not significantly influence the outcome prediction.

5. Conclusion and future work

Stock market data is very complex and generally hard for understanding the variations. In this study, we proposed explainable hybrid quantum neural networks to analyze the influence of tweets on stock price. We used historical stock price data and tweets for

Table 12
Ablation studies.

Company name	Evaluation metrics	w/o sentiment score	w/o parameters embedding	w/o QNN layer 1	w/o QNN layer 2	w/o QNN layer 3
Apple (AAPL)	MSE	0.46	13.22	12.48	15.27	11.17
	RMSE	0.67	3.63	3.53	3.90	3.34
	MAE	0.50	11.03	15.14	19.29	16.46
	(R^2)	98.62	82.17	88.74	86.21	89.12
Amazon (AMZN)	MSE	0.42	14.28	10.15	12.24	14.14
	RMSE	0.64	3.77	3.18	3.49	3.76
	MAE	0.64	2.12	1.95	2.48	2.62
	(R^2)	98.81	81.21	86.12	88.45	87.98
Google (GOOG)	MSE	0.19	21.48	28.82	34.47	29.93
	RMSE	0.43	4.63	5.36	5.87	5.47
	MAE	0.63	8.25	9.14	10.27	9.74
	(R^2)	98.96	79.96	82.14	81.17	83.19
Google-L (GOOG-L)	MSE	0.26	16.28	22.17	31.10	24.29
	RMSE	0.50	4.03	4.70	5.57	4.92
	MAE	0.55	8.11	12.19	19.46	14.81
	(R^2)	97.24	84.71	82.26	72.14	81.72
Microsoft (MSFT)	MSE	0.41	11.35	12.14	18.78	9.14
	RMSE	0.64	3.36	3.48	4.33	3.02
	MAE	0.83	2.14	3.01	2.87	3.44
	(R^2)	99.14	91.27	90.37	92.96	91.45
Tesla (TSLA)	MSE	0.34	10.48	11.69	19.41	9.89
	RMSE	0.58	3.23	3.14	4.40	3.14
	MAE	0.71	3.19	2.48	1.96	2.93
	(R^2)	98.94	89.92	87.21	91.17	92.24

5 calendar years of six different organizations. The accuracy of the proposed model is more than 99% on all six different data sets. Thus, the proposed hybrid quantum neural networks analyze the stock movements in an effective way and optimize the number of computations as compared to the classical machine learning/deep learning paradigms. Based on the explainability analysis, we observed that the attributes *open*, *high*, and *low* contribute to a greater extent to arrive at decisions of the proposed hybrid quantum neural network and the daily average tweet sentiment score did not influence the stock variations. However, it is essential to interpret these findings within the context of broader market dynamics and consider other factors that may influence stock price movements. Although the proposed explainable hybrid quantum neural networks predict the stock closing price in an efficient and accurate manner, the calculation of the daily average sentiment score from the tweet datasets is very complex and time consuming for the dataset having more than four million tweets. While X (Twitter) can be a valuable source of data for establishing certain assertions or trends, it is not the only source available. Depending on the nature of the assertion or the type of data needed, we can explore various other sources, including news articles and other online forums and social media communities. As a future avenue of exploration, further research may focus on developing sophisticated algorithms that leverage Bayesian optimization for ensemble compositions, potentially leading to effective forecasting performance across diverse and evolving datasets.

Code availability

(software application or custom code) Custom code developed.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Manoranjan Gandhudi reports financial support was provided by Conde Nast India Private Limited. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data is publicly available.

References

[1] Liu G, Ma W. A quantum artificial neural network for stock closing price prediction. Inform Sci 2022;598:75–85.
[2] Yıldırım DC, Toroslu IH, Fiore U. Forecasting directional movement of forex data using LSTM with technical and macroeconomic indicators. Financ Innov 2021;7:1–36.
[3] Wu D, Ma X, Olson DL. Financial distress prediction using integrated Z-score and multilayer perceptron neural networks. Decis Support Syst 2022;159:113814.
[4] Liu W, Yang Z, Cao Y, Huo J. Discovering the influences of the patent innovations on the stock market. Inf Process Manage 2022;59(3):102908.
[5] Yuan F, Zhan H. Stock market investment behavior based on behavioral finance based on data fusion algorithm. IETE J Res 2022;1–7.
[6] Xu H, Cao D, Li S. A self-regulated generative adversarial network for stock price movement prediction based on the historical price and tweets. Knowl-Based Syst 2022;247:108712.
[7] Ni H, Wang S, Cheng P. A hybrid approach for stock trend prediction based on tweets embedding and historical prices. World Wide Web 2021;24:849–68.
[8] Chiong R, Fan Z, Hu Z, Dhakal S. A novel ensemble learning approach for stock market prediction based on sentiment analysis and the sliding window method. IEEE Trans Comput Soc Syst 2022.
[9] Chaudhari K, Thakkar A. Data fusion with factored quantization for stock trend prediction using neural networks. Inf Process Manage 2023;60(3):103293.
[10] Yan X, Yang H, Yu Z, Zhang S, Zheng X. Portfolio optimization: A return-on-equity network analysis. IEEE Trans Comput Soc Syst 2023.
[11] Ma Y, Han R, Wang W. Portfolio optimization with return prediction using deep learning and machine learning. Expert Syst Appl 2021;165:113973.
[12] Liu J, Lim KH, Wood KL, Huang W, Guo C, Huang H-L. Hybrid quantum-classical convolutional neural networks. Sci China Phys Mech Astron 2021;64(9):290311.
[13] Adadi A, Berrada M. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). IEEE access 2018;6:52138–60.
[14] Omrani N, Rivieccio G, Fiore U, Schiavone F, Agreda SG. To trust or not to trust? An assessment of trust in AI-based systems: Concerns, ethics and contexts. Technol Forecast Soc Change 2022;181:121763.
[15] Kumar IE, Venkatasubramanian S, Scheidegger C, Friedler S. Problems with Shapley-value-based explanations as feature importance measures. In: International conference on machine learning. PMLR; 2020, p. 5491–500.
[16] Li Y, Pan Y. A novel ensemble deep learning model for stock prediction based on stock prices and news. Int J Data Sci Anal 2022;1–11.
[17] Carta S, Corriga A, Ferreira A, Podda AS, Recupero DR. A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning. Appl Intell 2021;51:889–905.
[18] Swathi T, Kasiviswanath N, Rao AA. An optimal deep learning-based LSTM for stock price prediction using twitter sentiment analysis. Appl Intell 2022;52(12):13675–88.
[19] Li X, Wu P, Wang W. Incorporating stock prices and news sentiments for stock market prediction: A case of Hong Kong. Inf Process Manage 2020;57(5):102212.
[20] Rezaei H, Faaljou H, Mansourfar G. Stock price prediction using deep learning and frequency decomposition. Expert Syst Appl 2021;169:114332.
[21] Long J, Chen Z, He W, Wu T, Ren J. An integrated framework of deep learning and knowledge graph for prediction of stock price trend: An application in Chinese stock exchange market. Appl Soft Comput 2020;91:106205.
[22] Leow EKW, Nguyen BP, Chua MCH. Robo-advisor using genetic algorithm and BERT sentiments from tweets for hybrid portfolio optimisation. Expert Syst Appl 2021;179:115060.
[23] Fiok K, Karwowski W, Gutierrez E, Wilamowski M. Analysis of sentiment in tweets addressed to a single domain-specific Twitter account: Comparison of model performance and explainability of predictions. Expert Syst Appl 2021;186:115771.
[24] Xu H, Chai L, Luo Z, Li S. Stock movement predictive network via incorporative attention mechanisms based on tweet and historical prices. Neurocomputing 2020;418:326–39.
[25] Liu H, Zhao T, Wang S, Li X. A stock rank prediction method combining industry attributes and price data of stocks. Inf Process Manage 2023;60(4):103358.
[26] Shi Q, Katuwal R, Suganthan PN, Tanveer M. Random vector functional link neural network based ensemble deep learning. Pattern Recognit 2021;117:107978.
[27] Paquet E, Soleymani F. QuantumLeap: Hybrid quantum neural network for financial predictions. Expert Syst Appl 2022;195:116583.
[28] Liu H, Long Z. An improved deep learning model for predicting stock market price time series. Digit Signal Process 2020;102:102741.
[29] Leippold M. Sentiment spin: Attacking financial sentiment with GPT-3. Finance Res Lett 2023;103957.
[30] Li L, Zhu F, Sun H, Hu Y, Yang Y, Jin D. Multi-source information fusion and deep-learning-based characteristics measurement for exploring the effects of peer engagement on stock price synchronicity. Inf Fusion 2021;69:1–21.
[31] Pira L, Ferrie C. An invitation to distributed quantum neural networks. Quantum Mach Intell 2023;5(2):1–24.
[32] Beer K, Bondarenko D, Farrelly T, Osborne TJ, Salzmann R, Scheiermann D, Wolf R. Training deep quantum neural networks. Nat Commun 2020;11(1):808.
[33] Acampora G, Chiatto A, Vitiello A. Genetic algorithms as classical optimizer for the quantum approximate optimization algorithm. Appl Soft Comput 2023;142:110296.
[34] Singh AK, Saxena D, Kumar J, Gupta V. A quantum approach towards the adaptive prediction of cloud workloads. IEEE Trans Parallel Distrib Syst 2021;32(12):2893–905.
[35] Sagheer A, Kotb M. Time series forecasting of petroleum production using deep LSTM recurrent networks. Neurocomputing 2019;323:203–13.
[36] Gao R, Li R, Hu M, Suganthan PN, Yuen KF. Dynamic ensemble deep echo state network for significant wave height forecasting. Appl Energy 2023;329:120261.
[37] Beigi S, Taghavi L. Quantum speedup based on classical decision trees. Quantum 2020;4:241.
[38] Anguita D, Ridella S, Rivieccio F, Zunino R. Quantum optimization for training support vector machines. Neural Netw 2003;16(5–6):763–70.
[39] Pereira DG, Afonso A, Medeiros FM. Overview of Friedman's test and post-hoc analysis. Comm Statist Simulation Comput 2015;44(10):2636–53.

Quantum Algorithms for Shapley Value Calculation

Iain Burge* , Michel Barbeau* , Joaquin Garcia-Alfaro†
*School of Computer Science, Carleton University, Ottawa, Canada
†SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, Palaiseau, France

***Abstract*—In the classical context, the cooperative game theory concept of the Shapley value has been adapted for post hoc explanations of Machine Learning (ML) models. However, this approach does not easily translate to eXplainable Quantum ML (XQML). Finding Shapley values can be highly computationally complex. We propose quantum algorithms which can extract Shapley values within some confidence interval. Our results perform in polynomial time. We demonstrate the validity of each approach under specific examples of cooperative voting games.**

***Index Terms*—Shapley Value, Quantum Computing, Cooperative Game Theory, Explainable Quantum Machine Learning, Machine Learning, Artificial Intelligence, Quantum Machine Learning.**

I. Introduction

Research in ML over the past decades deals with black box models. Unfortunately, black box models are inherently difficult to interpret. Inherently interpretable models would likely be best [19], as an explanation of an interpretable model is guaranteed to be correct. However, we ideally do not want to discard all of the previous research using black box models. As a result, there has been a substantial effort in implementing and improving post hoc explanation methods. Similarly, eXplainable Quantum ML (XQML) may benefit from adapting post hoc explanation methods to the quantum realm.

One of the most popular methods of generating post hoc explanations involves calculating Shapley values. Yet, classical strategies to approximate Shapley values are unwieldy to apply in the context of quantum computers. Therefore, it is necessary to explore a more native quantum solution to Shapley value approximation.

In this paper, we develop a flexible framework for the global evaluation of input factors in quantum circuits that approximates the Shapley values of such factors. Our framework has a one time increased circuit complexity of an additional $\mathcal{O}(an \log an)$ c-not gates, with a total additive increase in circuit depth of $\mathcal{O}(an)$, where n is the number of factors, and $a > 0$ is a real number. The change in space complexity for global evaluations is an additional $\mathcal{O}(\log an)$ qubits over the evaluated circuit. The circuit of increased complexity must then be repeated $\mathcal{O}(\epsilon^{-1})$ times. This procedure can achieve an error of $\mathcal{O}(a^{-1} + \epsilon)$ multiplied by a problem dependent upper bound. This starkly contrasts the $\mathcal{O}(2^n)$ assessments needed to calculate the Shapley values under the general case directly. It is also better than the $\mathcal{O}(\sigma^2\epsilon^{-2})$ complexity given by Monte Carlo approaches, where σ is standard deviation [15].

The paper is organized as follows[1]. Section II surveys related work. Section III provides background and preliminaries. Section IV introduces a guiding example problem. Sections V and VI present our methods. Section VII demonstrates the application of our methods to the problem discussed in Section IV. Section VIII concludes the work.

II. Related Work

Shapley values have been widely used to address multiple engineering problems, including regression, statistical analysis, and machine learning [12]. Finding Shapley values presents a difficult computational combinatorial problem. Our work proposes a novel quantum algorithm that reduces this combinatorial problem to an estimation problem which can leverage the power of quantum computation. Our approach performs in polynomial time. We apply our method to weighted voting games [13].

The deterministic computation of Shapley values in the context of weighted voting games is as difficult as NP-Hard [13], [17]. Since voting games are some of the simplest cooperative games, this result does not bode well for more complex scenarios. In the context of Shapley values for machine learning, it has also been shown that the calculation of Shapley values are not tractable for even regression models [23]. It was also proven that on the empirical distribution, finding a Shapley value takes exponential time [2].

The literature has also addressed the use of Shapley values on Quantum ML (QML). Indeed, XQML aims at adding explainability behind model predictions, e.g., in addition to providing classification [14]. XQML can be considered as an alternative research direction of QML instead of trying to justify quantum advantage [20]. The eventual goal of XQML is to provide, in addition to predictions, humanly understandable interpretations of the predictive models, e.g., for malware detection and classification, under a cybersecurity context [22].

Recent work by Heese et al. extends the notion of feature importance for model predictions in classical ML, to the QML realm [10]. In contrast to classical ML methods, in which Shapley values are applied to evaluate the importance of each feature for model predictions, Heese et al. apply Shapley values to evaluate the relevance of each quantum gate associated with a given parametrized quantum circuit. In their work, Heese et al. compute Shapley values involving stochastic processing. In general, the algorithms presented in this paper improve stochastic computation of Shapley values. Moreover,

[1]Peer-reviewed version [3].

our work justifies the interest in handling post hoc explanation frameworks such as the one of Heese et al., in a quantum extended manner.

III. Shapley Values

Cooperative game theory is the study of coalitional games. In this article, we are most interested in Shapley values. We now list some definitions and preliminaries,

Definition 1 (Coalitional game)**.** *A* coalitional game *is described as the pair $G = (F, V)$. $F = \{0, 1, ..., n\}$ is a set of $n + 1$ players. $V : \mathcal{P}(F) \to \mathbb{R}$ is a value function with $V(S) \in \mathbb{R}$ representing the value of a given coalition $S \subseteq F$, with the restriction that $V(\emptyset) = 0$.*

Definition 2 (Payoff vector)**.** *Given a game $G = (F, V)$, there exists a* payoff vector *$\Phi(G)$ of length $n + 1$. Each element $\Phi(G)_i \in \mathbb{R}$ represents the utility of player $i \in F$. A payoff vector is determined by the value function. Player i's payoff value $\Phi(G)_i$ is determined by how $V(S)$, $S \subseteq F$, is affected by i's inclusion or exclusion from S.*

There are a variety of solution concepts for constructing payoffs [1]. We focus on the Shapley solution concept, which returns a payoff vector [25]. Each element of the payoff vector $\Phi(G)_i$, is called player i's Shapley value. Shapley values have multiple different interpretations depending on the game being analyzed.

Shapley values are derived using one of several sets of axioms. We use the following four [25]. Suppose we have games $G = (F, V)$ and $G' = (F, V')$, and a payoff vector $\Phi(G)$, then:

1) Efficiency: The sum of all utility is equal to the utility of the grand coalition (the coalition containing all players),
$$\sum_{i=1}^{n} \Phi(G)_i = V(F)$$
2) Equal Treatment: Players i, j are said to be symmetrical if for all $S \subseteq F$, where $i, j \notin S$ we have that $V(S \cup \{i\}) = V(S \cup \{j\})$. If i, and j are symmetric in G, then they are treated equally, $\Phi(G)_i = \Phi(G)_j$.
3) Null Player: Consider a player $i \in F$, if for all $S \subseteq F$ such that $i \notin S$, we have $V(S) = V(S \cup \{i\})$, then i is a null player. If i is a null player then $\Phi(G)_i = 0$.
4) Additivity: If a player is in two games G and G', then the Shapley values of the two games is additive
$$\Phi(G + G')_i = \Phi(G)_i + \Phi(G')_i$$
where a game $G + G'$ is defined as $(F, V + V')$, and $(V + V')(S) = V(S) + V'(S)$, $S \subseteq F$.

These axioms lead to a single unique and intuitive division of utility [25]. The values of the payoff vectors can be interpreted as the responsibility of the respective players for the final outcome [9]. When player i has a small payoff $\Phi(G)_i$, then player i has a neutral impact on the final outcome. When player i has a large payoff, then player i has a large impact on the final outcome.

The Shapley value of i turns out to be the expected marginal contribution to a random coalition $S \subseteq F \setminus \{i\}$, where the marginal contribution is equal to $V(S \cup \{i\}) - V(S)$ [9].

Definition 3 (Shapley value [21])**.** *Let $G = (F, V)$, for simplicity sake, we write $\Phi(G)_i$ as Φ_i. The Shapley value of the i^{th} player Φ_i is described as:*
$$\Phi_i = \sum_{S \subseteq F \setminus \{i\}} \gamma(|F \setminus \{i\}|, |S|) \cdot (V(S \cup \{i\}) - V(S)) \quad (1)$$
where $n = |F \setminus \{i\}|$, and
$$\gamma(n, m) = \frac{1}{\binom{n}{m}(n+1)} = \frac{m!(n-m)!}{(n+1)!}$$

For this paper, we will define algorithms for a special case of games, monotonic games (Definition 4).

Definition 4 (monotonic game [21])**.** *A game is* monotonic *if for all $S, H \subseteq F$, we have, $V(S \cup H) \geq V(S)$.*

Note that when a game is monotonic, every summand in Equation (1) is non-negative.

IV. Weighted Voting Games

Let us consider a scenario where Shapley values correspond to voting power. The voting power of a player represents how many instances for which that player has the deciding vote. Three friends sit around a table. They are deliberating a grave matter. Should they get Chinese food for the second weekend in a row? They decide they should take a vote. Alice just got a promotion at work. To celebrate this, her friends agreed to give her three votes. Bob, who is the youngest of the group, also had good news, an incredible mark on his latest assignment! So, everyone decided Bob should get two votes. Charley, who had nothing to celebrate, gets one vote. The group decides to go out for Chinese food if there are four *yes* votes. In the end, all of the friends vote for Chinese food.

At the restaurant, they run into their friend David, who is a mathematician. David is intrigued when he hears about their vote. He begins to wonder how much power each friend had in the vote. The intuitive answer is that Alice had the most voting power, Bob had the second most, and Charley had the least. However, David notices something strange. Bob does not seem to have a more meaningful influence than Charley. There are no cases where Bob's two votes would do more than Charley's single vote. So, David concludes, there must be a more nuanced answer. Lucky for David, he has heard of cooperative game theory and Shapley values. So, he might be able to answer his question.

How can Definition 3 be applied to David's problem? Let us define the game $G = (F, V)$. The players are Alice (0), Bob (1), and Charley (2) represented as $F = \{0, 1, 2\}$. Denote each player's voting weight as $w_0 = 3$, $w_1 = 2$, and $w_2 = 1$. Recall that the vote threshold was $q = 4$. Then, for any subset $S \subseteq F$, we can define V as:
$$V(S) = \begin{cases} 1 & \text{if } \sum_{j \in S} w_j \geq q, \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

So, $V(S)$ is one if the sum of players' votes in S reaches the threshold of four. Otherwise, the vote fails, and so $V(S)$ is zero.

In general, this is called a weighted voting game [13]. One could easily add more players with arbitrary non-negative w_j and q. Note that weighted voting games fall into the family of monotonic games (Definition 4).

In this context, the terms in the Shapley value equation have an intuitive meaning. Take player i, and consider a set $S \subseteq F \setminus \{i\}$. If $V(S \cup \{i\}) - V(S) = 1$, then the i^{th} player is a deciding vote for the set of players S. Otherwise, player i is not a deciding vote.

Thus, for weighted voting games, player i's Shapley value represents a weighted count of how many times i is a deciding vote. We can work out the Shapley values by hand:

$$\begin{aligned} V(\emptyset) &= 0 & V(\{0,1\}) &= 1 \\ V(\{0\}) &= 0 & V(\{0,2\}) &= 1 \\ V(\{1\}) &= 0 & V(\{1,2\}) &= 0 \\ V(\{2\}) &= 0 & V(\{0,1,2\}) &= 1 \end{aligned}$$

From this, we have:

$$\begin{aligned} \Phi_0 &= \sum_{S \subseteq F \setminus \{i\}} \gamma(|F \setminus \{i\}|, |S|) \cdot (V(S \cup \{i\}) - V(S)) \\ &= \gamma(2,0) \cdot (V(\{0\}) - V(\emptyset)) + \gamma(2,1) \cdot (V(\{0,1\}) - V(\{1\})) \\ &\quad + \gamma(2,1) \cdot (V(\{0,2\}) - V(\{2\})) + \gamma(2,2) \cdot (V(\{0,1,2\}) \\ &\quad - V(\{1,2\})) = \gamma(2,0) \cdot (0-0) + \gamma(2,1) \cdot (1-0) \\ &\quad + \gamma(2,1) \cdot (1-0) + \gamma(2,2) \cdot (1-0) \\ &= 2 \cdot \gamma(2,1) + \gamma(2,2) = 2 \cdot \frac{1!(2-1)!}{(2+1)!} + \frac{2!(2-2)!}{(2+1)!} \\ &= 2 \cdot \frac{1}{6} + \frac{1}{3} = \frac{2}{3} \end{aligned}$$

This can be repeated to get,

$$\Phi_1, \Phi_2 = \frac{1}{6}$$

In the case of Alice, Bob, and Charley's voting game, it is trivial to calculate their respective Shapley values. However, what if one-hundred colleagues were choosing a venue for a party, all with different numbers of votes? In that case, a direct calculation would take 2^{100} assessments of V! for this more general case, we need to be clever.

V. Naive Quantum Representation of Shapley Value Calculation

We represent the Shapley value calculation problem in the quantum format. Consider an $n+1$ player monotonic game G represented by the pair (F, V), where $F = \{0, 1, \ldots, n\}$ and $V : \mathcal{P}(F) \to \mathbb{R}$, with $V(\emptyset) = 0$. Let us define the function,

$$W(S) = V(S \cup \{i\}) - V(S), \quad S \subseteq F \setminus \{i\}. \tag{3}$$

We define $W_{\max}$ as an upper bound of the absolute maximum change in value when adding player i to a subset S.

$$W_{\max} \geq \max_{S \subseteq F \setminus \{i\}} |W(S)|. \tag{4}$$

Let $i \in F$ be a given player. Consider the selection binary sequence $x = x_0 x_1 \ldots x_{i-1} x_{i+1} \ldots x_n$. Let S_x be the set of all players $j \in F$ such that $x_j = 1$. Then S_x could encode any player coalition that excludes the player i. Next, define,

$$\hat{W}(x) := \frac{W(S_x)}{W_{\max}}. \tag{5}$$

We define the following block diagonal matrix:

$$B = \bigoplus_{k=0}^{2^n-1} \begin{pmatrix} \sqrt{1-\phi(k,n)} & \sqrt{\phi(k,n)} \\ \sqrt{\phi(k,n)} & -\sqrt{1-\phi(k,n)} \end{pmatrix}$$

where

$$\phi(k,n) = \gamma(n, c(k)) \cdot \hat{W}(k)$$

and $c(k)$ is the number of ones in the binary representation of k, over $\log n$ bits.

Theorem 1. *The block diagonal matrix $B(n)$ is unitary.*

Proof. It follows from the fact that each block

$$\begin{pmatrix} \sqrt{1-\phi(k,n)} & \sqrt{\phi(k,n)} \\ \sqrt{\phi(k,n)} & -\sqrt{1-\phi(k,n)} \end{pmatrix}$$

of B is unitary. ∎

We construct the quantum system:

$$S = B(H^{\otimes n} \otimes I)|0\rangle^{\otimes n+1} \tag{6}$$

Theorem 2. *The expected values of the rightmost qubits of S is $\frac{\Phi_i}{2^n \cdot W_{max}}$.*

Proof. It follows from the fact that

$$(H^{\otimes n} \otimes I)|0\rangle^{\otimes n+1} = \sum_{k=0}^{2^n-1} \frac{1}{\sqrt{2^n}} |k\rangle |0\rangle$$

and the following sequence of equivalences:

$$\begin{aligned} &\sum_{k=0}^{2^n-1} (B_{k+1,k})^2 \\ &\quad = \sum_{k=0}^{2^n-1} \phi(k,n) = \sum_{k=0}^{2^n-1} \gamma(n, c(k)) \cdot W(k) \\ &\quad = \sum_{S \subseteq F \setminus \{i\}} \gamma(|F \setminus \{i\}|, |S|) \cdot \frac{(V(S \cup \{i\}) - V(S))}{W_{max}} \end{aligned}$$

Hence, the probability of measuring a one in the last qubit of the quantum system S is

$$\frac{1}{2^n \cdot W_{max}} \sum_{S \subseteq F \setminus \{i\}} \gamma(|F \setminus \{i\}|, |S|) \cdot (V(S \cup \{i\}) - V(S))$$

which is equivalent to $\frac{\Phi_i}{2^n \cdot W_{max}}$. ∎

The Shapley value Φ_i can be obtained by repeatedly creating the quantum system S, measuring their last qubit, taking the average, and finally multiplying by $2^n \cdot W_{max}$. As will be briefly discussed in Section VI, the value can also be extracted

with a more efficient strategy. Although, this representation requires the preparation of a quantum state with an exponential number of terms (2^n). In the following section, we develop a more efficient solution for Shapley value calculation of monotonic games.

VI. Quantum Algorithm for Monotonic Games

Consider an $n+1$ player game G represented by the pair (F, V), where $F = \{0, 1, \ldots, n\}$ and $V : \mathcal{P}(F) \to \mathbb{R}$, with $V(\emptyset) = 0$. A naive approach to finding the i^{th} player's Shapley value is through direct calculation using the Shapley Equation (1), completing the task in $\mathcal{O}(2^n)$ assessments of V. For structured games, it is occasionally possible to calculate Shapley values more efficiently. Otherwise, the only option is random sampling [6]. In each case, there are substantial trade-offs, in this section we propose a quantum algorithm which has some substantial advantages.

Suppose we have a quantum representation of the function $W(S)$, defined in Equation (3), which comprises two registers, a player register, and a utility register. In the player register, the computational basis states represent different subsets of players, where the j^{th} qubit represents the j^{th} player. In this encoding, the j^{th} qubit being $|1\rangle$ represents j's inclusion in the subset, and $|0\rangle$ represents its exclusion. Thus, every possible subset of players has a corresponding basis state. It follows that representing every subset of players simultaneously in a quantum superposition is possible. The one-qubit utility register encodes the output of W.

Computing the Shapley value method consists of the following steps:

1) Represent every possible subset of players in a quantum superposition, excluding player i, with amplitudes corresponding to weights ($\gamma(n, m)$) in the Shapley Equation (1).
2) Perform the quantum version of W controlled by the player register, and any auxiliary registers, on the utility register.

From this point, it is possible to approximate the Shapley value Φ_i if one has the expected value of measuring the utility register.

3) To approximate the expected value of measuring the utility register, one can either:
   - Use $\mathcal{O}(\epsilon^{-2})$ repetitions of steps 1 and 2 followed by a measurement of the utility register.
   - Perform amplitude estimation with $\mathcal{O}(\epsilon^{-1})$ iterations, where step 1 is $\mathcal{A}$ and step 2 is W as defined in [15]. In opposition to the previous choice, this process increases circuit depth.

In either case, the error in approximating the expected value for measuring the utility register is within $\pm\epsilon$ with some predetermined likelihood, where ϵ is a small positive real number.

The details of the method will now be described. For the sake of simplicity, we assume superadditivity; however, this algorithm could be easily extended to include general cooperative games [4]. The goal is to efficiently approximate the Shapley value Φ_i of a given player $i \in F$. Suppose quantum representation of function of function $\hat{W}(x)$, defined in Equation (5), is given as:

$$U_W |x\rangle |0\rangle := |x\rangle \left(\sqrt{1 - \hat{W}(x)} |0\rangle + \sqrt{\hat{W}(x)} |1\rangle \right),$$

where $|x\rangle$ is a vector in the computational basis (i.e., $x \in \{0, 1\}^n$).

A critical step for the algorithm is to approximate the weights of the Shapley value function. These weights correspond perfectly to a slightly modified beta function.

Definition 5 (Beta function)**.** *We define the beta function as:*

$$\beta_{n,m} = \int_0^1 x^m (1-x)^{n-m} dx, \quad 0 \le m \le n, \quad n, m \in \mathbb{N}.$$

Lemma 1. *We have the following recurrence relationship:*

$$\beta_{n,0} = \beta_{n,n} = \frac{1}{n+1} \text{ and } \beta_{n,m} = \frac{m}{n-(m-1)} \beta_{n,m-1}.$$

Proof. There are two cases.
Case 1 (m is equal to zero, or n). We have the following integration.

$$\beta_{n,0} = \int_0^1 (1-x)^n dx = -\frac{(1-x)^{n+1}}{n+1} \Big|_0^1 = \frac{1}{n+1}$$

A nearly identical calculation can be used to show $\beta_{n,n}$ is equal to $\frac{1}{n+1}$.
Case 2 ($0 < m < n$). We have the following partial integration.

$$\begin{aligned}
\beta_{n,m} &= \int_0^1 x^m (1-x)^{n-m} dx \\
&= \frac{x^m (1-x)^{n-(m-1)}}{n-(m-1)} \Big|_0^1 \\
&\quad - \int_0^1 \frac{-m}{n-(m-1)} x^{m-1} (1-x)^{n-(m-1)} dx \\
&= 0 + \frac{m}{n-(m-1)} \int_0^1 x^{m-1} (1-x)^{n-(m-1)} dx \\
&= \frac{m}{n-(m-1)} \beta_{n,m-1}
\end{aligned}$$

■

Theorem 3. *The beta function $\beta_{n,m}$ is equal to the Shapley weight function $\gamma(n, m)$, with $0 \le m \le n$ and $m, n \in \mathbb{N}$.*

Proof. The proof is by induction on m.
Base case ($m = 0$). According to Case 1 of Lemma 1, we have that $\beta_{n,0}$ is equal to $\frac{1}{n+1}$, which is equal to $\gamma(n, 0)$.
Inductive step ($m > 0$). Suppose $\beta_{n,k}$ is equal to $\gamma(n, k)$, $k \in \mathbb{N}$, we need to show $\beta_{n,k+1} = \gamma(n, k+1)$, $0 \le k < n$.

According to Case 2 of Lemma 1, $\beta_{n,k+1}$ is equal to $\frac{k+1}{n-k}\beta_{n,k}$. Using the inductive hypothesis, the latter is equivalent to

$$\frac{k+1}{n-k}\gamma(n,k) = \frac{k+1}{n-k}\cdot\frac{k!(n-k)!}{(n+1)!}$$

which matches the definition of $\gamma(n,k+1)$. ∎

The beta function, and by extension the Shapley weights, are approximated with Riemann sums which roughly represent the area under the function $x^m(1-x)^{n-m}$ using partitions of the interval $[0,1]$ with respect to x. It will become apparent that function $x^m(1-x)^{n-m}$ can be implemented efficiently on a quantum computer.

Definition 6 (Riemann sum)**.** *A* Riemann sum [18, page 276] *of a function f with respect to a partition $P = (t_0, \cdots, t_s)$ of the interval $[a,b]$ is an approximation of the integral of f from a to b of the form:*

$$\sum_{k=0}^{s-1}(t_{k+1}-t_k)\cdot f(x_k)$$

Where $t_{k+1}-t_k$ is the width of the subinterval, and $f(x_k), x \in [t_k, t_{k+1}]$, is the height.

The following is the initial quantum state:

$$|\psi_0\rangle = |0\rangle_{\text{Pt}} \otimes |0\rangle_{\text{Pl}} \otimes |0\rangle_{\text{Ut}}\,, \tag{7}$$

where Pt, Pl, and Ut respectively denote the partition, player, and utility registers. Pt is used it to prepare the $\gamma(n,m)$ weights. Suppose the number of qubits $\ell \in \mathbb{N}$ in Pt is $\ell = \lceil \log_2 an \rceil$, for some fixed $a > 0$, thus $\ell = \mathcal{O}(\log an)$. Then the partition register can be loaded with an arbitrary quantum state in $\mathcal{O}(an)$ time [16]. Let the number of qubits in the player register be n, one qubit per player excluding player i. Let the number of qubits in the utility register be one.

Consider the following function,

$$t_\ell(k) = \sin^2\left(\frac{\pi k}{2^{\ell+1}}\right) \text{ with } k = 0, 1, \ldots, 2^\ell$$

with which the partition $P_\ell = (t_\ell(k))_{k=0}^{2^\ell}$ of interval $[0,1]$ is constructed. This partition is computationally simple to implement on a quantum computer. Finally, let us define $w_\ell(k)$ to be the width of the k^{th} subinterval of P_ℓ, $w_\ell(k) = t_\ell(k+1) - t_\ell(k)$, $k = 0, 1, \ldots, 2^\ell - 1$.

Let us prepare the partition register to be,

$$\sum_{k=0}^{2^\ell-1}\sqrt{w_\ell(k)}\,|k\rangle\,.$$

the new state of the quantum system becomes,

$$|\psi_1\rangle = \sum_{k=0}^{2^\ell-1}\sqrt{w_\ell(k)}\,|k\rangle_{\text{Pt}}\,|0\rangle_{\text{Pl}}\,|0\rangle_{\text{Ut}}\,.$$

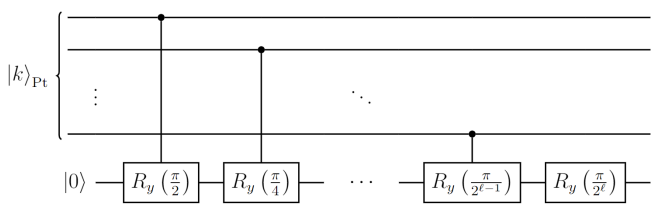


Fig. 1. This circuit R is a controlled rotation, where $R_y(\theta) = (\cos(\theta/2), -\sin(\theta/2); \sin(\theta/2), \cos(\theta/2))$. (Note: Library used for visualizing circuits can be found here: [11])

Next, perform a series of controlled rotations R (circuit in Figure 1) of the form,

$$R\,|k\rangle\,|0\rangle := |k\rangle\left(\sqrt{1-t'_\ell(k)}\,|0\rangle + \sqrt{t'_\ell(k)}\,|1\rangle\right),$$

where $t'_\ell(k) = t_{\ell+1}(2k+1)$ is used to sample the height of the k^{th} subinterval in the Riemann sum. Note that the value of $t'_\ell(k)$ is always somewhere in the range $[t_\ell(k), t_\ell(k+1)]$.

Unitary R is performed on each qubit in the player register, controlled by the partition register. The entirety of the applications of R can be performed with $\mathcal{O}(an\log an)$ gates in $\mathcal{O}(an)$ layers and yields the quantum state:

$$|\psi_2\rangle = \sum_{k=0}^{2^\ell-1}\sqrt{w_\ell(k)}\,|k\rangle_{\text{Pt}}\left(\sqrt{1-t'_\ell(k)}\,|0\rangle + \sqrt{t'_\ell(k)}\,|1\rangle\right)^{\otimes n}|0\rangle_{\text{Ut}}\,.$$

Note that the player register is of size n qubits. Let H_m be the set of binary numbers of hamming distance m from 0 in n bits, then we can rewrite $|\psi_2\rangle$ as:

$$|\psi_2\rangle = \sum_{k=0}^{2^\ell-1}\sqrt{w_\ell(k)}\,|k\rangle_{\text{Pt}}\cdot\sum_{m=0}^{n}\sqrt{t'_\ell(k)^m(1-t'_\ell(k))^{n-m}}\cdot\sum_{h\in H_m}|h\rangle_{\text{Pl}}\,|0\rangle_{\text{Ut}}$$

Note that, with this encoding style for S_x, x's hamming distance from 0 in n bits is equal to the size of S_x. In other words, if $h \in H_m$, then S_h contains m players.

Example 1. *Let us consider an example where the number of players is three (n is two). We have,*

$$\left(\sqrt{1-t'_\ell(k)}\,|0\rangle + \sqrt{t'_\ell(k)}\,|1\rangle\right)^{\otimes 2} = \sqrt{(1-t'_\ell(k))^2}\,|00\rangle + \sqrt{t'_\ell(k)(1-t'_\ell(k))}\,|01\rangle + \sqrt{t'_\ell(k)(1-t'_\ell(k))}\,|10\rangle + \sqrt{t'_\ell(k)^2}\,|11\rangle\,.$$

Note that $|00\rangle$ is hamming distance 0 from $|00\rangle$, $|01\rangle$ and $|10\rangle$ are hamming distance 1 from $|00\rangle$, and $|11\rangle$ is hamming distance 2 from $|00\rangle$. With this knowledge in hand, we can rewrite the quantum state of Example 1 as,

$$\sqrt{(1-t'_\ell(k))^2}\sum_{h\in H_0}|h\rangle + \sqrt{t'_\ell(k)(1-t'_\ell(k))}\sum_{h\in H_1}|h\rangle + \sqrt{t'_\ell(k)^2}\sum_{h\in H_2}|h\rangle$$

This can now be arranged into the general form,

$$\sum_{m=0}^{n}\sqrt{t'_\ell(k)^m(1-t'_\ell(k))^{n-m}}\sum_{h\in H_m}|h\rangle\,,\quad n=2.$$

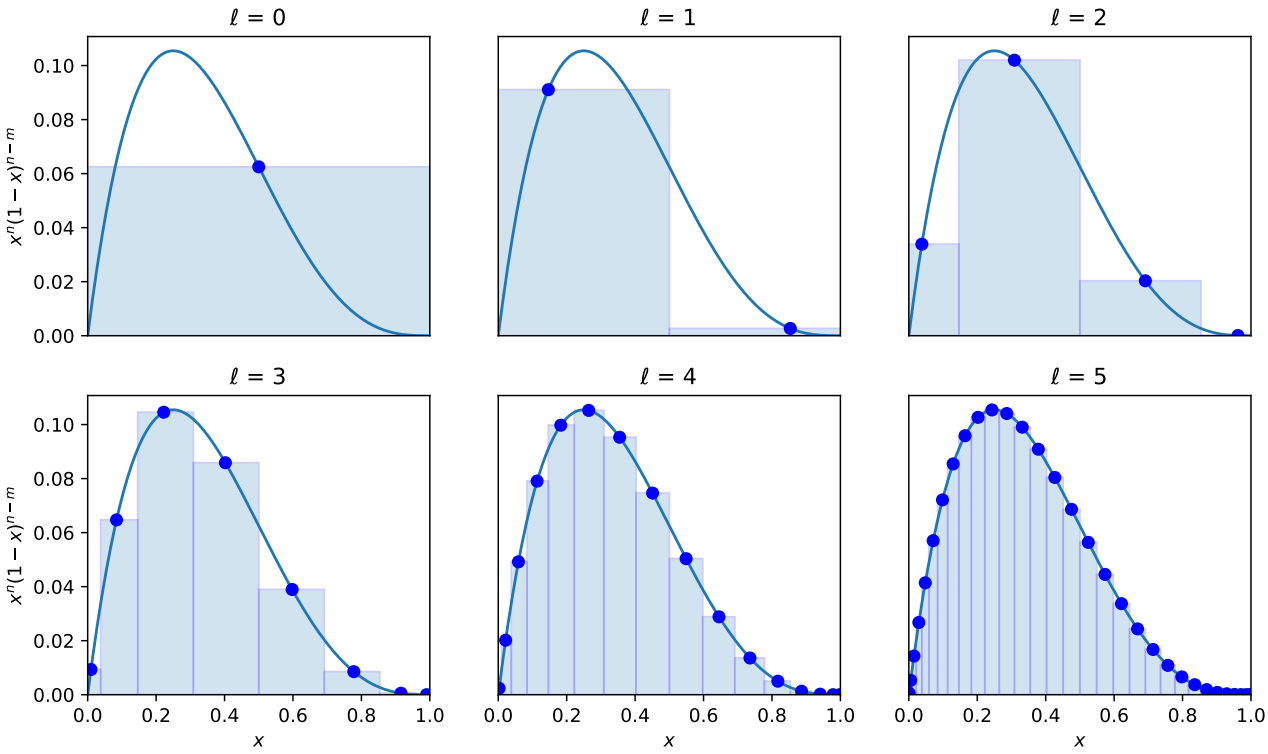


Fig. 2. Visual representation of $\beta_{n,m}$ being approximated using Riemann sums with partition P_ℓ over function $x^m(1-x)^{n-m}$, $t \in [0,1]$, $n = 4$, $m = 1$. The k^{th} rectangle's height is $(t'_\ell(k))^m(1 - t'_\ell(k))^{n-m}$, and its width is $w_\ell(k)$.

This completes the example.

Rearranging the quantum state $|\psi_2\rangle$ gives,

$$|\psi_2\rangle = \sum_{m=0}^{n} \sum_{h \in H_m} \sum_{k=0}^{2^\ell - 1} \sqrt{w_\ell(k) t'_\ell(k)^m (1 - t'_\ell(k))^{n-m}} \, |k\rangle_{\text{Pt}} \, |h\rangle_{\text{Pl}} \, |0\rangle_{\text{Ut}} \, .$$

Next, we perform U_W on the utility register controlled by the player register. For convenience, let us for the moment write $U_W \, |h\rangle \, |0\rangle = |h\rangle \, |W(h)\rangle$, where,

$$|W(h)\rangle = \sqrt{1 - \hat{W}(h)} \, |0\rangle + \sqrt{\hat{W}(h)} \, |1\rangle$$

Applying $U_W \, |h\rangle_{\text{Pl}} \, |0\rangle_{\text{Ut}}$ gives $|\psi_3\rangle$, which is equal to,

$$\sum_{m=0}^{n} \sum_{h \in H_m} \sum_{k=0}^{2^\ell - 1} \sqrt{w_\ell(k) t'_\ell(k)^m (1 - t'_\ell(k))^{n-m}} \, |k\rangle_{\text{Pt}} \, |h\rangle_{\text{Pl}} \, |W(h)\rangle_{\text{Ut}} \, .$$

This operation is wholly dependent on the game being analyzed and its complexity. Assuming the algorithm is being implemented with a look-up table, one could likely use qRAM [7]. This approach has a time complexity of $\mathcal{O}(n)$ at the cost of $\mathcal{O}(2^n)$ qubits for storage. However, depending on the problem, there are often far less resource-intense methods of implementing U_W, as will be seen with the implementation of weighted voting games (Section VII).

This is the final quantum state. Let us now analyze this state through the lens of density matrices. Taking the partial trace with respect to the partition and player registers yields,

$$\text{tr}_{\text{Pt,Pl}} \left(|\psi_3\rangle\langle\psi_3| \right) = \sum_{m=0}^{n} \sum_{h \in H_m} \left(\sum_{k=0}^{2^\ell - 1} w_\ell(k) t'_\ell(k)^m (1 - t'_\ell(k))^{n-m} \right) \cdot |W(h)\rangle_{\text{Ut}} \langle W(h)|_{\text{Ut}} \, .$$

Theorem 4. *The Riemann sum using partition P_ℓ to approximate area under $x^m(1-x)^{n-m}$ for $x \in [0,1]$ asymptotically approaches $\gamma(n,m)$. Formally,*

$$\lim_{\ell \to \infty} \sum_{k=0}^{2^\ell - 1} w_\ell(k) t'_\ell(k)^m (1 - t'_\ell(k))^{n-m} = \gamma(n,m) \tag{8}$$

Proof. Let $f(x) = x^m(1-x)^{n-m}$, and recall our definition for partition of $[0,1]$, $P_\ell = (t_\ell(k))_{k=0}^{2^\ell}$.

Define $\text{mesh}(P_\ell) = \sup\{w_\ell(k) : k = 0, 1, \ldots, 2^\ell - 1\}$ [18, page 275]. Since,

$$w_\ell(k) = \sin^2\left(\frac{\pi(k+1)}{2^{\ell+1}}\right) - \sin^2\left(\frac{\pi k}{2^{\ell+1}}\right),$$

we can bound $w_\ell(k)$,

$$w_\ell(k) \leq \frac{\pi}{2}\left(\left(\frac{\pi(k+1)}{2^{\ell+1}}\right) - \left(\frac{\pi k}{2^{\ell+1}}\right)\right) = \frac{\pi^2}{2^{\ell+2}}.$$

This is valid because $d/dx \sin^2((\pi/2)x) \leq d/dx(\pi/2)x$ for all $x \in [0,1]$. It is then clear that as $\ell \to \infty$, $\text{mesh}(P_\ell) \to 0$. Additionally, f is integrable over $[a,b]$, as it is a polynomial and hence continuous [18, page 278].

By [18, page 278, Corollary 32.10], as ℓ increases, any Riemann sum of f using partition P_ℓ converges to:

$$\int_0^1 x^m (1-x)^{n-m},$$

which is equal to $\gamma(n, m)$. ∎

The Riemann sum approximation of the modified beta function, $\beta_{n,m} = \gamma(n, m)$, is visualized in Figure 2. Applying a Riemann sum approximation for $\gamma(n, m)$, we have,

$$\mathrm{tr}_{\mathrm{Pt,Pl}}(|\psi_3\rangle\langle\psi_3|) \approx \sum_{m=0}^{n} \sum_{h \in H_m} \gamma(n, m) \, |W(h)\rangle_{\mathrm{Ut}} \langle W(h)|_{\mathrm{Ut}} \, .$$

Finally, suppose we measuring the utility register in the computational basis. This yields the following expected value with empirical error less than $\mathcal{O}(a^{-1})$ (Section VII):

$$\sum_{m=0}^{n} \sum_{h \in H_m} \gamma(n, m) \hat{W}(h)$$

Plugging in the definition for $\hat{W}$, we have

$$\frac{1}{W_{\max}} \sum_{m=0}^{n} \sum_{h \in H_m} \gamma(n, m) \left(V\left(S_h \cup \{i\}\right) - V\left(S_h\right) \right)$$

Notice that, in the S_x encoding, H_m represents each subset of $F \setminus \{i\}$ of size m. As a result, the equation is, in effect, summing over each subset of $F \setminus \{i\}$. As a final step, multiply by $W_{\max}$:

$$\sum_{S \subseteq F \setminus \{i\}} \gamma(|F \setminus \{i\}|, |S|) \cdot \left(V\left(S \cup \{i\}\right) - V\left(S\right) \right).$$

This expected value is precisely the Shapley value Φ_i to some error which is shown empirically in Section VII.

With the ability to craft these states, we can now extract the required information to find a close approximation to the Shapley value. Assuming we could get the expected value instantly, we would (empirically; see Section VII) have an error of $\mathcal{O}(a^{-1})$. However, it takes some work to approximate an expected value. This can be achieved with accuracy ϵ with some chosen confidence using amplitude amplification [15], resulting in circuit depth $\mathcal{O}(\epsilon^{-1})$ times our algorithm. Alternatively, for simplicity, we can repeatedly construct state $|\psi_3\rangle$ and measure the utility register roughly $\mathcal{O}(\epsilon^{-2})$ times. The resulting measurements of $|0\rangle$s and $|1\rangle$s can be analyzed using binomial distributions. We can estimate the underlying probability, from which we can extract the Shapley value, within some confidence interval [24]. This estimation takes up the bulk of the runtime of the algorithm. Thus the total error can be as low as $\mathcal{O}(W_{\max} \cdot (a^{-1} + \epsilon))$. This can be even further modified to depend on the standard deviation of V, based on the techniques from [15].

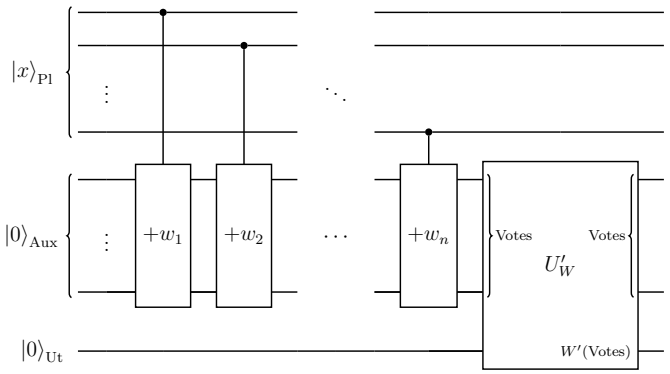


Fig. 3. Circuit of U_W for a weighted voting game. This circuit takes an input x and outputs $W(S_x)$ in the utility register (Recall, S_x is defined in Section VI). The auxiliary register contains the total vote count. Just before U'_W, the Aux register is in a basis state corresponding to the vote count of S_x. U'_W uses the auxiliary register as an input, and outputs whether the vote count is in the correct range for player i to be a deciding vote. Note that there is no $+w_i$ gate activation, as S_x does not include player i by definition. Results and simulation code are available in a companion GitHub repository [5].

VII. Numerical Examples

Perhaps we can help David solve his problem (cf. Section IV) using our quantum approach. We intend to apply the method presented in Section VI for weighted voting games. Additional results, together with the simulation code, are available in a companion github repository [5]. Let us approximate each player's Shapley value, Φ_i. We have a game $G = (F, V)$, where $F = \{0, 1, 2\}$, $n = 2$, and V is defined in Equation 2. In this case, $W(S)$, $S \subseteq F \setminus \{i\}$, represents whether or not player i has the deciding vote assuming those in S are voting for Chinese food. For example, Alice (0) is a deciding vote for $S = \{1\}$ (S contains Bob). Let the voting weights be $w_0 = 3$, $w_1 = 2$, and $w_2 = 1$. Thus, we can define $W(S)$, $S \in F \setminus \{i\}$, for player i to be,

$$W(S) = W'\left(\sum_{j \in S} w_j\right)$$

where,

$$W'(Votes) = \begin{cases} 1 & \text{if } q - w_i \leq Votes < q, \\ 0 & \text{otherwise.} \end{cases}$$

Note that, $W(S)$ is either 0 or 1. Thus, we can take $W_{\max} = 1$, so $\hat{W}(x) = W(S_x)$. Thus we can define U_W to be:

$$U_W \, |x\rangle \, |0\rangle = |x\rangle \otimes \left[\left(1 - \hat{W}(x)\right) |0\rangle + \hat{W}(x) \, |1\rangle \right]$$

The quantum circuit for the scenario is shown in Figure 3.

With U_W defined, all other steps are entirely agnostic to voting games. Let $\ell = 2$, and suppose for simplicity's sake $\epsilon = 0$. This is not a realistic scenario, but it demonstrates how quickly the expected value of the utility register converges. With these parameters, we get the following approximations for the Shapley values:

$$\tilde{\Phi}_0 \approx 0.6617, \quad \tilde{\Phi}_1, \tilde{\Phi}_2 \approx 0.1616$$

The direct calculation for Alice can be seen in the Appendix.

To rigorously demonstrate efficacy, we performed many trials on random weighted voting games (Figure 4). Visual inspection confirms that, in most cases, when ℓ is incremented, the error is divided by more than a factor of two (so the reciprocal more than doubles). As incrementing ℓ implies doubling the work to prepare $|\psi_1\rangle$, we have empirically shown a linear or better relationship between complexity and error for Step 1 (Section VI). So, the empirical error for this step is $\mathcal{O}(a^{-1})$.

Step 2 depends entirely on the game. However, if it is possible to implement on a classical computer, much like with Grover's algorithm, then it can be implemented in a quantum setting [8].

Step 3 is well studied, and there are two valid approaches to finding the expected value of measuring the utility register with error ϵ. Either (i), repeat steps one and two of Section VI and measure the utility register $\mathcal{O}(\epsilon^{-2})$ times. Otherwise, proceed with (ii), use amplitude estimation as described in Section VI with the techniques described in [15]. Note that (ii) results in an increase to the circuit depth by $\mathcal{O}(\epsilon^{-1})$ times, but only needs to be repeated $\mathcal{O}(1)$ times assuming a fixed confidence probability.

Let us summarize based on our empirical findings and the assumption W is more complex than Step 1. We have that complexity in terms of applications of W is $\mathcal{O}(\epsilon^{-1})$ in the case of weighted voting games. For more complex problems, where the value function W is not restricted to $W : \mathcal{P}(F) \to [0, 1]$ the approach can be slightly modified such that W is applied $\mathcal{O}(\sigma\epsilon^{-1})$ times [15], where σ is the standard deviation of the W given the γ distribution.

VIII. Conclusion

We have addressed one of the main problems of generating post hoc explanations of ML models in the quantum context. More precisely, we have addressed the problem of efficiently generating post hoc explanations using the concept of the Shapley value. Under the XQML context, the challenge of explainability is amplified since measuring a quantum system destroys the information. Using the classical concept of Shapley values for post hoc explanations in quantum computing does not translate trivially. We have proposed novel algorithms to reduce the problem of accurately estimating the Shapley values of a quantum algorithm into the solved problem of estimating the expected value of a quantum algorithm. Finally, we have determined the efficacy of the algorithms by using empirical and simulation methods.

Acknowledgments — We thank Prof. Yuly Billig (Carleton University) for his notational suggestion regarding hamming distances. We would also like to thank a peer of Iain, Michael Ripa, for many productive conversations regarding this research. The research was supported by Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

[1] R. J. Aumann. Some non-superadditive games, and their Shapley values, in the Talmud. *International Journal of Game Theory*, 39:1–10, 2010.

[2] L. Bertossi, J. Li, M. Schleich, D. Suciu, and Z. Vagena. Causality-based explanation of classification outcomes. In *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*, pages 1–10, 2020.

[3] I. Burge, M. Barbeau, and J. Garcia-Alfaro. Quantum Algorithms for Shapley Value Calculation, 2023 IEEE International Conference on Quantum Computing and Engineering (QCE), https://doi.org/10.1109/QCE57702.2023.00024, November 2023.

[4] I. Burge, M. Barbeau, and J. Garcia-Alfaro. A Quantum Algorithm for Shapley Value Estimation, arXiv:2301.04727, https://doi.org/10.48550/arXiv.2301.04727, March 2023.

[5] I. Burge, M. Barbeau, and J. Garcia-Alfaro. Quantum Algorithms for Shapley Value Calculation [extended simulation github repository], github, https://github.com/iain-burge/QuantumShapleyValueAlgorithm, May 2023.

[6] J. Castro, D. Gómez, and J. Tejada. Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009.

[7] V. Giovannetti, S. Lloyd, and L. Maccone. Quantum random access memory. *Physical review letters*, 100(16):160501, 2008.

[8] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[9] S. Hart. Shapley value. In *Game theory*, pages 210–216. The New Palgrave. Palgrave Macmillan, London, 1989.

[10] R. Heese, T. Gerlach, S. Mücke, S. Müller, M. Jakobs, and N. Piatkowski. Explaining quantum circuits with shapley values: Towards explainable quantum machine learning, arXiv: 2301.09138, https://doi.org/10.48550/arXiv.2301.09138, March 2023.

[11] A. Kay. Tutorial on the Quantikz package, arXiv: 1809.03842, https://arxiv.org/abs/1809.03842, March 2023.

[12] S. Lipovetsky. Quantum-like data modeling in applied sciences. *Stats*, 6(1):345–353, 2023.

[13] Y. Matsui and T. Matsui. NP-completeness for calculating power indices of weighted majority games. *Theoretical Computer Science*, 263(1-2):305–310, 2001.

[14] F. Mercaldo, G. Ciaramella, G. Iadarola, M. Storto, F. Martinelli, and A. Santone. Towards explainable quantum machine learning for mobile malware detection and classification. *Applied Sciences*, 12(23):12025, 2022.

[15] A. Montanaro. Quantum speedup of Monte Carlo methods. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2181):20150301, 2015.

[16] M. Plesch and Č. Brukner. Quantum-state preparation with universal gate decompositions. *Physical Review A*, 83(3):032302, 2011.

[17] K. Prasad and J. S. Kelly. NP-completeness of some problems concerning voting games. *International Journal of Game Theory*, 19(1):1–9, 1990.

[18] K. A. Ross. *Elementary analysis*. Springer, New York, NY, 2013.

[19] C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[20] M. Schuld and N. Killoran. Is quantum advantage the right goal for quantum machine learning? *PRX Quantum*, 3(3):030101, jul 2022.

[21] L. S. Shapley. *A Value for N-Person Games*. RAND Corporation, Santa Monica, CA, 1952.

[22] P. Steinmüller, T. Schulz, F. Graf, and D. Herr. eXplainable AI for Quantum Machine Learning. *arXiv preprint arXiv:2211.01441*, 2022.

[23] G. Van den Broeck, A. Lykov, M. Schleich, and D. Suciu. On the tractability of SHAP explanations. *Journal of Artificial Intelligence Research*, 74:851–886, 2022.

[24] S. Wallis. Binomial confidence intervals and contingency tests: mathematical fundamentals and the evaluation of alternative methods. *Journal of Quantitative Linguistics*, 20(3):178–208, 2013.

[25] E. Winter. The Shapley value. *Handbook of game theory with economic applications*, 3:2025–2054, 2002.

Appendix

Quantum estimation of Alice's Shapley value by hand. Let $\ell = 2$. Note that the Auxiliary register stores vote count. We begin with the state:

$$|00\rangle_{\text{Pt}} |00\rangle_{\text{Pl}} |000\rangle_{\text{Aux}} |0\rangle_{\text{Ut}}$$

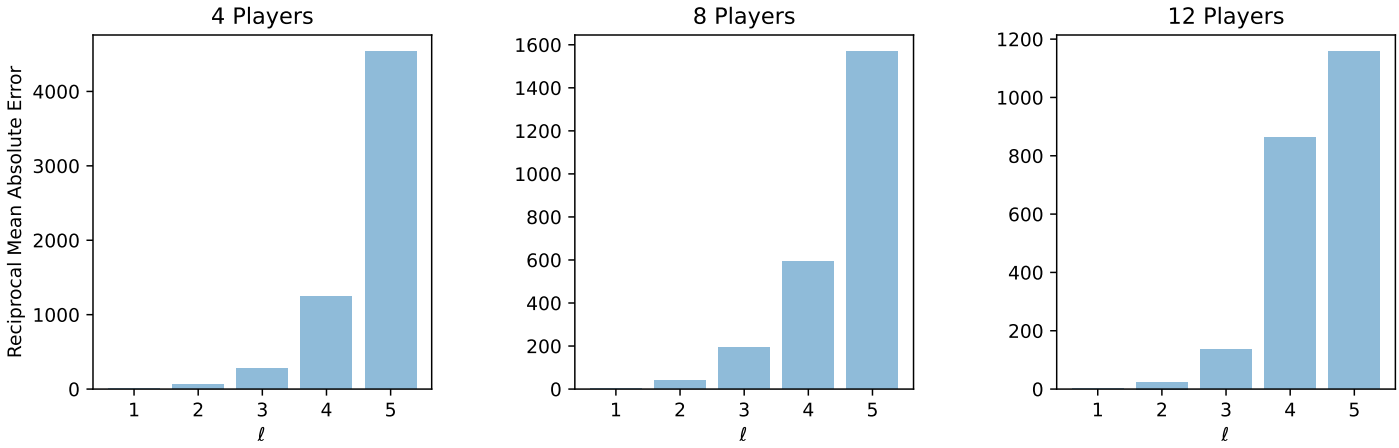


Fig. 4. We generated 32 random weighted voting games for each condition. We generated random weights $w_j \in \mathbb{N}$ for each case such that $q \leq \sum_j w_j < 2q$. There were three primary scenarios: (1) Four players, voting threshold $q = 8$; (2) Eight players, voting threshold $q = 16$; and (3) 12 players, voting threshold $q = 32$. We approximated every player's Shapley value for each scenario with our quantum algorithm using various ℓ's, assuming $\epsilon = 0$. Next, we found the absolute error of our approximation by comparing each approximated Shapley value to its true value. Finally, we took the reciprocal of the mean for all Shapley value errors in each random game for each condition.

We perform the first step of the algorithm from Section VI, yielding:

$$\sum_{k=0}^{3} w_2(k) \left|k\right\rangle_{\text{Pt}} \Big[(1 - t_2'(k)) \left|00\right\rangle_{\text{Pl}} + \sqrt{t_2'(k)(1 - t_2'(k))} \left|01\right\rangle_{\text{Pl}} + \sqrt{t_2'(k)(1 - t_2'(k))} \left|10\right\rangle_{\text{Pl}} + t_2'(k) \left|11\right\rangle_{\text{Pl}} \Big] \left|000\right\rangle_{\text{Aux}} \left|0\right\rangle_{\text{Ut}}$$

Next, we tally the votes. This step is the first half of the circuit in Figure 3, up to and without including U_W'. We get,

$$\sum_{k=0}^{3} w_2(k) \left|k\right\rangle_{\text{Pt}} \Big[(1 - t_2'(k)) \left|00\right\rangle_{\text{Pl}} \left|000\right\rangle_{\text{Aux}} + \sqrt{t_2'(k)(1 - t_2'(k))} \left|01\right\rangle_{\text{Pl}} \left|001\right\rangle_{\text{Aux}} + \sqrt{t_2'(k)(1 - t_2'(k))} \left|10\right\rangle_{\text{Pl}} \left|010\right\rangle_{\text{Aux}} + t_2'(k) \left|11\right\rangle_{\text{Pl}} \left|011\right\rangle_{\text{Aux}} \Big] \left|0\right\rangle_{\text{Ut}}$$

Performing the remainder of U_W gives,

$$\sum_{k=0}^{3} w_2(k) \left|k\right\rangle_{\text{Pt}} \Big[(1 - t_2'(k)) \left|00\right\rangle_{\text{Pl}} \left|000\right\rangle_{\text{Aux}} \left|0\right\rangle_{\text{Ut}} + \sqrt{t_2'(k)(1 - t_2'(k))} \left|01\right\rangle_{\text{Pl}} \left|001\right\rangle_{\text{Aux}} \left|1\right\rangle_{\text{Ut}} + \sqrt{t_2'(k)(1 - t_2'(k))} \left|10\right\rangle_{\text{Pl}} \left|010\right\rangle_{\text{Aux}} \left|1\right\rangle_{\text{Ut}} + t_2'(k) \left|11\right\rangle_{\text{Pl}} \left|011\right\rangle_{\text{Aux}} \left|1\right\rangle_{\text{Ut}} \Big]$$

The expected value when measuring the utility register is ≈ 0.6617, a close approximation for $\Phi_0 = 2/3 = 0.6666\cdots$.

On the Interpretability of Quantum Neural Networks

Lirandë Pira[1,*] and Chris Ferrie[1]

[1] *University of Technology Sydney, Centre for Quantum Software and Information, Ultimo NSW 2007, Australia*

(Dated: April 22, 2024)

Interpretability of artificial intelligence (AI) methods, particularly deep neural networks, is of great interest. This heightened focus stems from the widespread use of AI-backed systems. These systems, often relying on intricate neural architectures, can exhibit behavior that is challenging to explain and comprehend. The interpretability of such models is a crucial component of building trusted systems. Many methods exist to approach this problem, but they do not apply straightforwardly to the quantum setting. Here, we explore the interpretability of quantum neural networks using local model-agnostic interpretability measures commonly utilized for classical neural networks. Following this analysis, we generalize a classical technique called LIME, introducing Q-LIME, which produces explanations of quantum neural networks. A feature of our explanations is the delineation of the region in which data samples have been given a random label, likely subjects of inherently random quantum measurements. We view this as a step toward understanding how to build responsible and accountable quantum AI models.

I. INTRODUCTION

Artificial intelligence (AI) has become ubiquitous. Often manifested in machine learning algorithms, AI systems promise to be evermore present in everyday high-stakes tasks [1, 2]. This is why building fair, responsible, and ethical systems is crucial to the design process of AI algorithms. Central to the topic of *trusting* AI-generated results is the notion of *interpretability*, also known as *explainability*. The interpretability of AI models is a pivotal concern in contemporary AI research, particularly with the ubiquity of deep neural networks. This has given rise to research topics under the umbrella of interpretable machine learning (or IML) and explainable AI (or XAI), noting that the terms *interpretable* and *explainable* are used synonymously throughout the corresponding literature. Generically, interpretability is understood as the extent to which humans comprehend the output of an AI model that leads to decision-making [3]. Humans strive to understand the "thought process" behind the decisions of the AI model — otherwise, the system is referred to as a "black box". Even though the terms "interpretation" and "explanation" are used colloquially in the literature with varying degrees, throughout this paper, we use them as follows. *Interpretable* entails a model that humans can understand and comprehend through direct observation of its internal workings or outputs. It implies that the model is intuitive to the human observer and that no further tools are required. On the other hand, *explanation* refers to the output of a tool used to articulate the behavior of a model. In our case, and indeed many others, an explanation is a model itself, which is justifiably called such when it is interpretable. Complex models and their simpler explanations are often called black-box and white-box models, respectively.

The precise definition of a model's interpretability has been the subject of much debate [4, 5]. Naturally, there exist learning models which are more interpretable than others, such as simple decision trees. On the other hand, the models we prefer best for solving complex tasks, such as deep neural networks (DNNs), happen to be highly non-interpretable, which is due to their inherent non-linear layered architecture [6]. We note that DNNs are one of the most widely used techniques in machine learning. Thus, the interpretability of neural networks is an essential topic within the field of interpretable ML research [7, 8]. In this work, we focus on precisely the topic of interpretability as we consider the quantum side of neural networks.

In parallel, recent years have witnessed a surge of research efforts in *quantum* machine learning (QML)

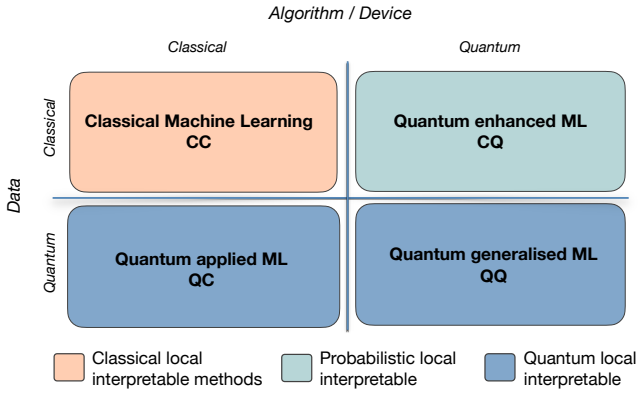


FIG. 1. **Categorization of interpretability techniques as they apply to classical and quantum resources.** Here, the well-known QML diagram represents data and an algorithm or device, which can be classical (C) or quantum (Q) in four different scenarios. We consider a reformulation of interpretable techniques to be required in the CQ scenario. In the QC and QQ quadrants, the design of explicitly quantum interpretable methods may be required. The scope of this paper covers CQ approaches.

* lirande.pira@student.uts.edu.au

[9, 10]. This research area sits at the intersection of machine learning and quantum computing. The development of QML has undergone different stages. Initially, the field started with the quest for speedups or quantum advantages. More recently, the target has morphed into further pursuits in expressivity and generalization power of quantum models. Nowadays, rather than "competing" with classical models, quantum models are further being enhanced on their own, which could, in turn, improve classical machine learning techniques. One of the key techniques currently used in QML research is the variational quantum algorithm, which acts as the quantum equivalent to classical neural networks [11]. To clarify the analogy, we will refer to such models as quantum neural networks (QNNs) [12].

Given the close conceptual correspondence to classical neural networks, it is natural to analyze their interpretability, which is important for several reasons. Firstly, QNNs may complement classical AI algorithm design, making their interpretability at least as important as classical DNNs. Secondly, the quantum paradigms embedded into QNNs deserve to be understood and explained in their own right. The unique non-intuitive characteristics of quantum nature can make QNNs more complicated to interpret from the point of view of human understandability. Finally, with the growing interest and capabilities of quantum technologies, it is crucial to identify and mitigate potential sources of errors that plague conventional AI due to a lack of transparency.

In this work, we define and establish key notions pertaining to the interpretability of quantum neural networks. Through this formalization, we aim to contribute a robust foundation for advancing the discourse on the interpretability of quantum neural networks within the broader context of quantum machine learning research. In doing so, we generalize some well-known interpretable techniques to the quantum domain. Consider the standard relationship diagram in QML between data and algorithm (or device) type, where either can be classical (C) or quantum (Q). This entails the following combinations (CC, CQ, QC, and QQ), shown in Fig. 1. Classical interpretable techniques are the apparent domain of CC. We will discuss, but not dwell on, the potential need for entirely new techniques when the data is quantum (QC and QQ). In CQ, the domain that covers the so-called quantum-enhanced machine learning techniques, although the data is classical, the output of the quantum devices is irreversibly probabilistic. CQ can be seen as a specialization of the quantum data techniques above. Generalizing classical notions of interpretability to this domain is the subject of our work.

The question of interpretability in quantum machine learning models more broadly, as well as of QNNs more specifically, has already started to receive attention [13–15], particularly involving the concept of Shapley values [16], which attempt to quantify the importance of features in making predictions. In [13], interpretability is explored using Shapley values for quantum models by quantifying the importance of each gate to a given prediction. The complexity of computing Shapley values for generalized quantum scenarios is analyzed in [17]. In [15], Shapley values are computed for a particular class of QNNs. Our work complements these efforts using an alternative notion of explainability to be discussed in detail next.

II. INTERPRETABILITY IN AI

A. Taxonomy

There are several layers to the design of interpretability techniques. To start, they can be *model-specific* or *model-agnostic*. As the name suggests, model-specific methods are more restrictive in terms of which models they can be used to explain. As such, they are designed to explain one single type of model. In contrast, model-agnostic methods allow for more flexibility in usage as they can be used on a wide range of model types. At large, model-agnostic methods can have a *global* or *local* interpretability dimension. Locality determines the scope of explanations with respect to the model. Interpretability at a global level explains the average predictions of the model as a whole. At the same time, local interpretability gives explanations at the level of each sample. In another axis, these techniques can be *active* (inherently interpretable) or *passive* (post-hoc). The state of these interpretable paradigms implies the level of involvement of interpretable techniques in the outcome of the other parameters. Active techniques change the structure of the model itself by leaning towards making it more interpretable. In contrast, passive methods explain the model outcome once the training has finished. In comparison to model-agnostic methods, which work with samples at large, there also exist example-based explanations that explain selected data samples from a dataset. An example of this method is the k-nearest neighbors models, which average the outcome of k nearest selected points.

Other than the idea of building interpretable techniques, or more precisely, techniques that interpret various models, there exist models that are inherently interpretable. Such models include linear regression, logistic regression, naive Bayes classifiers, decision trees, and more. This feature makes them good candidates as surrogate models for interpretability. Based on this paradigm, the concept of surrogate models exists, which uses interpretable techniques as a building block for designing other interpretable methods. Such important techniques are, for example, local interpretable model-agnostic explanations (LIME) [18] and Shapley additive explanations (known as SHAP) [16].

B. Interpretability of neural networks

The interpretability of neural networks remains a challenge on its own. This tends to amplify in complex models with many layers and parameters. Nevertheless, there is active research in the field and several proposed interpretable techniques [7]. Such techniques that aim to gain insights into the decision-making of a neural network include saliency maps [19], feature visualization [20, 21], perturbation or occlusion-based methods [16, 18], and layerwise relevance propagation (also known by its acronym LRP) [22].

To expand further on the above-mentioned techniques, saliency maps use backpropagation and gradient information to identify the most influential regions contributing to the output result. This technique is also called pixel attribution [4]. Feature visualization, particularly useful for convolutional natural networks, is a technique that analyses the importance of particular features in a dataset by visualizing the patterns that activate the output. In the same remark, in terms of network visualizations, Ref. [23] goes deeper into the layers of a convolutional neural network to gain an understanding of the features. This result, in particular, shows the intricacies and the rather intuitive process involved in the decision-making procedure of a network as it goes through deeper layers. Occlusion-based methods aim to perturb or manipulate certain parts of the data samples to observe how the explanations change. These methods are important in highlighting deeper issues in neural networks. Similarly, layerwise relevance propagation techniques reassign importance weight to the input data by analyzing the output. This helps the understanding by providing a hypothesis over the output decision. Finally, the class of surrogate-based methods mentioned above is certainly applicable in neural networks as well.

The importance of these techniques is also beyond the interpretability measures for human understanding. They can also be seen as methods of debugging and thus improving the result of a neural network as in Ref. [23]. Below, we take a closer look at surrogate model-agnostic local interpretable techniques, which are applicable to DNNs as well.

C. Local interpretable methods

Local interpretable methods tend to focus on individual data samples of interest. One of these methods relies on explaining a black-box model using inherently interpretable models, also known as surrogate methods. These methods act as a bridge between the two model types. The prototype of these techniques is the so-called local interpretable model-agnostic explanations (LIME), which has received much attention since its invention in 2016 [18]. Local surrogate methods work by training an interpretable surrogate model that approximates the result of the black-box model to be explained. LIME, for instance, is categorized as a perturbation-based technique that perturbs the input dataset. Locality in LIME refers to the fact that the surrogate model is trained on the data point of interest, as opposed to the whole dataset (which would be the idea behind *global* surrogate methods). Eq. (1) represents the explanation ξ of a sample x via its two main terms, namely the term $L(f, g, \pi_x)$ representing the loss, which is the variable to be minimized, and $\Omega(g)$ which is the complexity measure, which encodes the degree of interpretability. Here f is the black-box model, g is the surrogate model, and π_x defines the region in data space local to x (See Alg. 1 pseudocode). In broader terms, LIME is a trade-off between interpretability and accuracy,

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}}\, L(f, g, \pi_x) + \Omega(g) \tag{1}$$

In the following, we make use of the concept of local surrogacy to understand the interpretability of quantum models using LIME as a starting point. Much like LIME, we develop a *framework* to provide explanations of black-box models in the quantum domain. The class of surrogate models, the locality measure, and the complexity measure are free parameters that must be specified and justified in each application of the framework.

Algorithm1 LIME Algorithm [18]

function LIME(f, x, D, K)
  for $i = 1$ to K **do**
    Sample z_i from D
    $g_i = f(z_i)$ ▷ g_i is the prediction of the classifier on the perturbed sample
  end for
  $\xi(x) = \operatorname{argmin}_{g \in G} L(f, g, \pi_x)$ ▷ L is a loss function, π_x is a locality measure to the data point, and G is the set of surrogate models
  return $\xi(x)$
end function
function Loss(f, g, π_x)
  $L(f, g, \pi_x) = \sum_{z_i \in D} \pi_x(z_i)[f(z_i) \neq g(z_i)] + \Omega(g)$ ▷ Penalize the difference between predictions
  return $L(f, g, \pi_x)$
end function
function Locality(π_x, z, x)
  $\pi_x(z) = \exp\left(-\frac{d(x,z)^2}{\sigma^2}\right)$ ▷ d is a distance metric, σ is a bandwidth parameter
  return $\pi_x(z)$
end function

III. THE CASE FOR QUANTUM AI INTERPRETABILITY

As mentioned in Section I, interpretability in the quantum literature in the context of machine learning can take different directions. We consider the case when data is classical and encoded into a quantum state, which is manipulated by a variational quantum circuit before outputting a classical decision via quantum measurement.

Our focus is on interpreting the classical output of the quantum model.

A quantum machine learning model f takes as input data x and first produces quantum data $|\psi(x)\rangle$. A trained quantum algorithm, such as the QNN, subsequently processes the quantum data and generates a classification decision based on the outcome of a quantum measurement. This is not conceptually different from a classical neural network beyond the fact that the weights and biases have been replaced by parameters of quantum processes, except for one crucial difference — quantum measurements are unavoidably probabilistic.

Probabilities, or quantities interpreted as such, often arise in conventional neural networks. However, these numbers are encoded in bits and directly accessible, so they are typically used to generate deterministic results (through thresholding, for example). Qubits, on the other hand, are not directly accessible. While procedures exist to reconstruct qubits from repeated measurements (generally called *tomography*), these are inefficient — defeating any purpose of encoding information into qubits in the first place. Hence, QML uniquely forces us to deal with uncertainty in interpreting its decisions.

In the case of probabilistic decisions, the notion of a *decision boundary* is undefined. A reasonable alternative might be to define the boundary as those locations in data space where the classification is purely random (probability $\frac{1}{2}$). A data point here is randomly assigned a label. For such a point, any *explanation* for its label in a particular realization of the random decision process would be arbitrary and prone to error. It would be more accurate to admit that the algorithm is *indecisive* at such a point. This rationale is equally valid for data points near such locations. Thus, we define the *region of indecision* as follows,

$$R = \left\{ x' \in X : \left| P(f(x') = 1) - \frac{1}{2} \right| < \epsilon \right\}, \tag{2}$$

where ϵ is a small positive constant representing a threshold of uncertainty tolerated in the classification decision. In a certain context, points situated within this rationale lack a discernible *rationale*, hence explanation, for their specific outcomes, relying instead on random circumstances.

At present, although certain data points possess labels that were randomly assigned, one may pose the question as to the underlying rationale for such assignments. In other words, even data points lying within the region of indecision demand an explanation. Next, we will show how the ideas of local interpretability can be extended to apply to the probabilistic or quantum settings. In doing so, we aim to provide a nuanced and comprehensive understanding of model predictions.

A. Quantum LIME: Probabilistic local interpretability

Here, we define Quantum LIME (or Q-LIME) as an extension of the classical counterpart to quantum models, namely quantum local interpretable model-agnostic explanations. A specialization of the definition of Q-LIME is the hybrid version in the CQ, such as in Fig. 1.

In the context of the LIME algorithms, the loss function is typically chosen to compare models and their potential surrogates on a per-sample basis. However, if the model's output is random, the loss function will also be a random variable. An obvious strategy would be to define loss via expectation:

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}}\, \mathbb{E}[L(f, g, \pi_x)] + \Omega(g). \tag{3}$$

Nevertheless, even under these circumstances, a definitive assertion regarding ξ as an explanation remains elusive. This stems from the recognition that its predictions are only capturing the average behavior of the underlying model's randomness. In fact, the label provided by ξ may be the opposite of that assigned to x by the model in any particular instance, which clearly does not capture a reasonable notion of *explanation*.

To mitigate this, we generalize the classical definition and call an *explanation* the distribution Ξ of trained surrogate models g obtained through the application of LIME to the random output of any probabilistic classifier. Note again that g is random, trained on synthetic local data with random labels assigned by the underlying model. Thus, the explanation inherits any randomness from the underlying model. It is not the case that the explanation provides an interpretation of the randomness *per se* — however, we can utilize the distribution of surrogate models to simplify the region of indecision, hence providing an interpretation of it.

The symbol Ξ represents the Q-LIME explanation, emphasizing the interpretability framework tailored for QNNs. Additionally, Alg. 2 refers to a Monte Carlo approximation method employed within the Q-LIME process. This algorithm plays a pivotal role in facilitating the computation of interpretable insights by utilizing a Monte Carlo sampling approach, thereby enhancing the efficiency of the interpretation process.

Algorithm2 Quantum LIME (Q-LIME)

function QLIME(U, x, D, K)
    $\Xi \leftarrow$ Empty list to store quantum surrogate models
    for $i = 1$ to K **do**
        $D_i \leftarrow$ Generate synthetic quantum data locally around x
        $\gamma_i \leftarrow$ LIME(U, x, D_i, K) ▷ Apply LIME to generate a quantum surrogate model (see Alg. 1)
        Add γ_i to Ξ
    end for
    return Ξ
end function

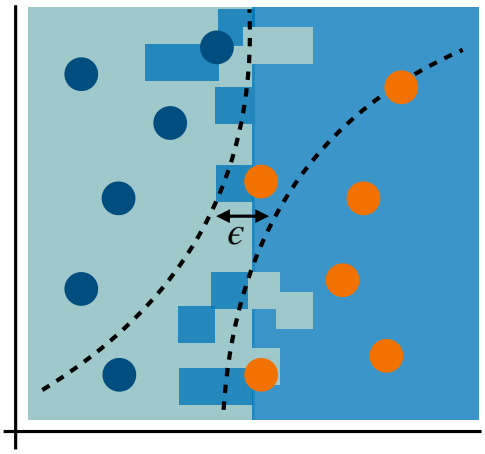


FIG. 2. **Depiction of the concept of the *local region of indecision*.** The space within the dashed lines represents the region in the decision space where data samples exhibit ambiguous classification due to randomness. The figure showcases a two-class classification task in a two-dimensional space with two features represented along the horizontal and vertical axes. Here, ϵ is the pre-defined threshold. Likely, data samples, either inside or close to the band, are interpreted as randomly assigned.

B. Local region of indecision

In this section, we define the *local region of indecision*. In a broad sense, this is the region of indecision interpreted locally through a distribution of surrogate models. Suppose a particular data point lies within its own local region of indecision. The explanation for its label is thus the data was given a random label, even though its explanation may not be a satisfying one. Moreover, this is a strong statement because — in principle — all possible interpretable surrogate models have been considered in the optimization.

The local region of indecision can be defined as the region of the input space where the classification decision of the quantum model is uncertain. More formally, we can define the local region of indecision B for a data point x in a dataset X as,

$$B = \left\{ x' \in X : \left| P(g(x') = 1 | f, \Xi, \pi_x) - \frac{1}{2} \right| < \epsilon \right\}, \quad (4)$$

where ϵ is again a small positive constant representing a threshold of uncertainty tolerated in the classification decision — this time with reference to the *explanation* rather than the underlying model.

The exact relationship between B and R is complicated, depending on the details of the loss function. However, for a $0-1$ loss function and π_x defining a closed boundary around x, $B = R$ within π_x. In a more general case, B approximates R up to the freedom that elements of Ξ are allowed to deviate from f. Note that the distribution in Eq. (4) is over Ξ as each g provides deterministic labels. This has been formalized in Alg. 3.

In much the same way that an interpretable model approximates decision boundaries locally in the classical context, a local region of indecision approximates the proper region of indecision in the quantum (or probabilistic) context. The size and shape of this region will depend on several factors, such as the choice of the interpretability technique, the complexity of the surrogate model, and the number of features in the dataset. We call the region a "band" as it describes the simplest schematic presented in Fig. 2.

In the examples presented in this paper, we chose the interquartile range (IQR) as a summary measure for the set of quantum surrogate models Ξ. However, other options are undoubtedly possible. It is important to note that in higher-dimensional spaces, the complexity of quantum models and the diversity of decision boundaries may necessitate alternative summary measures.

Algorithm3 Range

$B \leftarrow$ Empty list to store quantum decision boundaries
$\Xi \leftarrow \text{QLIME}(U, x, D, K)$ ▷ Generate quantum surrogate models using Quantum LIME (see Alg. 2)
for $i = 1$ to M **do** ▷ Repeat for M iterations
    $\xi \leftarrow$ Sample from Ξ
    $B_i \leftarrow \text{Boundary}(\xi)$ ▷ Obtain the quantum decision boundary from the sampled quantum surrogate model
    Add B_i to B
end for
$IQR \leftarrow \text{InterquartileRange}(B)$ ▷ Compute the interquartile range of quantum decision boundaries
return IQR
function InterquartileRange(B)
    Sort the quantum decision boundaries in B
    $Q_1 \leftarrow$ The median of the lower half of B
    $Q_3 \leftarrow$ The median of the upper half of B
    $IQR \leftarrow Q_3 - Q_1$ ▷ Interquartile range
    return IQR
end function

IV. NUMERICAL EXPERIMENTS FOR INTERPRETING QNNS

We use the well-known Iris dataset [24] for our numerical experiments. To enhance the explicability of our methodology, we opt to simplify the analysis by framing it as a binary classification problem. This entails the exclusive utilization of two out of the three available classes within the dataset, coupled with the restriction to employ only two out of the four available features. In consideration of our indifference toward classifying the two categories of flowers, we abstract the names of these classes and labels below.

The trained quantum model to be explained is a hybrid QNN trained using simultaneous perturbation stochastic approximation (better known by its acronym SPSA) [25],

which is an optimization algorithm that efficiently estimates the gradient of an objective function. This model has been built and simulated using the Qiskit framework [26]. Each data point is encoded into a quantum state with the angle encoding [27], which acts by mapping numerical values to specific angles, facilitating their integration into quantum models. The QNN model is an autoencoder with alternating layers of single qubit rotations and entangling gates [28]. Since our goal here is to illustrate the local region of indecision, as in Eq. 4, we do not optimize over the complexity of surrogate models and instead fix our search to the class of logistic regression models with two features.

The shaded background in each plot of Figs. 3 and 4 show the decision region of the trained QNN. Upon inspection, it is clear that these decision regions, and the implied boundary, change with each execution of the QNN. In other words, the decision boundary is ill-defined. In Fig. 3, we naively apply the LIME methodology to two data points — one in the ambiguous region and one deep within the region corresponding to one of the labels. In the latter case, the output of the QNN is nearly deterministic in the local neighborhood of the chosen data point, and LIME works as expected — it produces a consistent explanation.

Nonetheless, in the first example, the data point receives a random label. It is clearly within the *region of indecision* for a reasonably small choice of ϵ. The "explanation" provided by LIME (summarized by its decision boundary shown as the solid line in Fig. 3) is random. In other words, each application of LIME will produce a different explanation. For the chosen data point, the explanation itself produces opposite interpretations roughly half the time, and the predictions it makes are counter to the actual label provided by the QNN model to be explained roughly half the time. Clearly, this is an inappropriate situation to be applying such interpretability techniques. Heuristically, if a data point lies near the decision boundary of the surrogate model for QNN, we should not expect that it provides a satisfactory explanation for its label. Q-LIME and the *local region of indecision* rectify this.

For the same sample data points, the local region of indecision is shown in Fig. 4. Data points within their own band should be regarded as having been classified due purely to chance — and hence have no "explanation" in the conventional sense for the label they have been assigned. Note that the band itself is unlikely to yield an analytic form. Hence, some numerical approximation is required to calculate it in practice. Our approach, conceptually, was to repeat what was done to produce Fig. 3 many times and summarize the statistics to produce Fig. 4. A more detailed description follows, and the implementation to reproduce the results presented here can be found at [29].

Approximating the local region of indecision requires a trained QNN, a sample data point, a function defining "local", and some meta-parameters defining the quality

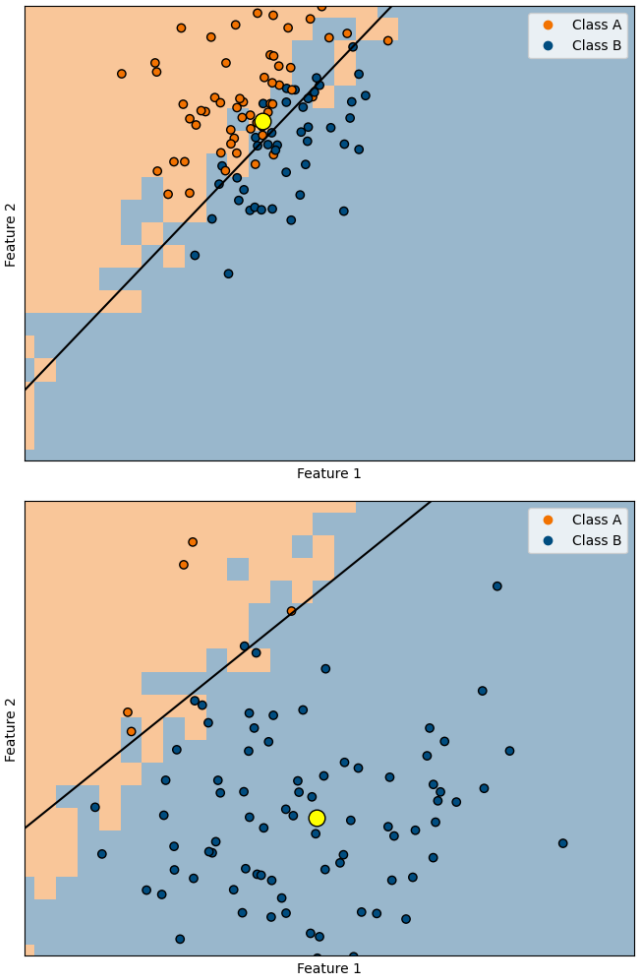


FIG. 3. **The classification uncertainty for a chosen data sample (yellow)**. The two shadings represent the decision boundary of the QNN, which is clearly randomly defined. The synthetic data and the produced surrogate linear decision boundary are shown for a single instance of QNN labels. (top) A clear example of an undefined classification result for the selected data point. (bottom) Decision boundary unambiguously separates the who classes with respect to the selected data point.

of the numerical approximation, such as the amount of synthetic data to be used and the number of Monte Carlo samples to take. Once these are defined, the general procedure is as follows, which is formalized in Alg. 2 and Alg. 3.

1. Create synthetic data locally around the chosen data point;
2. Apply LIME to generate a surrogate model;
3. Repeat 1 and 2 several times to produce an ensemble of decision boundaries.
4. Take the interquartile range (for example) of those boundaries.

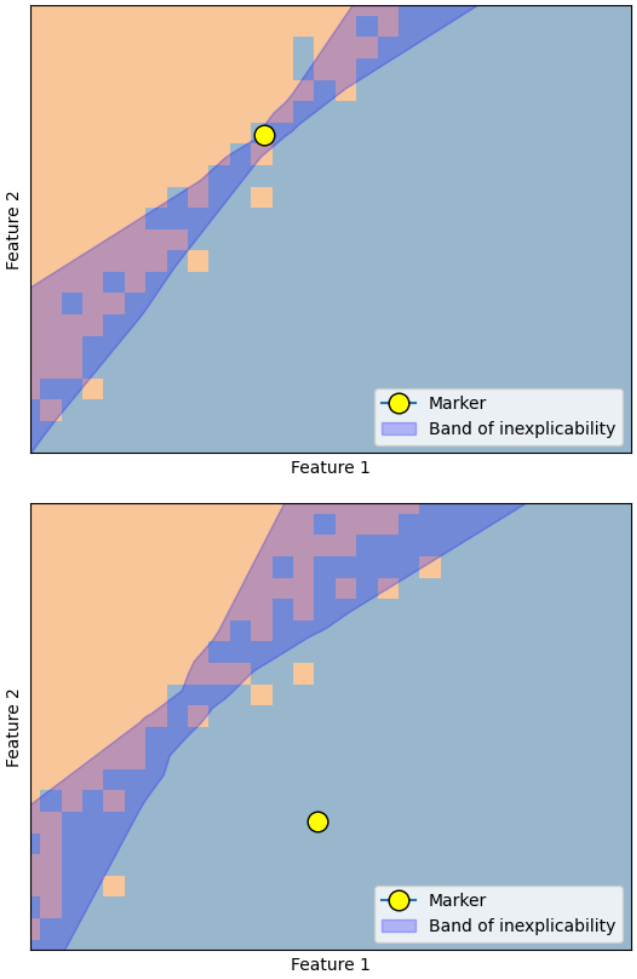


FIG. 4. **The approximated local region of indecision.** (top) An example of a marked data point that lies on the local region of indecision. (bottom) A data point that is outside of this region can be assessed for interpretability as per the interpretable techniques.

To make evident, Fig. 3 is produced through a single application of steps 1 and 2 while Fig. 4 is produced after the final step. It is important to recognize that, in a practical or real-world application, it would be infeasible to plot the entire decision regions to any reasonable level of accuracy. The computational demands and complexities involved in plotting such extensive regions can be prohibitive. This makes the region of indecision both valuable and convenient in interpreting QNNs.

V. DISCUSSION & FUTURE WORK

In summary, in interpreting the classification output of a QNN, we have defined Q-LIME and the local region of indecision, a simple description of a region where the QNN is interpreted to be indecisive. The data samples within their associated band should not be expected to have an "explanation" in the deterministic classical sense. While directly useful for hybrid quantum-classical models, we hope this stimulates further research on the fundamental differences between quantum and classical model interpretability. In the remainder of the paper, we discuss possible future research directions.

Our results are pointed squarely at the randomness of quantum measurements, which might suggest that they are "backward compatible" with classical models. Indeed, randomness is present in training classical DNNs due to random weight initialization or the optimization techniques used. However, this type of randomness can be "controlled" via the concept of a *seed*. Moreover, the penultimate output of DNNs (just before label assignment) is often a probability distribution. Nevertheless, these are produced through arbitrary choices of the activation function (i.e., the Softmax function), which force the output of a vector that merely *resembles* a probability distribution. Each case of randomness in classical DNNs is very different from the innate and unavoidable randomness of quantum models. While our techniques *could* be applied in a classical setting, the conclusions drawn from them may ironically be more complicated to justifiably action.

In this work, we have provided concrete definitions for QNNs applied to binary classification problems. Using a probability of $\frac{1}{2}$ would not be a suitable reference point in multi-class decision problems. There are many avenues to generalize our definitions, which would mirror standard generalizations in the move from binary to multi-class decision problems. One such example would be defining the region of indecision as that with nearly maximum entropy.

We took as an act of brevity the omission of the word *local* in many places where it may play a pivotal role. For example, the strongest conclusion our technique can reach is that a QNN is merely *locally* inexplicable in the classical sense. In such cases, we could concede that (for some regions of data space) the behavior of QNN is inherently stochastic, lacking any discernible patterns or deterministic explanations. Alternatively, we can use this conclusion to signal that an explanation at a higher level of abstraction is required. This shift toward higher-level abstraction becomes imperative for gaining more insights into QNNs, acknowledging their inherent quantum nature and the limitations imposed by classical interpretability paradigms.

Classically, should a data point inquire, "Why has this label been assigned to me?", within a quantum framework, an initial response may invoke the inherent stochastic nature of quantum systems. Nonetheless, persistent interrogation from the data point could necessitate a quantum extension of *global* interpretability methodologies to clarify the specific features or attributes contributing to the assigned label.

Referring back to Fig. 1, we have focused here on CQ quantum machine learning models. Nevertheless, the core idea behind local surrogate models remains applica-

ble in the context of quantum data — use interpretable *quantum* models as surrogate models to explain black box models producing quantum data. Of course, one of our assumptions is the parallels between the classical interpretable models we mentioned above, with their quantum equivalents. This can be a line for future work. The ideas here encapsulate inherently quantum models such as matrix product states or tensor network states, which can act as surrogate models for quantum models as they may be considered more interpretable. These surrogate models can serve as interpretable proxies, facilitating a comprehension of the underlying quantum processes, thus paving the way for enhanced interpretability of quantum models.

Furthermore, the idea behind *interpreting* or "opening up" black-box models may be of interest in control theory [30–32]. In this scenario, the concept of "grey-box" models — portions of which encode specific physical models — give insights into how to engineer certain parameters in a system. These grey-box models can thus be considered *partially explainable* models. The proposed algorithm in [33] may also be of interest in terms of creating intrinsically quantum interpretable models, which would act as surrogates for other more complex quantum models.

Advanced ideas for future work entail exploring and developing specific metrics tailored for assessing interpretability in QML models. This may involve defining measures that capture the degree of coherence or entanglement contributing to model predictions.

An obvious open question that inspires future research remains to investigate the difference in computational tractability of interpretability methods in quantum versus classical. This will lead to understanding whether it is more difficult to interpret quantum models as opposed to classical models. We hope such results shed light on more philosophical questions as well, such as *is inexplicability, viz. complexity, necessary for learning?*. This exploration holds the potential to advance our understanding not only of the interpretability challenges in quantum models, but also of the fundamental relationship between complexity and the learning process itself.

For completeness, the case for the interpretability of machine learning models does not go without critique. Certain viewpoints contend that the pursuit of insights into a model's decision-making process should not come at the expense of sacrificing performance, and, in practical terms, this prioritization might not always be feasible or deemed necessary [34, 35]. It is often posited that simpler models inherently possess a greater degree of explainability. Nonetheless, it is the more complex models that require explanations, as they may be more likely employed in critical applications.

Regardless of the two distinct camps of beliefs, the niche field of interpretable machine learning keeps growing in volume. An argument is that having a more complete picture of the model's performance can help improve the performance of the model overall. As QML becomes increasingly relevant to AI research, we expect the need for quantum interpretability will also be in demand. This research contributes to a broader understanding of explicability in quantum AI models, paving the way for the development of accountable and transparent systems in the quantum computing and AI era.

Acknowledgments: LP was supported by the Sydney Quantum Academy, Sydney, NSW, Australia.

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. (Prentice Hall, 2010).
[2] T. M. Mitchell, *Machine Learning*, 1st ed. (McGraw-Hill, Inc., USA, 1997).
[3] C. Molnar, *Interpretable Machine Learning*, 2nd ed. (2022).
[4] C. Molnar, G. Casalicchio, and B. Bischl, Interpretable machine learning – a brief history, state-of-the-art and challenges, in *ECML PKDD 2020 Workshops* (Springer International Publishing, 2020) pp. 417–431.
[5] Z. C. Lipton, The mythos of model interpretability, Queue **16**, 31 (2018).
[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016) http://www.deeplearningbook.org.
[7] Y. Zhang, P. Tino, A. Leonardis, and K. Tang, A survey on neural network interpretability, IEEE Transactions on Emerging Topics in Computational Intelligence **5**, 726 (2021).
[8] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, A survey of methods for explaining black box models, ACM computing surveys (CSUR) **51**, 1 (2018).
[9] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum Machine Learning, Nature **549**, 195 (2016).
[10] M. Schuld and F. Petruccione, *Machine Learning with Quantum Computers* (Springer International Publishing, 2021).
[11] M. Cerezo, G. Verdon, H.-Y. Huang, L. Cincio, and P. Coles, Challenges and opportunities in quantum machine learning, Nature Computational Science (2022).
[12] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, Variational quantum algorithms, Nature Reviews Physics **3**, 625 (2021).
[13] R. Heese, T. Gerlach, S. Mücke, S. Müller, M. Jakobs, and N. Piatkowski, Explaining quantum circuits with shapley values: Towards explainable quantum machine learning, arXiv preprint arXiv:2301.09138 10.48550/arXiv.2301.09138 (2023).
[14] F. Mercaldo, G. Ciaramella, G. Iadarola, M. Storto, F. Martinelli, and A. Santone, Towards explainable quantum machine learning for mobile malware detection and classification, Applied Sciences **12**, 10.3390/app122312025 (2022).

[15] P. Steinmüller, T. Schulz, F. Graf, and D. Herr, explainable ai for quantum machine learning, arXiv preprint arXiv:2211.01441 10.48550/arXiv.2211.01441 (2022).

[16] S. M. Lundberg and S.-I. Lee, A unified approach to interpreting model predictions, in *Advances in Neural Information Processing Systems*, Vol. 30 (Curran Associates, Inc., 2017).

[17] I. Burge, M. Barbeau, and J. Garcia-Alfaro, A quantum algorithm for shapley value estimation, arXiv preprint arXiv:2301.04727 10.48550/arXiv.2301.04727 (2023).

[18] M. T. Ribeiro, S. Singh, and C. Guestrin, "why should i trust you?": Explaining the predictions of any classifier, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16 (Association for Computing Machinery, New York, NY, USA, 2016) p. 1135–1144.

[19] K. Simonyan, A. Vedaldi, and A. Zisserman, Deep inside convolutional networks: Visualising image classification models and saliency maps, arXiv preprint arXiv:1312.6034 10.48550/arXiv.1312.6034 (2013).

[20] C. Olah, A. Mordvintsev, and M. Tyka, Feature visualization: How neural networks build up their understanding of images, Distill **2** (2017).

[21] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, Understanding neural networks through deep visualization, in *Proceedings of the 32nd International Conference on Machine Learning (ICML)* (2015) pp. 1582–1591.

[22] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation, PLoS ONE **10**, 10(7): e0130140 (2015).

[23] M. D. Zeiler and R. Fergus, Visualizing and understanding convolutional networks, European conference on computer vision , 818 (2014).

[24] R. A. Fisher, The use of multiple measurements in taxonomic problems (1936).

[25] J. C. Spall, An overview of the simultaneous perturbation method for efficient optimization, Johns Hopkins APL Technical Digest **19**, 482 (1998).

[26] IBM, Qiskit: An Open-Source Framework for Quantum Computing (2021), accessed on May 1, 2023.

[27] M. Schuld and F. Petruccione, *Supervised Learning with Quantum Computers*, Quantum Science and Technology (Springer International Publishing, 2018).

[28] E. Farhi and H. Neven, Classification with quantum neural networks on near term processors, arXiv preprint arXiv:1802.06002 10.48550/arXiv.1802.06002 (2018).

[29] L. Pira and C. Ferrie, Interpret QNN: Explicability and Inexplicability in the Interpretation of Quantum Neural Networks, `https://github.com/lirandepira/interpret-qnn` (2023), accessed on July 24, 2023.

[30] A. Youssry, Y. Yang, R. J. Chapman, B. Haylock, F. Lenzini, M. Lobino, and A. Peruzzo, Experimental graybox quantum system identification and control, arXiv preprint arXiv:2206.12201 10.48550/arXiv.2206.12201 (2023).

[31] A. Youssry, G. A. Paz-Silva, and C. Ferrie, Characterization and control of open quantum systems beyond quantum noise spectroscopy, npj Quantum Information **6**, 95 (2020).

[32] A. Youssry, G. A. Paz-Silva, and C. Ferrie, Noise detection with spectator qubits and quantum feature engineering, New Journal of Physics **25**, 073004 (2023).

[33] G. Weitz, L. Pira, C. Ferrie, and J. Combes, Sub-universal variational circuits for combinatorial optimization problems (2023).

[34] A. Sarkar, Is explainable AI a race against model complexity?, arXiv preprint arXiv:2205.10119 10.48550/arXiv.2205.10119 (2022).

[35] L. McCoy, C. Brenna, S. Chen, K. Vold, and S. Das, Believing in Black Boxes: Must Machine Learning in Healthcare Be Explainable to Be Evidence-Based?, Journal of Clinical Epidemiology 10.1016/j.jclinepi.2021.11.001 (2022).

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 258 (2025) 3723–3730

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Machine Learning and Data Engineering

Explainable Artificial Intelligence for Computer Vision and Quantum Machine Learning

Asharul Islam Khan [a]*, Ali Al Badi [b], Mohammed Alqahtani[c]

[a] *OCCI, HURC, Sultan Qaboos University, P.O.Box 33 AlKhodh, Muscat, P.C.123, Sultanate of Oman*
[b]*Dean, Gulf College, P.O.Box 885, Al Mabaila, Muscat, P.C.133, Sultanate of Oman*
[c]*Department of Computer Information Systems, Imam Abdulrahman Bin Faisal University, P.O. Box 1982, Dammam 31441, Saudi Arabia*

Abstract

Artificial Intelligence and Machine Learning (AI/ML) are the emulation of human intelligence by computer systems. The AI/ML models have made inroads into Computer vision and Quantum computing due to their tremendous ability for reasoning and problem-solving. Computer vision allows computers to interpret visual information (photos, videos) similar to humans while Quantum computers execute complex tasks more efficiently than regular computers. However, how to explain the reasons for specific outputs is an important challenge with AI/ML models when applied to Computer vision and Quantum computing. Without the explainability of the AI/ML, the reliability and validity of Computer vision and Quantum computing outputs, when AI/ML is applied to them, is of great concern for researchers. To solve the issue of trustworthiness and transparency researchers have come up with the idea of explainable AI/ML (XAI). Therefore, to understand the extent of research on XAI in the area of Computer vision and Quantum computing, this study used the existing literature from the Scopus database and statistically presented the results. The study has found that the research on XAI in Computer vision and Quantum computing has been growing particularly since 2019. For AI/ML in Computer vision, the maximum studies on the XAI are in the healthcare sector accounting for 57% followed by autonomous vehicles. As far as AI/ML in Quantum computing is concerned, the majority of XAI research is related to the healthcare sector accounting for 46% followed by cyber security (20%).

Keywords: Explainable Artificial Intelligence; explainable Machine learning; XAI; explainable healthcare; computer vision; Quantum computing

* Corresponding author.
E-mail address: a.khan2@squ.edu.om/ashar.367@gmail.com

1877-0509
10.1016/j.procs.2025.04.627

1. Introduction

AI/ML is a new industrial revolution. According to the Statista forecast by 2030, AI/ML will have a market volume of US$503.40bn while the market volume of Computer vision will be US$46.96bn by 2030 moving at an annual growth rate (CAGR 2024-2030) of 10.50%[†]. Computer vision uses AI/ML techniques and algorithms to detect and classify objects in multimedia files. Nowadays researchers are leveraging the benefits of Quantum computing to greatly accelerate classical machine learning tasks. Quantum machine learning is executed on quantum computers. It takes advantage of quantum computers' information processing efficiency [1]. Quantum computing uses counterintuitive quantum physics principles relying on quantum bits and qbits rather than 0 or 1 used in traditional computing.

Unfortunately, one of the challenges with AI/ML is the black-box nature of its algorithms. There is no explanation of what happens inside AI/ML classifiers like Recurrent Neural Network (RNN), Deep Neural Network (DNN), and the user community often suspects the outputs. As a result, many AI/ML-based outputs and judgments are difficult to understand by both researchers and common consumers. In safety-critical systems such as autonomous driving, the AI/ML model interpretability is important [2]. The AI/ML model for stock price prediction has limitations of generalization [3]. Even the Convolutional Neural Network (CNN), used in rescue robotics is limited by explainability [4]. Similarly, Computer vision in medical imaging has issues of explainability [5]. Likewise, despite Quantum Machine Learning's popularity, quantum neural networks (QNN) are highly counterintuitive and difficult to understand due to their architecture's unique quantum-specific layers (for example, data encoding and measurement). It prohibits QNN users and researchers from fully comprehending its inner workings and investigating the model's training status [6].

To solve the issue of expansibility in Computer vision and Quantum Machine Learning, the researchers have come up with the idea of XAI. The XAI aims to improve trustworthiness, transparency, and confidence in the AI/ML outputs. XAI can provide reliable answers to questions regarding how data is evaluated and what criteria are considered during the decision-making process. Hence, this study aims to investigate the existing research and contributions made on the subject of XAI in the area of Computer vision and Machine Learning in Quantum computing besides suggesting relevant research directions. This study will encourage scholars and practitioners to devote financial and non-financial resources to the nascent field of XAI in the area of Computer vision and Quantum computing. There are five sections in this paper. Section 2 describes the basics of Computer vision and Quantum computing, and XAI. Section 3 presents the methods and findings. Section 4 presents the discussion. The last section summarizes the study.

2. Background

This section describes the basics of AI/ML, Computer vision, and Quantum computing. In today's age tons of multimedia information are available. The indexing and information extraction from such huge data sources requires AI/ML-based computing interventions. With the help of Computer vision, one can detect, identify, and extract relevant information from multimedia files. Computer vision uses AI/ML techniques either supervised, unsupervised, or semi-supervised to render the information from the multimedia files. The biggest advantages of such algorithms and their models are in classification mainly in healthcare such as if a disease is benign or malignant. Medical imaging is an emerging area with a number of applications in scientific and technological fields, including health sciences and engineering. The unsupervised algorithm looks for patterns in the input data rather than depending on the input labels. For instance, Generative Adversarial Networks (GAN) and Auto Encoders. The reinforcement algorithms depend on a reward function where neural network weight is optimized for the sake of maximizing the reward function. Computer vision is useful for object identification, classification, and extracting meaningful information from photos, graphic documents, and videos. For instance, figure 1 shows the Computer vision implementation in image processing.

[†] https://www.statista.com/outlook/tmo/artificial-intelligence/computer-vision/worldwide

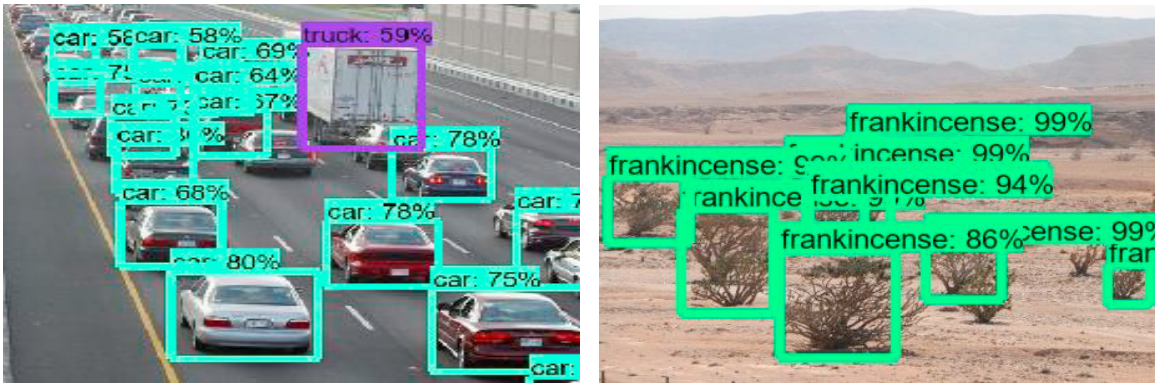


Figure 1. Computer vision implementation in image processing (Deep learning and Faster-RCNN-Inception-V2 model)

Quantum computers' huge computational capacity and speed have the potential to create a quantum leap in various fields. Quantum machine learning is now gaining popularity in a variety of fields because of its better performance and capabilities. However, Quantum machine learning requires comprehension by users which is a complex and challenging task [7] due to the black-box nature of AI/ML-based techniques. The AI/ML models created with AI/ML algorithms such as Support Vector Machines (SVM), Random Forests, and K-Nearest Neighbors (KNN), as well as Deep learning algorithms such as Artificial Neural Networks, CNN, and RNN, are difficult to interpret, even for an expert in AI/ML field. Therefore, the question arises to what extent the research in the XAI field has progressed and what segments of the Computer vision and Quantum computing sector it has impacted.

3. Method and Findings

This study has used a review of the existing literature from Scopus-indexed literature focused on XAI in Computer vision and Quantum computing to investigate the extent of the research in the field. Scopus is one of the most popular and largest databases. As a result, the research team conducted a study using the Scopus database. A number of keywords were used. They are "*explainable*", "*Machine learning*", "*XAI*", "*interpretable*", "quantum", and "*commuter vision*", and "*Artificial Intelligence.*" The keyword search yielded a total of 35 relevant publications for analysis. Moreover, statistical analysis was conducted on the collected information to make it presentable form.

The study has found that XAI in Computer vision and Quantum computing has been growing since 2019. The majority of the publications came in 2022, approximately 90%. Figure 2 shows the explainable AI/ML studies in Computer vision and Quantum computing.

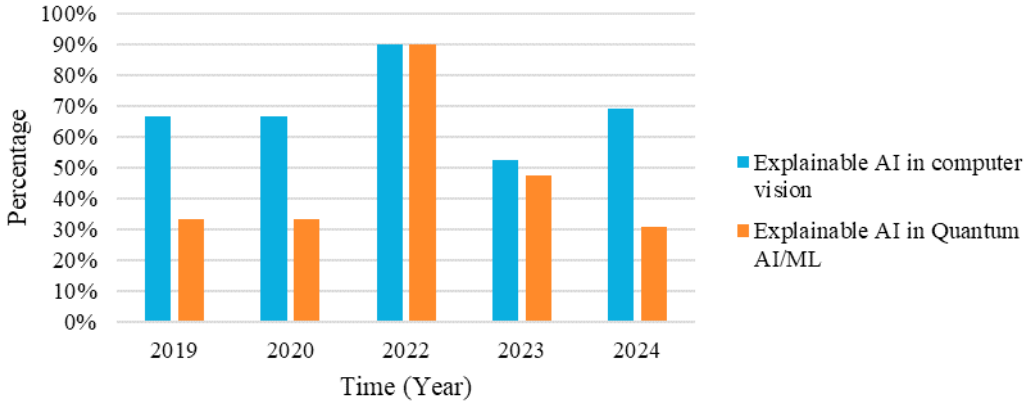


Figure 2. Explainable AI/ML studies in the area of Computer vision and Quantum computing

3.1. Explainable AI/ML in Computer vision and Quantum computing

The healthcare sector (57%) has received the maximum attention from researchers for explainable AI/ML in Computer vision followed by autonomous vehicles (22%). The application of AI/ML techniques to medical image processing has transformed the field of healthcare. However, classical AI models' lack of interpretability and transparency has hampered their broad use in clinical practice. XAI addresses this issue by offering interpretable explanations for AI/ML-generated conclusions. In the self-driving vehicles, the Computer vision plays critical role helping in the object detection and recognition on the streets. However, the AI/ML algorithms should be explainable in terms of their actions [8]. Figure 3 shows the explainable AI/ML in Computer vision.

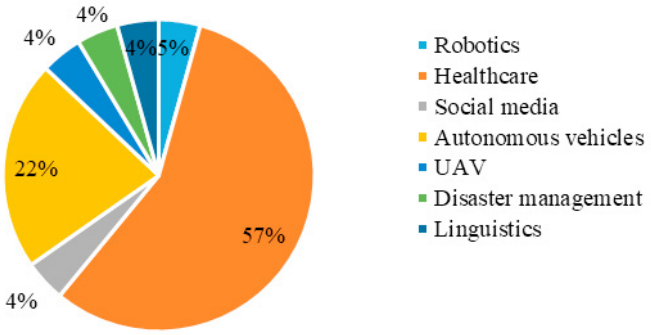


Figure 3. Explainable AI/ML in Computer vision

The healthcare sector (46%) has received the maximum attention of researchers for explainable AI/ML in Quantum computing followed by cyber security (20%). The outcomes of XAI models in medical image processing allow healthcare workers to comprehend reasons for AI/ML model outputs, resulting in increased diagnostic accuracy, and trust in AI/ML systems. Figure 4 shows the explainable AI/ML in Quantum computing.

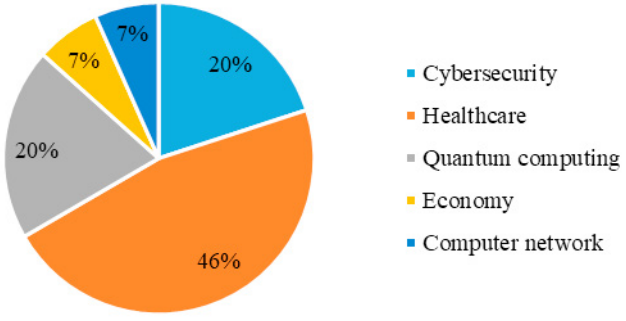


Figure 4. Explainable AI/ML in Quantum computing

There are various research themes related to explainable AI/ML in Computer vision and Quantum computing. For instance, robots for search and rescue, face expression classification, object detection, etc. However, the majority corresponds to Malware detection (15.4%) and image processing (38%). Table 1 shows XAI research themes in Computer vision and Quantum computing.

Table 1. XAI research themes in Computer vision and Quantum computing

XAI research in Quantum AI/ML with key themes	Percentage	XAI research in Computer vision with key themes	Percentage
Breast cancer images	7.7%	Robot for search and rescue	3.4 %
COVID-19 detection and classification	7.7%	Face expression classification	13.8 %
False information detection	7.7%	Image processing	34.5 %
Malware detection	15.4%	Object detection	37.9 %
MRI-Radiomic Quantum Neural Network	7.7%	Review	3.4 %
Predict the risk of heart disease	7.7%	Alzheimer's disease	3.4 %
Quantum clustering method	7.7%	Glaucoma diagnosis	3.4 %
Quantum neural networks	7.7%		
Signal modulation classification	7.7%		
Small quantum states	7.7%		
Stock price prediction	7.7%		
Wearable electronic optical data analysis	7.7%		

3.2. Explainable AI/ML approaches

There are different XAI approaches developed by researchers such as Intrinsic, Post-hoc, Local, Global, etc. [9, 10]. Posthoc looks into "what went wrong" in a black box environment [11]. Model agnostic explainable approaches are appropriate for clinical application [12]. The four important XAI approaches are explained below.

Local and Global Explanation: The local approaches explain a single data point or forecast while the Global method interprets either a number of data points or the complete model [12]. Global approaches seek to describe in detail the model's overall performance.

Model-specific and Model-agnostic: Model-specific methods function with only one data type or specific model types (e.g., SHAP) but Model-agnostic techniques work with any data type. The Model agnostic approach is based on the principle of looking for change in the output as a consequence of input changes [12].

Intrinsic and Post-hoc: Intrinsic has a structure that is simple enough and inherently interpretable. In such a case linear model or the splits learned with a Decision Tree can be used to deduce why a model makes the predictions it does. Post-hoc explanation strategies, on the other hand, provide explanations after the classification has been formed.

Attribution and Non-attribution: In order to determine which of the neural network's inputs are the most relevant in terms of the network's output, attribution methods are used. Here the important neural network characteristics are determined on the basis of class score when a feature is removed [12].

3.3. Explainable AI/ML frameworks

The working of XAI includes a number of elements. Figure 5 shows a simple illustration of explainable XAI working.

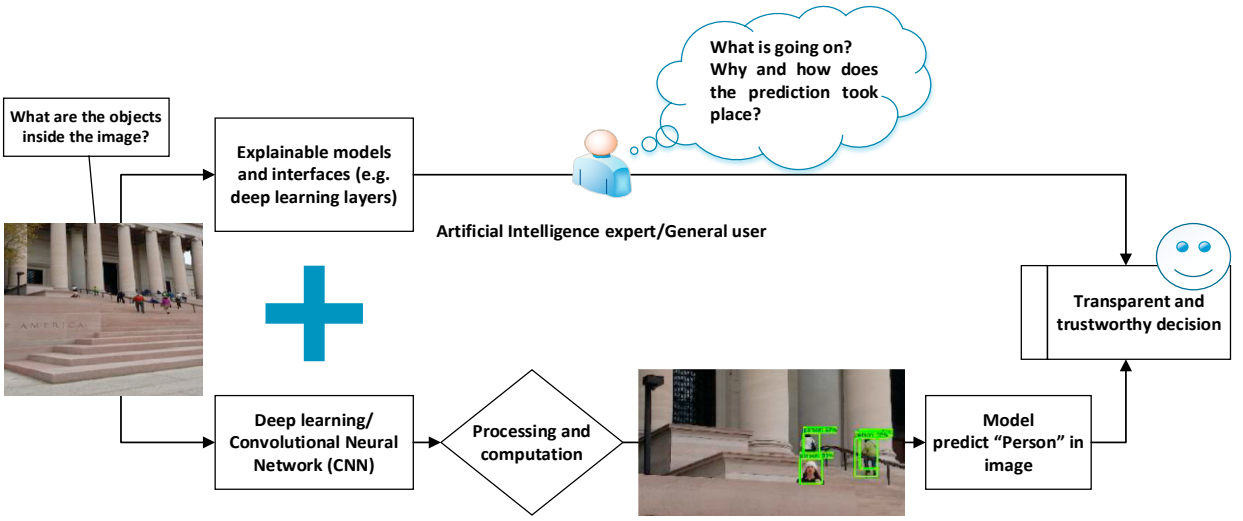


Figure 5. Simple illustration of XAI working (Authors own work)

There are a variety of frameworks and techniques used in the XAI. The Accumulated local effects (ALE), Contrastive Explanation Method (CEM), Partial Dependence Plot (PDP or PD plot), SHapley Additive exPlanations (SHAP), Local interpretable model-agnostic explanations (LIME), and Global Interpretation with Recursive Partitioning (GIRP) are the XAI frameworks. The prominent ones are listed in the table2.

Table 2. Frameworks and techniques used in the XAI

XAI frameworks	Description	Global/ Local
SHAP [13]	SHAP is based on Shapley values, which are commonly used for optimal credit allocation. It is a paradigm for explaining the outcome of any black-box model. However, it is more efficient on certain model types (such as tree ensembles)	Global and Local
LIME [14]	This framework generates or automates interpretable models for the AI/ML prediction models.	Local
PDP [15]	This framework measures the small changes in the outputs of AI/ML models as a consequence of one or two factors.	Global
ALE [16]	This technique works on classification and regression models for measuring the effects of features.	Global
CEM [17]	CEM interprets the classification model outputs	Local
GIRP [18]	GIRP works on the most important decision rules of the AI/ML model for interpretations	Global

4. Discussion

Gesture and facial identification, recognition of handwritten letters and digits, sophisticated driver assistance systems, and behavioral investigations are some of the applications of Computer vision [19]. In healthcare, it is used in medicine, surgery, and diagnostics. In the industry it is used to perform predictive maintenance, therefore helping the industries to replace parts and machines before breakdowns leading to efficiency and efficiency of producing units. Object identification models such as DNN used in UAVs or self-driving cars suffer from a lack of transparency [20]. This could lead to safety issues in many instances.

People find it challenging to understand how these algorithms get their findings. It is not only necessary to examine the explanation, but also to assess its quality and relevance to the given setting. Even professionals in the fields find it challenging to grasp the outcomes of AI/ML models since they are black boxes. It is critical to balance AI/ML accuracy and interpretability because the best-performing systems, such as Deep Learning, are typically the

least transparent in explanation, as opposed to Decision Trees. Such a difficult scenario demands an extra layer of interpretation. XAI solves these problems by incorporating extra explanations into the AI/ML outputs [21].

Object detection can be used to avoid traffic collisions, recognize facial expressions, and identify emotions based on human postures. In applications like Face Verification, it is critical to ensure decision transparency, justice, and accountability. Therefore, more research on XAI is needed [22].

The rise of next-generation AI/ML has recently brought responsible, ethical, and trustworthy decision-making to the forefront as one of society's most urgent concerns [23]. The laws and regulations in many countries require an explanation of the black box algorithms of AI, besides the ethical requirements. For example, regulations such as Canada's Personal Information Protection and Electronic Documents Act (PIPEDA), the USA's Health Insurance Portability and Accountability Act (HIPAA), and the European Union's General Data Protection Regulation (GDPR) have been established to address privacy concerns [24]. Additionally, the Defense Advanced Research Projects Agency (DARPA) has shown interest in explainable AI (XAI) and has developed the DARPA-XAI program [24].

In the end, the XAI paradigm emphasizes justice and accountability. XAI is not only beneficial and critical in the healthcare environment, but it also opens a wide range of possibilities for AI/ML solutions overall. As a result, the opacity of AI/ML, which has been widely criticized, can be decreased, and critical confidence can be developed. This is exactly what will boost future customer acceptance over time.

5. Conclusion

The AI/ML has already emerged as one of the century's most crucial technologies. However, questions regarding its safe and dependable use are increasing. For example, self-driving automobiles that use Computer vision have long been a subject of contention in advanced countries such as the United States. The same might be said for whether Computer vision can or should help with, or even make, healthcare decisions. Despite significant breakthroughs, the lack of transparency and interpretation abilities in AI/ML-based systems has remained a major barrier to adoption. This has sparked a new discussion about the value of AI/ML in the lifesaving and vital decision-making systems mainly utilizing technologies such as Computer vision and Quantum computing. This study found that the XAI in Computer vision and Quantum computing is an emerging area of research. The interpretation and explanations of the outputs of AI/ML in Computer vision and Quantum computing are moving forward with the development of new processes, models, and methodologies. Much of the research in the XAI area is in the healthcare sector and cybersecurity. The XAI is essential in healthcare due to its credibility, explanation, application, dependability, equality, openness, interaction, and greater sense of privacy. However, other areas also need attention including disaster management and robotics. One of the major issues is that XAI techniques are designed for general usage without considering the context of the difficulties. The findings can serve as a foundation for future research developments and encourage experts and professionals from various fields to embrace the benefits of explainable AI (XAI) within their industries.

References

[1] Schuld, Maria, and Petruccione, Francesco. (2021)" *Machine learning with quantum computers."* Springer. **676** (2021)

[2] Gierse, Daniel, Neubürger, Felix, and Kopinski, Thomas. (2023). A Novel Architecture for Robust Explainable AI Approaches in Critical Object Detection Scenarios Based on Bayesian Neural Networks. In *World Conference on Explainable Artificial Intelligence*. Springer 126-147

[3] Gandhudi, M., P.J.A, A., Fiore, U., and G.R, G. (2024) "Explainable hybrid quantum neural networks for analyzing the influence of tweets on stock price prediction." *Computers and Electrical Engineering*. **118** (2024): 1-9

[4] Hönig, Peter, and Wöber, Wilfried. (2023). Explainable Object Detection in the Field of Search and Rescue Robotics. In *International Conference on Robotics in Alpe-Adria Danube Region*. Springer. 2023: 37-44

[5] Sohail, A., Fahmy, M. A., and Khan, U. A. (2023) "XAI hybrid multi-staged algorithm for routine & quantum boosted oncological medical imaging." *Computational Particle Mechanics*. **10** (2023): 209-219

[6] Ruan, S., Liang, Z., Guan, Q., Griffin, P., Wen, X., Lin, Y., and Wang, Y. (2024) "VIOLET: Visual Analytics for Explainable Quantum Neural Networks." *IEEE Transactions on Visualization and Computer Graphics.* **30**(2024): 2862-2874

[7] Frohnert, F., and van Nieuwenburg, E. (2024) "Explainable representation learning of small quantum states." *Machine Learning: Science and Technology*. ***5*** (2024)

[8] Nowak, T., Nowicki, M. R., Cwian, K., and Skrzypczynski, P. (2019). How to improve object detection in a driver assistance system applying explainable deep learning. In *IEEE Intelligent Vehicles Symposium, Proceedings*. **2019** (6): 226-231

[9] Arrieta, Alejandro Barredo, Díaz-Rodríguez, Natalia, Del Ser, Javier, Bennetot, Adrien, Tabik, Siham, Barbado, Alberto, García, Salvador, Gil-López, Sergio, Molina, Daniel, and Benjamins, Richard. (2020) "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI." *Information Fusion* ***58***: 82-115,

[10] Adadi, Amina, and Berrada, Mohammed. (2018) "Peeking inside the black-box: a survey on explainable artificial intelligence (XAI)." *IEEE Access.* **6**(2018): 52138-52160

[11] Kumar, K. P., Thiruthuvanathan, M. M., K.K, S., and Chandra, D. R. (2023)" Human AI: Explainable and responsible models in computer vision" In *Emotional AI and Human-AI Interactions in Social Networking*. **2023:** 237-254

[12] Gulum, Mehmet A., Trombley, Christopher M., and Kantardzic, Mehmed. (2021) "A Review of Explainable Deep Learning Cancer Detection Models in Medical Imaging." *Applied Sciences.* **11**(2021): 1-21.

[13] Lundberg, S. (2017).”A unified approach to interpreting model predictions.” 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA: **2017:** 1-10

[14] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). " Why should i trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. **2016:** 1135-1144

[15] Friedman, J. H. (2001). “Greedy function approximation: a gradient boosting machine.” *Annals of statistics*. **29**: 1189-1232

[16] Apley, D. W., & Zhu, J. (2020). “Visualizing the effects of predictor variables in black box supervised learning models.” *Journal of the Royal Statistical Society Series B: Statistical Methodology*, **82**(4), 1059-1086

[17] Dhurandhar, A., Chen, P. Y., Luss, R., Tu, C. C., Ting, P., Shanmugam, K., & Das, P. (2018). “Explanations based on the missing: Towards contrastive explanations with pertinent negatives.” *Advances in neural information processing systems*, **31:** 1-12

[18] Yang, C., Rangarajan, A., & Ranka, S. (2018). “Global model interpretation via recursive partitioning.” In 20th International Conference on High Performance Computing and Communications, IEEE. **20:** 1563-1570

[19] Al-Badi, A., Sharma, S. K., Jain, V., & Khan, A. I. (2020). “Investigating emerging technologies role in smart cities’ solutions.” In *Re-imagining Diffusion and Adoption of Information Technology and Systems: A Continuing Conversation: IFIP WG 8.6 International Conference on Transfer and Diffusion of IT, TDIT 2020, Tiruchirappalli, India, December 18–19,* Springer. **2020:** 230-241

[20] Hogan, M., Aouf, N., Spencer, P., and Almond, J. (2022). Explainable Object Detection for Uncrewed Aerial Vehicles using KernelSHAP. In *2022 IEEE International Conference on Autonomous Robot Systems and Competitions, ICARSC 2022.* **2022**: 136-141

[21] Borriero, Alessio, Milazzo, Martina, Diano, Matteo, Orsenigo, Davide, Villa, Maria Chiara, DiFazio, Chiara, Tamietto, Marco, and Perotti, Alan. (2024). Explainable Emotion Decoding for Human and Computer Vision. In *World Conference on Explainable Artificial Intelligence.* Springer. **2024:** 178-201

[22] Liu, G., Zhang, J., Chan, A. B., and Hsiao, J. H. (2024) "Human attention guided explainable artificial intelligence for computer vision models." *Neural Networks*. **177**(2024): 1-16

[23] Anzum, F., Asha, A. Z., Dey, L., Gavrilov, A., Iffath, F., Ohi, A. Q., Pond, L., Shopon, M., and Gavrilova, M. L. (2024)" A comprehensive review of trustworthy, ethical, and explainable computer vision advancements in online social media" In *Global Perspectives on the Applications of Computer Vision in Cybersecurity*. **2024**: 1-46

[24] Morphy, Erika. (2018). “What Is Explainable AI (XAI).” *CMS Wire*. Avaibale at: https://www.cmswire.com/digital-experience/what-is-explainable-ai-xai/. Acceessed on: 30 Sept 2024

Intelligent Automation & Soft Computing
DOI: 10.32604/iasc.2023.032262
Article

Tech Science Press

Explainable Heart Disease Prediction Using Ensemble-Quantum Machine Learning Approach

Ghada Abdulsalam[1], Souham Meshoul[2,*] and Hadil Shaiba[3]

[1]College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O.Box 84428, Riyadh, 11671, Saudi Arabia
[2]Department of Information Technology, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O.Box 84428, Riyadh, 11671, Saudi Arabia
[3]Department of Computer Science, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O.Box 84428, Riyadh, 11671, Saudi Arabia
*Corresponding Author: Souham Meshoul. Email: sbmeshoul@pnu.edu.sa
Received: 12 May 2022; Accepted: 24 June 2022

Abstract: Nowadays, quantum machine learning is attracting great interest in a wide range of fields due to its potential superior performance and capabilities. The massive increase in computational capacity and speed of quantum computers can lead to a quantum leap in the healthcare field. Heart disease seriously threatens human health since it is the leading cause of death worldwide. Quantum machine learning methods can propose effective solutions to predict heart disease and aid in early diagnosis. In this study, an ensemble machine learning model based on quantum machine learning classifiers is proposed to predict the risk of heart disease. The proposed model is a bagging ensemble learning model where a quantum support vector classifier was used as a base classifier. Furthermore, in order to make the model's outcomes more explainable, the importance of every single feature in the prediction is computed and visualized using SHapley Additive exPlanations (SHAP) framework. In the experimental study, other standalone quantum classifiers, namely, Quantum Support Vector Classifier (QSVC), Quantum Neural Network (QNN), and Variational Quantum Classifier (VQC) are applied and compared with classical machine learning classifiers such as Support Vector Machine (SVM), and Artificial Neural Network (ANN). The experimental results on the Cleveland dataset reveal the superiority of QSVC compared to the others, which explains its use in the proposed bagging model. The Bagging-QSVC model outperforms all aforementioned classifiers with an accuracy of 90.16% while showing great competitiveness compared to some state-of-the-art models using the same dataset. The results of the study indicate that quantum machine learning classifiers perform better than classical machine learning classifiers in predicting heart disease. In addition, the study reveals that the bagging ensemble learning technique is effective in improving the prediction accuracy of quantum classifiers.

Keywords: Machine learning; ensemble learning; quantum machine learning; explainable machine learning; heart disease prediction

1 Introduction

Healthcare is one of the most influential fields on the global population's safety. The continuous evolution of the healthcare sector facilitates disease prediction, treatment, diagnosis, and cure. The advancement of research and technologies in healthcare and public health has significantly decreased global mortality, with the advanced healthcare system helping in the prevention of disease progression and improvement of life quality. However, the healthcare sector has recently experienced an explosion of data and increased system complexity. The scope and quality of healthcare data open up new opportunities for healthcare practitioners to utilize advances in data science to extract valuable insights from these enormous databases. Advancements in data analytics, computing power, and algorithms are rapidly changing the prospect of healthcare by improving clinical and operational decision-making [1]. Artificial intelligence's (AI) ability to derive conclusions from input data has the potential to revolutionize the delivery of care and assist in addressing a variety of healthcare challenges. Indeed, AI can improve healthcare outcomes, healthcare systems, patient experience, and treatment. The investment of AI in healthcare is growing rapidly, as it increases the efficiency and effectiveness of care delivery and enables healthcare systems to provide care to more people [1]. Machine learning (ML) is a subfield of artificial intelligence used to solve specific problems, such as predictive problems, by teaching a model to learn from an input dataset to predict the output. Machine learning plays a crucial role in the healthcare industry, where it is primarily used to assist in decision-making for a variety of diseases involving prediction, diagnosis, and medical image analysis [2]. Ensemble learning (EL) is a notable technique for enhancing the performance of these machine learning models. EL refers to the application of ensemble methods in which multiple ML models individually contribute to the prediction task. They have been demonstrated to be a successful addition to the field of predictive ML models, as their predictive performance is superior to that of their constituents. On another side, quantum machine learning (QML), the intersection of classical machine learning and quantum computing, is a recently developed field that has attracted researchers from a variety of disciplines due to its flexibility, representation power, and promising scalability and speed results. Quantum computers take advantage of quantum mechanical properties to improve the processing efficiency of classical computers. Consequently, QML algorithms running on quantum computers have the potential to perform extremely rapid calculations for problems that are challenging to solve with classical computers. QML can handle complex medical situations and improve system performance, thereby providing substantial benefits to the healthcare industry. Quantum computing can enable a variety of quantum use cases that are essential to the ongoing transformation of healthcare, such as diagnostic assistance, precision medicine, and price optimization. Numerous studies have demonstrated that QML algorithms offer substantial advantages over traditional machine learning algorithms for a variety of applications, including healthcare [3,4].

Cardiovascular diseases (CDVs) is a medical term that refers to diseases that affect both the heart and blood vessels and can lead to heart attack, stroke, and heart failure. According to the World Health Organization (WHO), CDVs cause approximately 17.9 million deaths annually, or 32% of all deaths worldwide. Among the risk factors for these diseases are high blood pressure, high blood glucose levels, obesity, and high blood lipid levels [5]. Heart disease is among the most prevalent cardiovascular diseases that threaten global public health. The prevalence of heart disease and consequently the increased risk of death pose a grave threat to human security. In order to prevent the disease's progression and lessen its long-term effects, it is crucial to investigate the early detection of heart disease through specific physical indicators. The treatment of heart disease has been an active area of research for a very long time, so that preventative measures can be taken in the early stages. Consequently, a number of researchers utilized machine learning techniques to predict heart disease risks. A variety of medical fields have adopted machine learning strategies, as researchers strive to enhance and optimize medical decision-making. Methods of machine learning are primarily used to automatically detect patterns in the intended dataset,

without the need for human intervention. However, classical machine learning approaches can be enhanced to obtain better results and more reliable models by employing various machine learning techniques, such as ensemble learning, which has been shown to improve the performance of models in predicting heart disease [6–10]. Alternatively, quantum computing can provide a more efficient framework for machine learning than classical approaches. Quantum machine learning is a relatively new field of study that has made considerable progress in recent years. Motivated by the development of various machine learning models to predict the risk of heart disease and in an effort to improve classification performance, the objective of this study was to explore the potential of ensemble learning in addition to quantum machine learning for predicting the risk of heart disease by designing an ensemble learning model based on a quantum machine learning model. Given the fact that machine learning methods can provide accurate solutions for predicting the presence of heart disease, the research was conducted using both well-known classical machine learning algorithms and quantum machine learning algorithms.

This research utilized a four-step methodology: Initially, the data are prepared using multiple pre-processing techniques, including feature selection, feature extraction, and normalization. Second, analyzing the performance of classical machine learning classifiers such as Support Vector Classifier (SVC) and Artificial Neural Network (ANN) and quantum machine learning classifiers to investigate the potential of QML models in comparison to traditional ones. Then, applying three distinct quantum machine learning classifiers namely Quantum Support Vector Classifier (QSVC), Quantum Neural Network (QNN), and Variational Quantum Classifier (VQC)). The objective of this step is to identify the best performing QML model for the given dataset. Finally, designing, applying and evaluating a Quantum Support Vector Classifier-based bagging ensemble learning model (Bagging-QSVC). In addition to explaining the significance and indication of the findings, the interpretation of a machine learning model facilitates the understanding of why a particular decision was made. The interpretability of the proposed model, where the importance and contribution of each feature to the prediction are computed and visualized using the SHapley Additive exPlanations (SHAP) framework, is thus another novel aspect of this work. To evaluate the models, several performance metrics, including accuracy, recall, precision, F1-score, and ROC, were calculated, with the results indicating that the quantum classifiers outperform the classical classifiers in terms of performance. In addition, the study demonstrated that ensemble learning yields superior predictive performance for quantum classifiers compared to classical classifiers. In addition, it can be deduced that combining ensemble learning with quantum classifiers can produce successful results.

The remaining sections are organized as follows. In Section 2, a review of related work is presented. Section 3 is devoted to the description of the background material used in this study, which is comprised of the two main ingredients EL and QML, as well as the methodology employed, with a focus on the data set, the investigated models, and the proposed ensemble model. Section 4 provides the experimental study, obtained results, and model interpretation. In Section 5, conclusions and future work are drawn.

2 Related Work

Machine learning classification algorithms are vastly utilized in many fields to solve numerous problems. A field such as healthcare is considered a rich machine learning domain, where machine learning can be employed to tackle various medical decisions. Heart disease is a major health problem investigated by researchers using novel machine learning methods. Ensemble learning is one of the machine learning methods that has proven to boost machine learning performance. A remarkable number of previous works utilized ensemble learning to improve the accuracy of heart disease prediction using multiple methodologies. In the light of recent past studies, Gao et al. [6] used boosting and bagging ensemble learning methods besides Principal Component Analysis (PCA) and Linear Discriminant

Analysis (LDA) feature extraction algorithms to improve heart disease prediction. The study also applied five different classical machine learning algorithms and compared their performance with the performance of the ensemble learning methods. The results of the experiment indicated that the bagging ensemble learning method based on the decision tree classifier and principal component analysis feature selection achieved the highest accuracy. Another study conducted by Mienye et al. [7] proposed an enhanced machine learning method that splits up the dataset randomly into smaller subsets, and the different subsets are then modelled by making use of the Classification and Regression Tree (CART). An ensemble learning model is then generated from the different CART models using a modified version of the Weighted Aging classifier Ensemble approach (WAE) known as Accuracy-Based Weighted Aging classifier Ensemble (AB-WAE). Latha et al. [8] presented a more comprehensive study that applied to the Cleveland heart disease dataset using bagging, boosting, majority voting, and stacking ensemble learning techniques. The result of the study showed that the majority voting approach induced the highest improvement of accuracy. In a slightly different manner, Anuradha et al. [9] modelled eXtreme Gradient Boosting (XGBoost), and Category Boosting (CatBoost) classifiers together with hard majority voting ensemble classifier on three different datasets, namely, Cleveland, Statlog, and South African heart disease. The outcomes of the study revealed that the hard majority vote ensemble classifier achieved the best performance on the Statlog dataset. Practically, Uddin et al. [10] evolved an intelligent agent using an ensemble method-based Multilayer Dynamic System (MLDS) to predict cardiovascular disease. In every layer, three classification algorithms (*i.e.*, Random Forest (RF), Naïve Bayes (NB), and Gradient Boosting (GB)) were applied to construct the ensemble model. Uddin and Halder used a realistic dataset consisting of 70,000 instances to test the proposed model effectively. The test results showed that MLDS has proven to be able to efficiently predict the risk of cardiovascular disease compared to five other different models. In the same context, Rahim et al. [11] suggested a Machine Learning-based Cardiovascular Disease Diagnosis framework (MaLCaDD) to improve the prediction accuracy of cardiovascular diseases. Rahim et al. stated that increasing the accuracy of the prediction using various feature selection and classification methods has taken the most attention of the researchers. However, handling missing values and class imbalance problems gained less attention while improving accuracy. In a more improved manner, Ali et al. [12] enhanced the automation process of heart disease prediction by developing a smart monitoring framework that consists of a heart disease prediction system based on an ensemble deep learning approach in addition to an ontology-based recommendation system. The ensemble deep learning model comprises a feed-forward neural network along with back-propagation techniques and gradient algorithms to minimize the errors. In the past decade, Das et al. [13] investigated the performance of bagging, Adaptive Boosting (Adaboost), and random subspace ensemble learning methods for diagnosing valvular heart disease. The empirical results of the study proved the efficiency of the ensemble learning approach over the single classifiers.

Among several studies, the majority voting ensemble learning approach is largely used in the literature of heart disease [2,8,10,14]. Raza [2] ensembled three different classifiers (*i.e.*, Logistic Regression (LR), Naïve Bayes (NB), and Artificial Neural Network (ANN)), and combined the results of these classifiers using a majority voting combiner which provided better performance over other combiners. Raza mentioned the three different forms of majority voting approach, which are unanimous voting, simple majority, and plurality voting. Likewise, Mehanović et al. [14] applied an ensemble learning model using Artificial Neural Network (ANN), k-Nearest Neighbour (KNN), and Support Vector Machine (SVM) classifiers to predict heart disease in patients using binary classification (0 for the absence of disease and 1 for the presence), and multi-classification (0 for the absence of disease and 1, 2, 3, 4 for the presence). In both classification cases, the highest accuracies were gained by using the majority voting approach.

On other hand, few researchers have addressed the ensemble learning approach besides quantum machine learning to solve the heart disease prediction problem [4]. Schuld et al. [15] applied quantum

ensemble learning on quantum classifiers, where quantum classifiers were assessed in parallel, and their integrated decision was accessed by a qubit measurement. Kumar et al. [4] utilized QML to detect heart failure on the Cleveland dataset. The empirical study proved that quantum-enhanced machine learning algorithms such as Quantum K-Nearest Neighbour (QKNN), Quantum Gaussian Naïve Bayes (QGNB), Quantum Decision Tree (QDT), and Quantum Random Forest (QRF) present better results than traditional machine learning algorithms in heart failure detection. In another context, Maheshwari et al. [16] implemented multiple ensemble learning models that combine classical and quantum methods on a diabetes dataset. The classical ensemble learning methods used in the study were Random Forest (RF), XGBoost, and Adaboost, whereas, the Quantum Boosting (QBoost) classifier was the quantum ensemble learning method that uses Quantum Annealing (QA) to find the best learners collection. Our work expands on these efforts by investigating the potential of quantum machine learning-based ensemble learning in heart disease prediction while explaining the model's outcomes. This study investigates several classical and quantum machine learning classifiers as well as a novel heart disease prediction model to achieve accurate heart disease prediction. The novel model combines ensemble learning with quantum machine learning. In addition, this is the first study to propose an interpretation of the heart disease prediction model using the Cleveland data set and the proposed model based on the SHapley Additive exPlanations (SHAP) framework. To the best of our knowledge, no previous studies have looked into this.

3 Materials and Methods

3.1 Ensemble Learning

The field of machine learning investigates algorithms and techniques that allow computers to automatically find solutions to complex problems that traditional programming methods cannot solve. Machine learning can be leveraged to provide insights into the pattern in a dataset by trying to design an efficient model that learns from a training dataset to predict outcomes [17,18]. Typically, the success of machine learning algorithms is determined by the structure of the data, which can be learned in a variety of ways. However, the ML model's balanced complexity is critical for obtaining accurate results. All ML algorithms have some level of bias and variance, where a model with higher bias does not fit training and test data well due to the model's low complexity, whereas a model with higher variance does not fit test data well but has a much lower error on training data due to the model's high complexity. In general, there are four learning models in machine learning, namely, supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning [17]. Machine learning applications are used in a wide range of fields, including Natural Language Processing (NLP), speech processing, computer vision, text classification, and computational biology [18].

Ensemble learning is a popular machine learning approach that combines several models and then assembles their outputs to make more accurate predictions [18,19]. In fact, each learning algorithm has strengths and weaknesses since it is not expected for a single model to fit all scenarios correctly. As a result, rather than creating a single model that may result in poor performance, ensemble learning techniques aggregate various models to achieve better generalization performance than any of the ensemble model's single basic components [19,20]. Ensemble learning was initially developed to reduce variance in automated decision-making systems, but it has since gained widespread attention due to its flexibility and effectiveness in a variety of machine learning areas and experimental applications. A variety of machine learning problems, including feature selection, class imbalanced data, confidence estimation, incremental learning, missing feature, and error correction, have been addressed using ensemble learning systems [21]. In general, three approaches are commonly used to generate ensemble systems: bagging, boosting, and stacking, as well as improved versions of these methods that developed to solve specific problems [22].

The bagging or (bootstrap aggregation) method produces one model at a time from a random sample or (bootstrap sample) that has the same size as the dataset [19]. As a result, models trained on dissimilar bootstrap samples will differ depending on the random bootstrap sample used to train the model [2]. The final prediction is generated after fitting and aggregating the models by calculating the average or majority voting of the overall prediction models. When results from multiple models are combined, the bagging ensemble model can produce an optimal model with higher confidence, lower variance, and lower bias error than single models. Random Forests (RF), which is based on the decision-tree algorithm, and K-Nearest Neighbour (KNN) subspace bagging, which is based on the k-Nearest Neighbour algorithm, are two significant extensions of the bagging approach [22].

3.2 Quantum Machine Learning

Personal computers are working on the concept of classical mechanics, which uses a bit of 0 or 1 as a fundamental unit of electronic circuits. However, quantum machines employ a quantum bit (or qubit) that can simultaneously occupy two fundamental states, 0 and 1 [23]. This property of quantum machines is known as the superposition of states, in which a qubit can be in both $|0\rangle$ and $|1\rangle$ states at the same time. The superposition of the states allows the operations to run in parallel rather than sequentially, reducing the number of operations required in any algorithm [24]. Furthermore, quantum computers make use of the entanglement property, which allows quantum particles to gain a stronger correlation by connecting to the properties of other particles. As a result, despite large physical distances, the qubits can be correlated with each other [25]. These quantum physics behaviours give quantum machines an advantage over classical machines by speeding up many tasks that would otherwise take a very long time using classical algorithms [23]. In quantum mechanics, the basis states 0 and 1 are represented by two-dimensional vectors, where:

$$\vec{0} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \vec{1} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{1}$$

The state of a quantum system is given by a vector that has a particular notation in quantum systems called the Dirac notation which is denoted by $|\psi\rangle$ [20]. The state $|\psi\rangle$ can demonstrate the linear combination of the basis states $|0\rangle$ and $|1\rangle$ as presented in Eq. (2).

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2}$$

The linear coefficients alpha (α), and beta (β) belong to the complex numbers, *i.e.*, $\alpha, \beta \in \mathbb{C}$. The complex numbers α and β indicate that the qubit has a probability of $|\alpha|^2$ to appear in state $|0\rangle$ and a probability of to appear in state $|1\rangle$ [23]. Thus, the sum of the probabilities of a qubit to appear in each one of the basis states is equal to 1 as shown in Eq. (3).

$$|\alpha|^2 + |\beta|^2 = 1 \tag{3}$$

A qubit can be visualized using a Bloch sphere which is a geometric representation of the qubit states. The continuous combination of the two states $|0\rangle$ and $|1\rangle$ can be placed in any potential points on the Bloch sphere. A qubit is represented on a Bloch sphere as a point on the surface of the sphere. Hence, a generic quantum state $|\psi\rangle$ can be represented by the three parameters θ, γ and ϕ of real numbers, Where, $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$, as shown in Eq. (4).

$$|\psi\rangle = e^{i\gamma}\left(\cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varnothing}\sin\left(\frac{\theta}{2}\right)|1\rangle\right) \tag{4}$$

The global factor $e^{i\gamma}$ has no noticeable impact among other factors, thus it can be neglected in the equation. Therefore, Eq. (5) illustrates the final equation after omitting the $e^{i\gamma}$ factor [26].

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varnothing}|1\rangle \tag{5}$$

The states in quantum computing can be represented in the Bloch sphere as a vector that starts at the original centre and ends on the sphere surface, where the vector is represented by an arrow pointing to a location on a sphere. The three-dimensional graphical representation of a single qubit using the Bloch sphere is represented in Fig. 1.

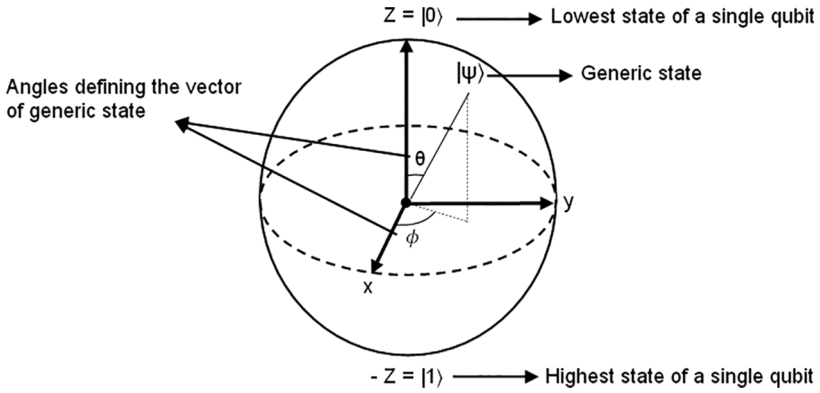


Figure 1: Qubit representation using Bloch sphere

Quantum Machine Learning (QML) field represents the intersection between the concepts of machine learning and quantum computing [25]. Quantum algorithms were developed in quantum machine learning to handle classical algorithms using quantum computers [27]. Quantum machine learning offers a great opportunity to improve the computational proficiency of classical machine learning algorithms and handle some of the more computationally complex problems that classical machine learning algorithms cannot effectively solve [23,28]. Quantum machine learning applies classical machine learning methods in quantum systems by using various quantum properties such as superposition and entanglement that were mentioned previously. The superposition of states property produces parallelism in quantum computers, allowing the evaluation of multiple input functions at the same time. Furthermore, the entanglement property provides a method for increasing storage capacity [27]. These two and other QML properties improve the performance of classical machine learning algorithms by providing significant computational acceleration at run-time based on complexity. Grover's Algorithm and the Harrow-Hassidim-Lloyd (HHL) Algorithm [25] are the two main quantum algorithms used by machine learning techniques to gain speedup. Regardless of the speedup property of QML, research on quantum machine learning focuses on other types of QML properties that represent advancements in quantum machine learning techniques over classical ML, such as model complexity, sample complexity, and robustness to noise [29]. Furthermore, several quantum algorithms, such as the Quantum Support Vector Machine (QSVC), Quantum Clustering technique, Quantum Neural Network (QNN), and Quantum Decision Tree, were developed to run classical algorithms on quantum computers (QDT). As a result, using supervised and unsupervised quantum algorithms implemented through quantum models, the data can be classified, categorized, and analysed [27].

3.3 Methodology

The purpose of this study is to investigate the potential of quantum machine learning algorithms for predicting heart disease. As a result, the research was divided into four distinct phases, the first of which dealt with the Cleveland dataset using various pre-processing techniques, including Recursive Feature

Elimination (RFE) for feature selection, Principal Component Analysis (PCA) for feature extraction, and Min-Max normalization. In the second phase, classical classifiers (SVC and ANN) were compared to quantum classifiers. Three different quantum machine learning classifiers (QSVC, QNN, and VQC) were investigated in the third phase. Finally, a bagging ensemble learning model based on Quantum Support Vector Classifier (Bagging-QSVC) has been designed and implemented.

3.3.1 Dataset

This research uses the UCI machine learning repository's [30] Cleveland benchmark dataset. The Cleveland dataset consists of 303 instances and 14 attributes, with a two-level target attribute representing a binary classification where label 1 indicates patients with heart diseases and 0 indicates patients without heart disease. Since 165 instances have label 1 and 138 instances have label 0, the dataset is roughly balanced. Tab. 1 describes the Cleveland dataset's features.

Table 1: Features description of heart disease dataset

Feature code	Feature name	Data type	Description
Age	Age	Numerical (continuous)	Age in years
Sex	Sex	Categorical (binary)	1 = male, and 0 = female
CP	Chest Pain types	Categorical (multi-valued)	Chest Pain types: 1 = typical angina, 2 = atypical angina, 3 = non-angina pain, and 4 = asymptomatic
Trestbps	Resting blood pressure	Numerical (continuous)	Resting blood pressure (in mm Hg)
Chol	cholesterol	Numerical (continuous)	Serum cholesterol (in mg/dl)
Fbs	Fasting blood sugar	Categorical (binary)	Fasting blood sugar > 120 mg/dl: 1 = true, or 0 = false
Thalach	Maximum heart rate	Numerical (continuous)	maximum heart rate reached during thallium test
Restecg	Resting electrocardiographic	Categorical (multi-valued)	Resting electrocardiographic result: 0 = normal, 1 = having ST-T wave abnormality, 2 = showing probable or definite left ventricular hypertrophy
Exang	Exercise-induced angina	Categorical (binary)	Exercise-induced angina: 0 = no, and 1= yes
Oldpeak	ST depression	Numerical (continuous)	ST depression caused by exercise relative to rest
Slope	ST slope	Categorical (multi-valued)	The peak exercise ST segment slope: 1 = ascending, 2 = flat, 3 = descending

(Continued)

Table 1 (continued)

Feature code	Feature name	Data type	Description
Ca	Number of major vessels	Categorical (multi-valued)	Number of main vessels: 0, 1, 2, 3, and 4
Thal	Thallium heart test	Categorical (multi-valued)	Exercise Thallium heart test result: 1 = normal, 2= fixed defect, and 3 = reversible defect
Target	Heart disease	Categorical (binary)	Heart disease diagnosing: (1 = yes, 0 = no)

3.3.2 Quantum Support Vector Classifier (QSVC)

In quantum computers, Quantum SVC is the quantum counterpart of the classical SVC. Since the classical SVC handles problems in higher dimension space, the computational resources required to solve them on classical computers can be costly and time-consuming [26]. The QSVC has a quantum advantage over the classical SVC in situations where it is difficult to estimate the feature map classically. Using a quantum kernel in QSVC algorithms, quantum computers can accelerate learning in SVC by using a quantum kernel. To investigate the potential of QSVC, the classical-quantum (CQ) method was used, which entails utilizing a classical dataset on quantum computers. Using quantum feature maps that map data points to quantum states, classical data can be encoded to be processed by a quantum computer [24]. Fig. 2 depicts the structure of the QSVC algorithm, in which the dataset was divided into training and testing datasets with ratios of 80:20 respectively, and pre-processed with RFE and Min-max normalization. The classical feature vectors were subsequently mapped to quantum spaces using a 5-qubit feature map (the number of qubits in the circuit is equal to the number of selected features namely Number of major vessels, Chest Pain types, Thallium heart test, Exercise-induced angina, and ST-slope. By taking the inner product of the quantum feature maps, the quantum kernel maps the quantum state data points into higher-dimensional space. After fitting the QSVC classifier to the training data and evaluating the performance of the model using the test data, for each classical input, the binary measurements decode the quantum data into the corresponding classical output data (a classical value of 0 or 1).

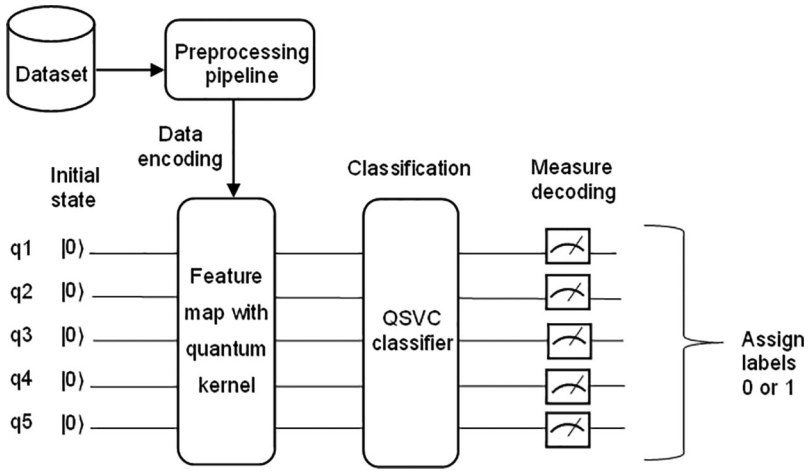


Figure 2: QSVC outline

3.3.3 Quantum Neural Network (QNN)

Quantum Neural Networks combine the fundamentals of conventional ANN with quantum computation models that outperform conventional ANN [31]. QNN is an algorithm for machine learning that employs quantum computers to classify datasets by training various parameters within quantum circuits. QNN can be used to solve problems requiring a large amount of memory and capacity [24]. Fig. 3 depicts the structure of the QNN algorithm where the dataset was divided into training and testing datasets using 8-fold cross-validation, and pre-processed using RFE, PCA and Min-max normalization. Thereafter, the classical feature vectors are mapped to quantum spaces using a feature map of 2 qubits (the number of qubits in the circuit is equal to the number of PCA components). A CircuitQNN network that is based on a parametrized quantum circuit was constructed after that to get the input data and weights as parameters and produces a batch of binary output. After fitting the QNN classifier with the training data and testing the model performance using cross-validation iterators, the binary measurements decode back the quantum data into appropriate output data (a classical value of 0 or 1).

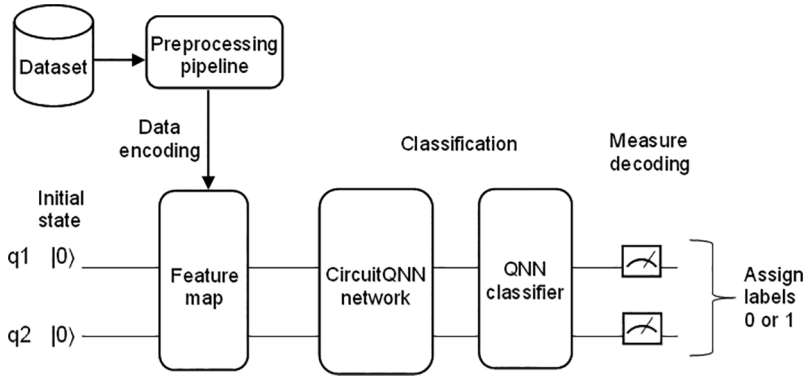


Figure 3: QNN classifier outline

3.3.4 Variational Quantum Classifier (VQC)

Variational Quantum Classifier [24] is a quantum supervised learning algorithm that employs variational circuits to perform classification tasks. In quantum variational classification, the data is mapped to a quantum state using the feature map circuit, and a parameterized and trained short-depth quantum circuit is applied to the feature state [32]. Fig. 4 presents the outline of the VQC algorithm where the dataset was divided into training and testing datasets with ratios of 80:20 respectively, and pre-processed using RFE, PCA and Min-max normalization. Thereafter, the classical feature vectors are mapped to quantum spaces using a feature map of 2 qubits (the number of qubits in the circuit is equal to the number of PCA components). After fitting the VQC classifier with the training data and testing the model performance using the test data, the binary measurements decode the quantum data into a classical value of 0 or 1.

3.3.5 The Proposed Bagging Model for Heart Disease Prediction

Since it has been proven that ensemble learning enhances the performance of the models, the model proposed in this work consists of a bagging ensemble model with QSVC Classifier (Bagging-QSVC). QSVC was chosen with the bagging model because it achieved the highest performance among the three quantum classifiers. Nevertheless, in the bagging ensemble method, each model was trained on a random sample of the dataset, where the random samples (or bootstrap samples) have the same size as the original dataset. Bagging models make use of sampling with a replacement that duplicates or ignores some instances from the original dataset in each bootstrap sample. Therefore, each subsample had different instances, and these subsamples were used to train 100 different QSVC models that fitted in

parallel. Subsequently, once the separated models in the ensemble had been trained, the ensemble aggregated their predictions by returning the class that gained the majority of the votes to get the final output of the ensemble model. The proposed Bagging-QSVC model is illustrated in Fig. 5.

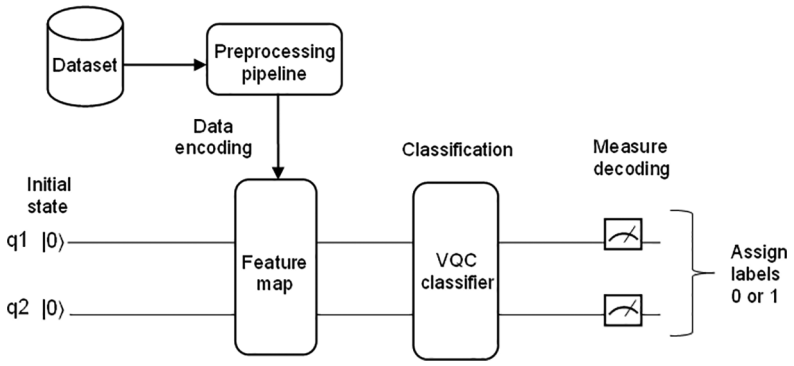


Figure 4: VQC outline

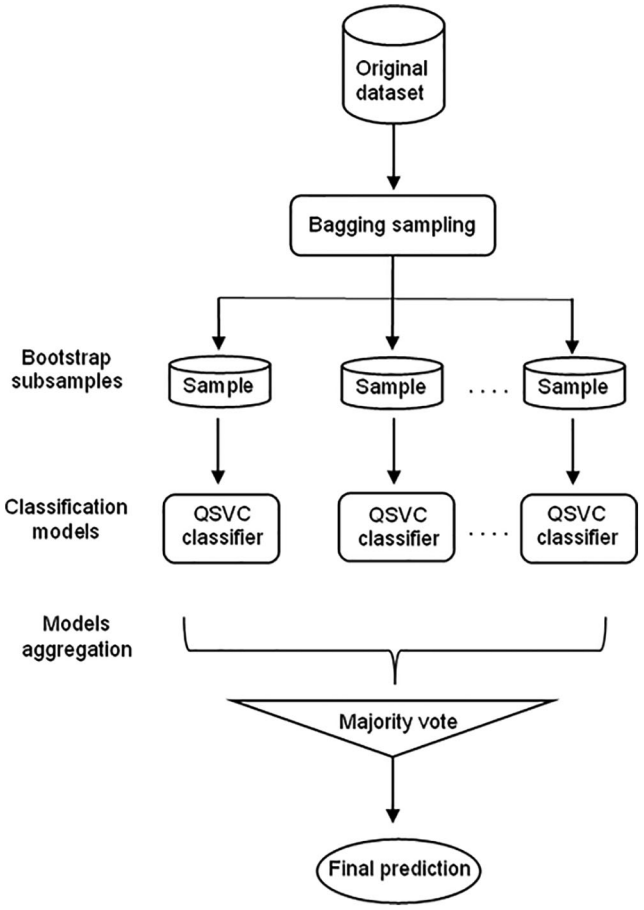


Figure 5: Bagging-QSVC architecture

4 Experimental Study and Results

4.1 Experimental Setups

The experimental study was conducted using qiskit in jupyter notebook and python. The code is accessible at [33]. In addition to quantum and classical classifiers, the Bagging-QSVC model was implemented on a simulator. The simulator was operating on a computer system with a 7th generation Core i7 processor and 8 GB of RAM.

4.2 Results and Discussion

To evaluate the performance of the predictive models applied in this work (QSVC, SVC, QNN, ANN, VQC, and Bagging-QSVC), a set of performance measures including accuracy, precision, recall, F_1-measure and area under the curve (AUC) or ROC index have been used. The definitions and equations of these performance measures are provided below Eqs. (6)–(9), where True Positive (TP) represents the positive instances classified correctly, True Negative (TN) represents the negative instances classified correctly, False Positive (FP) represents the positive instances classified incorrectly, and False Negative (FN) represents the negative instances classified incorrectly.

– Accuracy: The percentage of the total number of instances that are correctly classified relative to the number of all tested instances.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{6}$$

– Precision: The ratio between the number of positive instances that are correctly classified and all instances predicted as positive. The precision presents how confident an instance predicted with a positive target actually has a positive target level.

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

– Recall (or sensitivity): The ratio between the positive instances and the number of all actual positive instances. The recall presents how confident all the instances with a positive target the model found.

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

– F_1-measure: The harmonic mean of precision and recall measures. The F-measure, precision, and recall can assume values in the range [0,1], where the larger values indicate better performance.

$$F_1 measure = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)} \tag{9}$$

– The ROC index (or Area Under the Curve): The ROC curve relates the True Positive Rate (TPR) (the positive points correctly predicted as positive) to the False Positive Rate (FPR). The diagonal of the ROC curve represents the expected performance of a model with random predictions, while the closer the curve is to the upper left corner (or a higher AUC value), the more predictive the model. The ROC index or AUC can take on values between 0 and 1, with larger values indicating superior model performance [19].

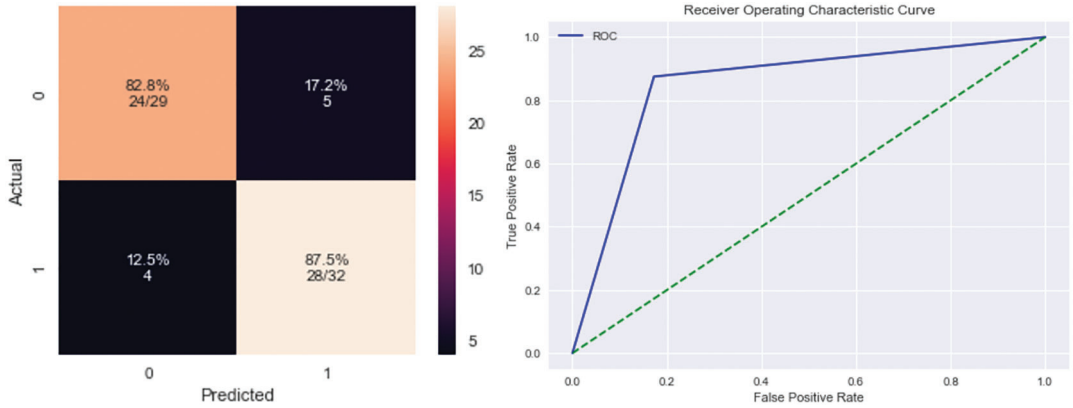
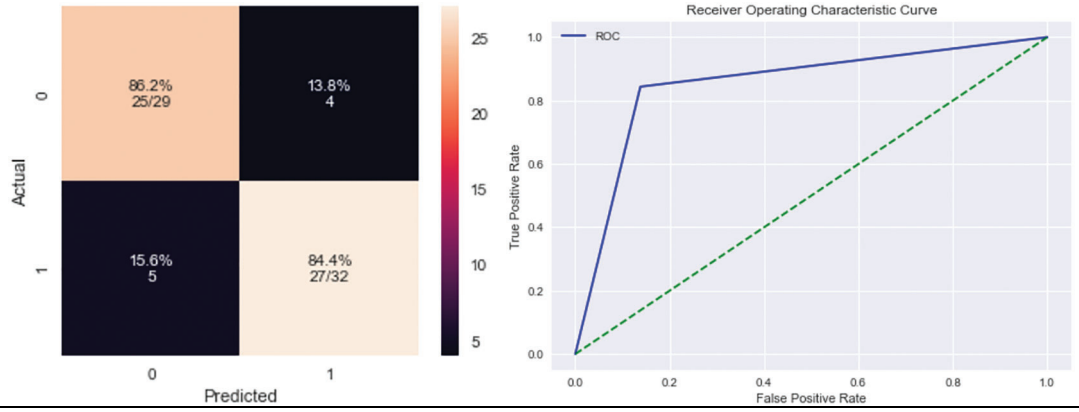
Tab. 2 shows the results based on the aforementioned performance measures of the classical and quantum classifiers, as well as the proposed model (Bagging-QSVC). According to the results of the experiment, QSVC achieved an accuracy of 88.52%, while traditional SVC achieved 85.24%. Similarly, QNN achieved an accuracy of 86.84%, whereas ANN only achieved 85.24%. QSVC, QNN, and VQC achieved the highest performance for quantum classifiers with 88.52%, 86.84%, and 85.25%,

respectively. This indicates that quantum classifiers are capable of producing better results than their counterpart classical classifiers. On the other hand, Tab. 2 depicts the improvement of the proposed model (or Bagging-QSVC) relative to other quantum and classical classifiers through the use of ensemble learning. The Bagging-QSVC outperformed all other classifiers with an accuracy of 90.16%, indicating that ensemble learning improves the performance of quantum classifiers. Tab. 3 also depicts the confusion matrix and the ROC curve of the predictive models.

Table 2: Performance results of the predictive models

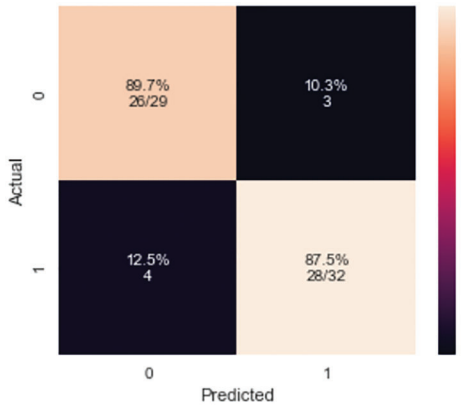
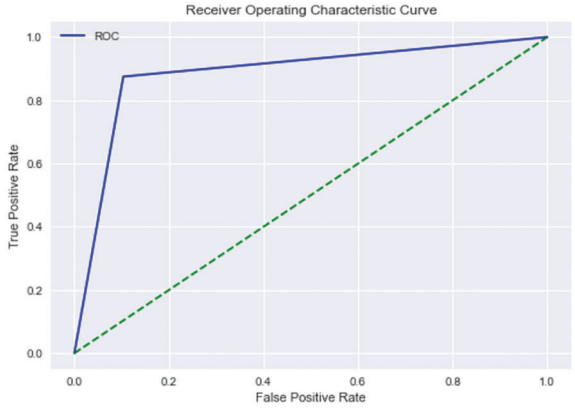
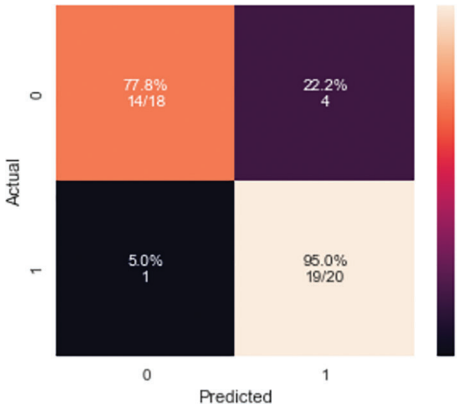
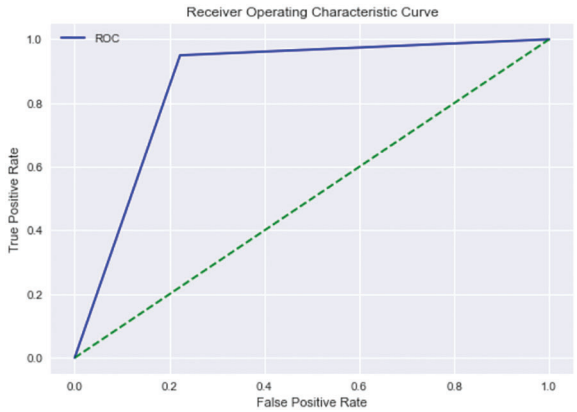
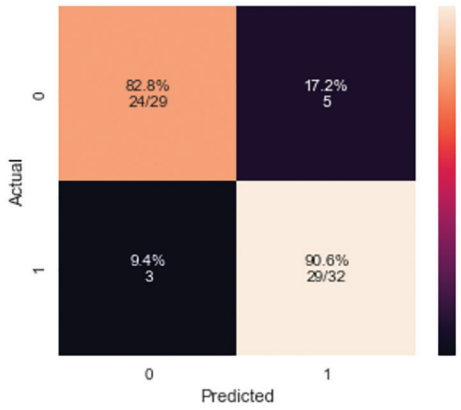
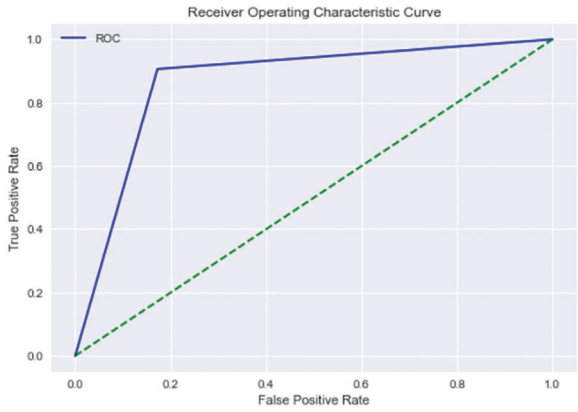
Classifier	Accuracy	Precision	Recall	F1-score
SVM	85.24%	0.85	0.85	0.85
ANN	85.24%	0.85	0.85	0.85
QSVC	88.52%	0.88	0.89	0.89
QNN	86.84%	0.88	0.86	0.87
VQC	86.89%	0.85	0.85	0.85
Bagging-QSVC	90.16%	0.90	0.90	0.90

Table 3: Confusion matrixes and the ROC curves of the predictive models

Classifier	Confusion matrix	ROC curve
SVM		
ANN		

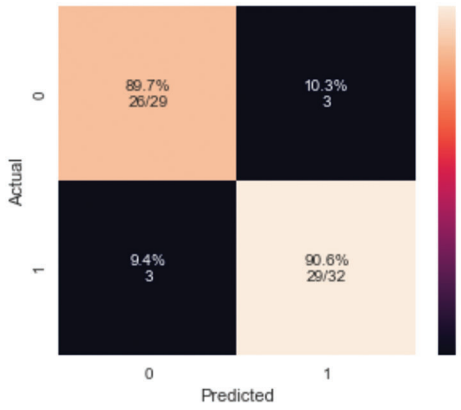
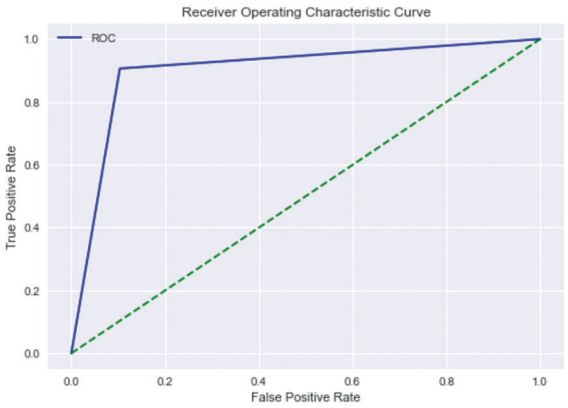
(Continued)

Table 3 (continued)

Classifier	Confusion matrix	ROC curve
QSVC		
QNN		
VQC		

(Continued)

Table 3 (continued)

Classifier	Confusion matrix	ROC curve
Bagging-QSVC		

The comparative analysis shows that the proposed Bagging-QSVC outperforms some previous studies in predicting heart disease with improved accuracy. Tab. 4 compares the results of our proposed model and models of other studies applied to the same dataset using different ensemble learning approaches.

Table 4: Comparison of our study with other studies on the Cleveland dataset

Study	Ensemble learning method	Accuracy achieved
Latha et al. [8]	Majority voting	85.48%
Tama et al. [34]	Stacking ensemble learning	85.71%
Mehanović et al. [14]	Majority voting	87.37%
Alim et al. [35]	Random Forest	86.94%
Kumar et al. [4]	Quantum random forest	89%
Our proposed model	Bagging-QSVC	90.16%

4.3 Model Interpretation

The SHapley Additive exPlanations (SHAP) framework was used to interpret and explain the model results [36]. As a result, the SHAP python library was used to compute and visualize the significance of each and every feature in the prediction. The calculation of SHAP values is the foundation of this framework. A SHAP value is a feature contribution measure that is used to improve the interpretability of machine learning models. SHAP values explain how to get from the predicted or base value E[f(x)] to the actual output f if the features are unknown (x). These values also show how features influence prediction by indicating the direction of the relationship between the features and the target variable. A feature with a SHAP value closer to 1 or −1 has a strong positive or strong negative contribution to the prediction of a specific data point, whereas a feature with a SHAP value closer to 0 has a small contribution to the prediction [36]. Several plots that help to understand the contribution of features can be obtained using this framework.

4.3.1 SHAP Beeswarm Plot

The Beeswarm plot represents feature importance in descending order, as well as feature impacts on prediction, whether positive or negative. The SHAP values are used to depict the impact of higher and lower feature values on the model output. The feature determines the position on the y-axis, and positive and negative SHAP values determine the position on the x-axis. Each point in the plot represents a single observation, and the colour represents how the higher and lower values of the feature affect the result, with red representing a higher value and blue representing a lower value of a feature. Fig. 6 shows the Beeswarm plot for our model. It can be observed clearly that the number of major vessels has the greatest influence on the predictions, followed by the thallium heart scan. Exercise-induced angina, on the other hand, has the smallest change in the prediction of heart disease probability and thus has the least importance. The feature values are also shown in Fig. 6 to represent the impact of each feature on the prediction. For example, ST slop values (1 = ascending, 2 = flat, and 3 = descending) indicate that ascending and flat ST slop are associated with a lower risk of heart disease, whereas descending ST slop is associated with a higher risk of heart disease.

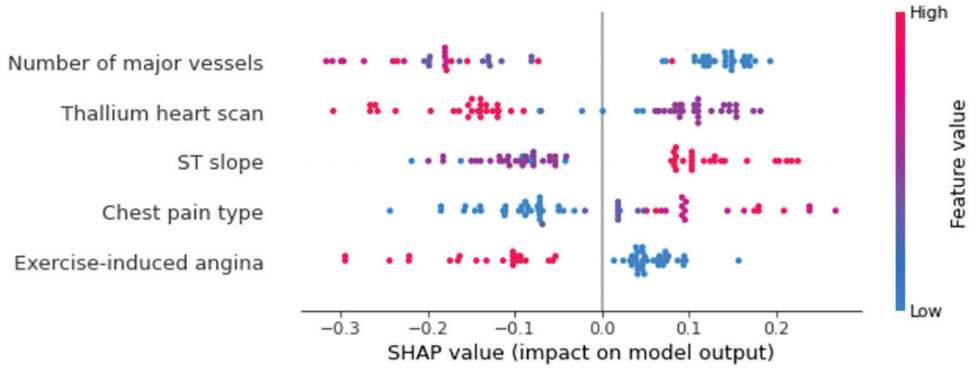


Figure 6: SHAP Beeswarm plot

4.3.2 SHAP Force Plot

The SHAP force plot visualizes the SHAP values of each feature as a force that either increases or decreases the prediction, representing the contribution of each feature to the model's prediction of a particular observation. Each feature's force is represented by a red or blue arrow, depending on whether it increases or decreases the model's score. The features that have a greater influence on the prediction are located closer to the dividing line, and the size of the arrow represents the magnitude of this influence. The force plot illustrates the significance of each feature in adjusting the model to increase or decrease the prediction based on a SHAP value baseline. These characteristics counterbalance one another to determine the final prediction of the data instance. Fig. 7 shows our model's force plot for a random observation. As can be seen, it reveals that the Chest Pain type has the greatest influence on the prediction, followed by Number of major vessels and Exercise-induced angina, respectively. These three predictive characteristics raise the model's score (to the right), which should result in a final prediction of 1. Otherwise, Thallium heart scan and ST slop have less influence, which reduces the model's predictive accuracy (to the left).

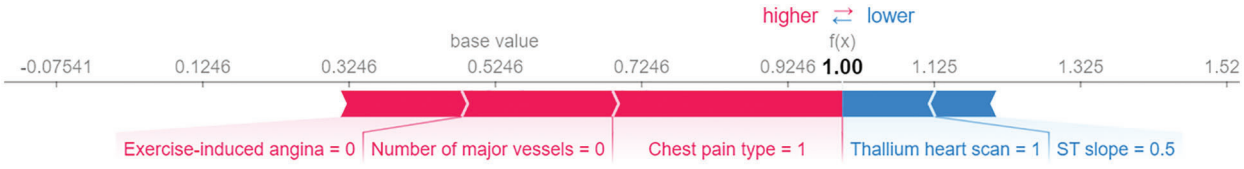


Figure 7: SHAP force plot

4.3.3 Stacked SHAP Force Plot

As implied by its name, the Stacked SHAP force plot combines multiple force plots, each of which depicts the prediction of an instance in the dataset. The Staked plot depicts the predictions for all samples in the dataset in a single plot by rotating the force plots of each instance by 90 degrees and stacking them vertically next to one another based on their clustering similarity. Each x-axis position represents an instance of the data, while each y-axis position represents the baseline prediction. The red SHAP values result in a higher prediction, whereas the blue SHAP values result in a lower prediction, and the values at the top of the vertical axis indicate a higher probability of being classified as class 1, whereas the values at the bottom of the vertical axis indicate a higher probability of being classified as class 0. Fig. 8 shows the model stacked SHAP force plot. Higher values on the vertical axis denote a greater likelihood of heart disease risk, whereas lower values denote a lesser likelihood of heart disease. Features that appear in red raise the model score (to the top), while features that appear in blue lower it (to the bottom).

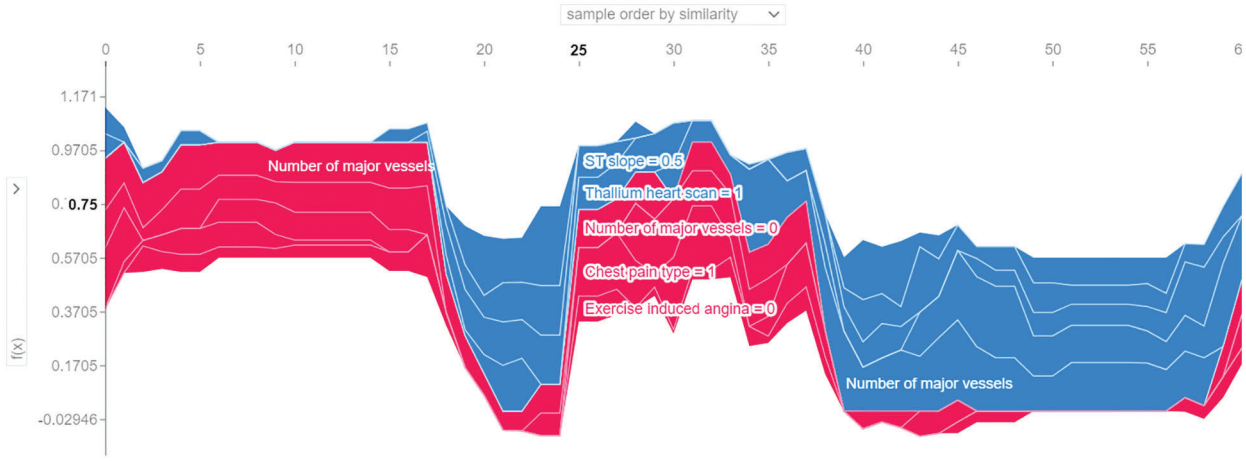


Figure 8: Stacked SHAP force plot

5 Conclusion

Heart disease is a major cause of death, and early detection can help prevent the disease's progression. Consequently, the objective of this study was to investigate the potential of quantum machine learning for predicting the risk of cardiovascular disease by developing an ensemble learning model based on quantum machine learning algorithms. The Bagging-QSVC model involves randomly subdividing the dataset into smaller subsets and modelling each subset using QSVC. Afterwards, an ensemble was formed using the majority voting method. On the Cleveland dataset, the proposed ensemble model achieved a higher classification accuracy of 90.16%, compared to 88.52%, 86.84%, and 85.25% for QSVC, QNN, and VQC, respectively, and 85.24% for both SVC and ANN. Accordingly, the various performance measures and ROC curves showed that the proposed model performed better than other machine learning models. According to the findings of this study, quantum classifiers are more effective than classical classifiers. Furthermore, results showed also that ensemble learning models can enhance the performance of quantum classifiers. In addition to predicting and diagnosing other cardiovascular diseases and aiding in medical decision-making, the model proposed in this study can be used for the early prediction of heart disease risk to avoid any serious consequences, as well as for predicting and diagnosing other cardiovascular diseases.

Acknowledgement: The authors would like to acknowledge the Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2022R196), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Data Availability: The data is publicly available at https://archive.ics.uci.edu/ml/datasets/heart+disease (accessed on 8 Feb 2022).

Funding Statement: This work is supported by Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2022R196), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] EIT Health and McKinsey & Company, "Transforming healthcare with AI," 2020. [Online]. Available: https://eithealth.eu/wp-content/uploads/2020/03/EIT-Health-and-McKinsey_Transforming-Healthcare-with-AI.pdf.

[2] K. Raza, Improving the prediction accuracy of heart disease with ensemble learning and majority voting rule. In: *U-Healthcare Monitoring Systems*, 1st. ed., vol. 1. New Delhi, India: Elsevier Inc, pp. 179–196, 2019.

[3] IBM Institute for Business Value, "Exploring computing quantum use cases for healthcare," 2020. [Online]. Available: https://www.ibm.com/thought-leadership/institute-business-value/report/exploring-quantum-financial#.

[4] Y. Kumar, A. Koul, P. S. Sisodia, J. Shafi, K. Verma *et al.,* "Heart failure detection using quantum-enhanced machine learning and traditional machine learning techniques for internet of artificially intelligent medical things," *Wireless Communications and Mobile Computing*, vol. 2021, no. 1, pp. 1–16, 2021.

[5] World Health Organization, "Cardiovascular diseases," 2020. [Online]. Available: https://www.who.int/health-topics/cardiovascular-diseases#tab=tab_1.

[6] X. Gao, A. A. Ali, H. S. Hassan and E. M. Anwar, "Improving the accuracy for analyzing heart diseases prediction based on the ensemble method," *Complexity*, vol. 2021, pp. 1–10, 2021.

[7] I. D. Mienye, Y. Sun and Z. Wang, "An improved ensemble learning approach for the prediction of heart disease risk," *Informatics in Medicine Unlocked*, vol. 20, no. 8, pp. 1–5, 2020.

[8] C. B. C. Latha and S. C. Jeeva, "Improving the accuracy of prediction of heart disease risk based on ensemble classification techniques," *Informatics in Medicine Unlocked*, vol. 16, pp. 1–9, 2019.

[9] P. Anuradha and V. K. David, "Feature selection and prediction of heart diseases using gradient boosting algorithms," in *Int. Conf. on Artificial Intelligence and Smart Systems (ICAIS 2021)*, Coimbatore, India, pp. 711–717, 2021.

[10] M. N. Uddin and R. K. Halder, "An ensemble method based multilayer dynamic system to predict cardiovascular disease using machine learning approach," *Informatics in Medicine Unlocked*, vol. 24, no. 7, pp. 1–19, 2021.

[11] A. Rahim, Y. Rasheed, F. Azam, M. W. Anwar, M. A. Rahim *et al.,* "An integrated machine learning framework for effective prediction of cardiovascular diseases," *IEEE Access*, vol. 9, pp. 106575–106588, 2021.

[12] F. Ali, S. El-Sappagh, S. M. R. Islam, D. Kwak, A. Ali *et al.,* "A smart healthcare monitoring system for heart disease prediction based on ensemble deep learning and feature fusion," *Information Fusion*, vol. 63, pp. 208–222, 2020.

[13] R. Das and A. Sengur, "Evaluation of ensemble methods for diagnosing of valvular heart disease," *Expert Systems with Applications*, vol. 37, no. 7, pp. 5110–5115, 2010.

[14] D. Mehanović, Z. Mašetić and D. Kečo, "Prediction of heart diseases using majority voting ensemble method," in *IFMBE Proc.*, Banja Luka, Bosnia & Herzegovina, pp. 491–498, 2020.

[15] M. Schuld and F. Petruccione, "Quantum ensembles of quantum classifiers," *Scientific Reports*, vol. 8, no. 1, pp. 1–12, 2018.

[16] D. Maheshwari, B. G. Zapirain and D. S. Soso, "Machine learning applied to diabetes dataset using quantum versus classical computation," in *IEEE Int. Symp. on Signal Processing and Information Technology (ISSPIT 2020)*, Louisville, KY, USA, pp. 1–6, 2020.

[17] G. Rebala, A. Ravi and S. Churiwala, "Machine learning definition and basics," in *An Introduction to Machine Learning*. Vol. 975. Cham, Switzerland: Springer Nature,1, pp. 1–16, 2019.

[18] M. Mohri, A. Rostamizadeh and A. Talwalkar, "Introduction," in *Foundations of Machine Learning*, 2nd. ed., vol. 1. Cambridge, Massachusetts: The MIT Press, pp. 1–7, 2018.

[19] J. D. Kelleher, B. M. Namee and A. D'Arcy, "Machine learning for predictive data analytics," in *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, And Case Studies*, 2. nd ed., vol. 1. Cambridge, Massachusetts: The MIT Press, pp. 3–21, 2020.

[20] P. Wittek, Quantum computing. In: *Quantum Machine Learning What Quantum Computing Means to Data Mining*, 1. st ed., vol. 34. Sweden: Elsevier Inc, pp. 41–52, 2014.

[21] C. Zhang and Y. Ma, "Ensemble learning," in *Ensemble Machine Learning Methods and Applications*. Vol. 1. USA: Springer, pp. 1–34, 2012.

[22] A. Dixit, "Introduction to ensemble learning," in *Ensemble Machine Learning*. Vol. 1. Birmingham, UK: Packt Publishing Ltd., pp. 32–55, 2017.

[23] S. Pattanayak, "Introduction to quantum computing," in *Quantum Machine Learning with Python*. Vol. 1. Bangalore, India: Apress, pp. 1–43, 2021.

[24] S. Ganguly, "Rise of the quantum machines: fundamentals," in *Quantum Machine Learning: An Applied Approach*. Vol. 1. Ashford, UK: Apress, pp. 1–39, 2021.

[25] A. Jhanwar and M. J. Nene, "Enhanced machine learning using quantum computing," in *2nd Int. Conf. on Electronics and Sustainable Communication Systems (ICESC 2021)*, Coimbatore, India, pp. 1407–1413, 2021.

[26] M. Senekane, Getting started with quantum information processing. In: *Hands-On Quantum Information Processing With Python*. Vol. 1. Birmingham, UK: Packt Publishing Ltd., pp. 4–10, 2021.

[27] N. Mishra, M. Kapil, H. Rakesh, A. Anand, N. Mishra *et al.,* "Quantum machine learning: A review and current status," in *Data Management, Analytics and Innovation. Advances in Intelligent Systems and Computing*. Vol. 1175. New Delhi, India: Springer, pp. 101–145, 2021.

[28] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe *et al.,* "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[29] L. Alchieri, D. Badalotti, P. Bonardi and S. Bianco, "An introduction to quantum machine learning: From quantum logic to quantum deep learning," *Quantum Machine Intelligence*, vol. 3, no. 2, pp. 1–30, 2021.

[30] UCI Machine Learning Repository, "Heart disease data set," 1988. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/heart+disease.

[31] S. K. Jeswal and S. Chakraverty, "Recent developments and applications in quantum neural network: A review," *Archives of Computational Methods in Engineering*, vol. 26, no. 4, pp. 793–807, 2019.

[32] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala *et al.,* "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, 2019.

[33] G. Abdulsalam, "Heart disease prediction using ensemble-quantum ML," 2022. [Online]. Available: https://github.com/ghada000/Heart_Disease_Prediction_Using_Ensemble-Quantum_ML.

[34] B. A. Tama, S. Im and S. Lee, "Improving an intelligent detection system for coronary heart disease using a two-tier classifier ensemble," *BioMed Research International*, vol. 2020, pp. 1–10, 2020.

[35] M. A. Alim, S. Habib, Y. Farooq and A. Rafay, "Robust heart disease prediction: a novel approach based on significant feature and ensemble learning model," in *3rd Int. Conf. on Computing, Mathematics and Engineering Technologies (iCoMET 2020)*, Sukkur, Pakistan, pp. 1–5, 2020.

[36] S. Lundberg and S. Lee, "A Unified approach to interpreting model predictions," in *31st Conf. on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA, pp. 1–10, 2017.

Feature Importance and Explainability in Quantum Machine Learning

Luke Power[1], Krishnendu Guha[2]
School of Computer Science and Information Technology, University College Cork, Ireland
Email: 120371316@umail.ucc.ie[1] (corresponding author), kguha@ucc.ie[2]

***Abstract*—Many Machine Learning (ML) models are referred to as black-box models, providing no real insights into why a prediction is made. Feature importance and explainability are important for increasing transparency and trust in ML models, particularly in settings such as healthcare and finance. With quantum computing's unique capabilities, such as leveraging quantum mechanical phenomena like superposition, this can be combined with ML techniques to create the field of Quantum Machine Learning (QML), and such techniques may be applied to QML models. This article explores feature importance and explainability insights in QML compared to Classical ML models. Utilizing the widely recognized Iris dataset, classical ML algorithms— SVM and Random Forests, are compared against hybrid quantum counterparts, implemented via IBM's Qiskit platform: the Variational Quantum Classifier (VQC) and Quantum Support Vector Classifier (QSVC). This article aims to provide an in-depth comparison of the insights generated in ML by employing permutation and leave-one-out feature importance methods, alongside ALE (Accumulated Local Effects) and SHAP (SHapley Additive exPlanations) explainers.**

***Index Terms*—ML, QML, Feature Importance, Explainability**

I. Introduction

This article aims to provide an understanding of quantum computing, and how it takes quantum mechanical phenomenon and integrates with classical machine learning to facilitate quantum machine learning (QML), alongwith explainability.
Given how many current ML models are considered 'black box' or 'opaque models', meaning that there is no real way to understand how or why an output is generated, there is a rise in the necessity to implement ways of explaining a models inner workings and rationalising its outputs. Currently, there are several methodologies to determine a model's most important features and explainability of individual predictions, and this article seeks to apply these methods to QML models and make an in-depth comparison of the results.

A. Objectives

The main objective of this article is to ascertain what new inferences could be made about a dataset via QML when compared to ML and attempt to explain how these differences come about. For this article, we are interested in using classical data, but quantum algorithms, in another way; the top-right quadrant of the image 1. This was achieved by using a common ML method, building an equivalent QML model, and making inferences about the data by measuring feature importance - i.e. how important a feature is in making an accurate classification by using feature omission and permutations to quantify the differences, and also applying explainability methods to the models, which can provide greater insights into a singular prediction.
As of writing, this paper is the first to provide a comprehensive comparison of the results of applying machine learning explainability to QML on multi-class data.
This article will also delve into the theory behind quantum computing and the implementations of some basic quantum computing models to give clear demonstrations of the 'quantum advantage'.

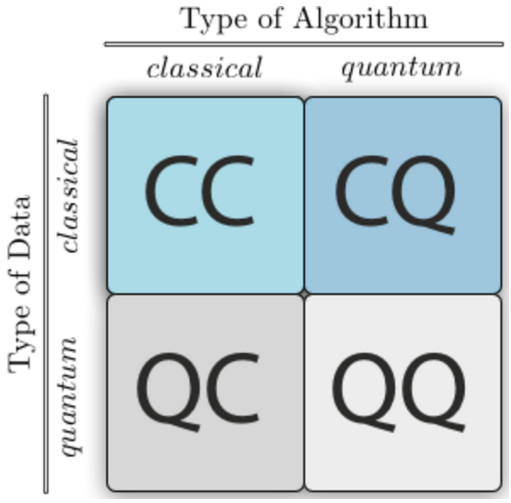


Fig. 1: Classical Data on Quantum Algorithms. We are using Classical data with Quantum algorithms. Image adapted from [1].

B. Work Overview

The rest of the Introduction will provide a general overview of the article; the main objectives as well as some theoretical background on the main concepts of quantum computing and ML and a brief overview of the primary technology and tools that were used over the course of completing this article.
Chapter Two, Theoretical Background, will provide a more in-depth exploration of the machine learning models used, and explore the theory and mathematics behind implementing the QML models. This chapter will also demonstrate the implementation of quantum algorithms that provide some examples of quantum advantage in some computing tasks. The end of this chapter will also contain a literature review of the

current research and papers in the field.
Chapter Three, Implementation goes through the actual implementation of the models on the Iris dataset and provides a comparison of the models' predictive performance, in the subsequent chapter, Results, the results of the implementations of the Feature Importance and Explainability methods are revealed, analysed and compared.
The Conclusions chapter will provide a personal reflection on the undertaking of this article, reviewing the experience in completing this article, how it was completed along with the future plans of the article.

C. Overview of Methodology

The approach to this article involved partaking in a course on quantum computing, familiarization with concepts of quantum theory, utilization of IBM's Qiskit platform which provides access to quantum hardware and building the necessary circuits and models. After several exploratory scripts and implementing algorithms which can provide proven demonstrations of the quantum advantage. After this, the initial QML models were built. The basic models were then expanded with different permutations of optimisers and ansatzes, to gain a level of accuracy comparable to the classical counterparts. After some initial model comparisons, we can then implement several methods of feature importance and explainability to both sets of models and compare the results. An overview of this process is displayed in the flow chart 2

II. Background

A. Concepts and Technologies

Machine Learning (ML) is a subsection of computer science and AI that utilizes statistical models to make inferences about the underlying patterns in a dataset and build algorithms that can make accurate predictions or classifications when presented with new, unseen data, without the need of explicit programming. [2]

Supervised learning, where models are trained on a labelled dataset, learning to predict outcomes for new data based on this training is the most widely used form of machine learning, and is the underlying method that was followed in the course of this article. Supervised learning involves feeding a model some labelled data which it will train its parameters on to minimise some cost function.

B. Qubits and Superposition

QML is a branch of quantum computing that uses phenomena seen in quantum physics, such as superposition and entanglement which allows for multiple computations to be conducted simultaneously.[3] The '**qubit**' is the quantum equivalent of the bit in classical computing. What is special about the qubit is that unlike bits, which can only ever have the states one or zero, a qubit can be in both states simultaneously. This phenomenon is known as **superposition**.

Schrodinger's Cat is a thought experiment to describe the idea of superposition, which is explained by thinking of a cat that is placed in a container containing a radioactive element that has a 50% probability of releasing and killing the cat. It is said that until the box is opened and the state of the cat is observed (the state being either alive or dead), the cat is said to be in a state of superposition of both alive and dead at the same time[4]. A more feline-friendly way to think about superposition is a coin that has been tossed in the air and is 'simultaneously heads and tails'. The ability to have data in superposition can allow for quantum algorithms to perform computations on data in multiple states simultaneously. This can lead to exponential speedups in computation and the ability to perform tasks on highly complex data.

C. Feature Importance & Explainable ML

Although ML models have provided great utility to many people and companies across a multitude of sectors, many of them are often regarded as 'black boxes'. The data is passed in and a result is returned, the actual inner workings of the model are not known to the user. This leads to an investigation into 'Explainable AI' XAI and 'Explainable ML', XML, which aims to shine a light on the inner workings of a model. Reasons for this could be in a denied loan application, and the ability to explain clearly why the model decided to deny the application, to a doctor or patient, explaining why a model predicted a certain medical diagnosis, or perhaps what settings are causing a machine to malfunction. Good explainability can also be utilised by data scientists to provide sanity checks or debugging on the model (what if someone's name was the reason behind the denied loan?). [5] [6]

D. Methods for Assessing Feature Importance:

- **Leave-One-Out (LOO) Feature Importance**: This method assesses the impact on model performance when each feature is individually left out from the model. A significant decrease in performance upon leaving out a feature implies increased importance.
- **Permutation Feature Importance**: This method assesses feature importance by measuring the increase in the model's prediction error after permuting the feature's values, which breaks the relationship between the feature and the outcome.[7]
- **SHapley Additive exPlanations (SHAP) Values**: Based on cooperative game theory, and Shapely values which in short, calculate how the winnings should be split among the team players, based on predictions on the outcomes had the players played independently or some permutations of them. This can be applied to ML through SHAP [8] values to explain the prediction of an instance by computing the contribution of each feature to the prediction. SHAP values offer both global and individual explanations of feature importance. [9]
- **Accumulated Local Effects (ALE)**: Used to interpret the influence of features in a model, especially in the context of prediction tasks. Unlike other feature importance metrics like in a Partial Dependency Plots, ALE values focus on the local effects of features, by splitting up the feature space into a number of 'windows' and calculating

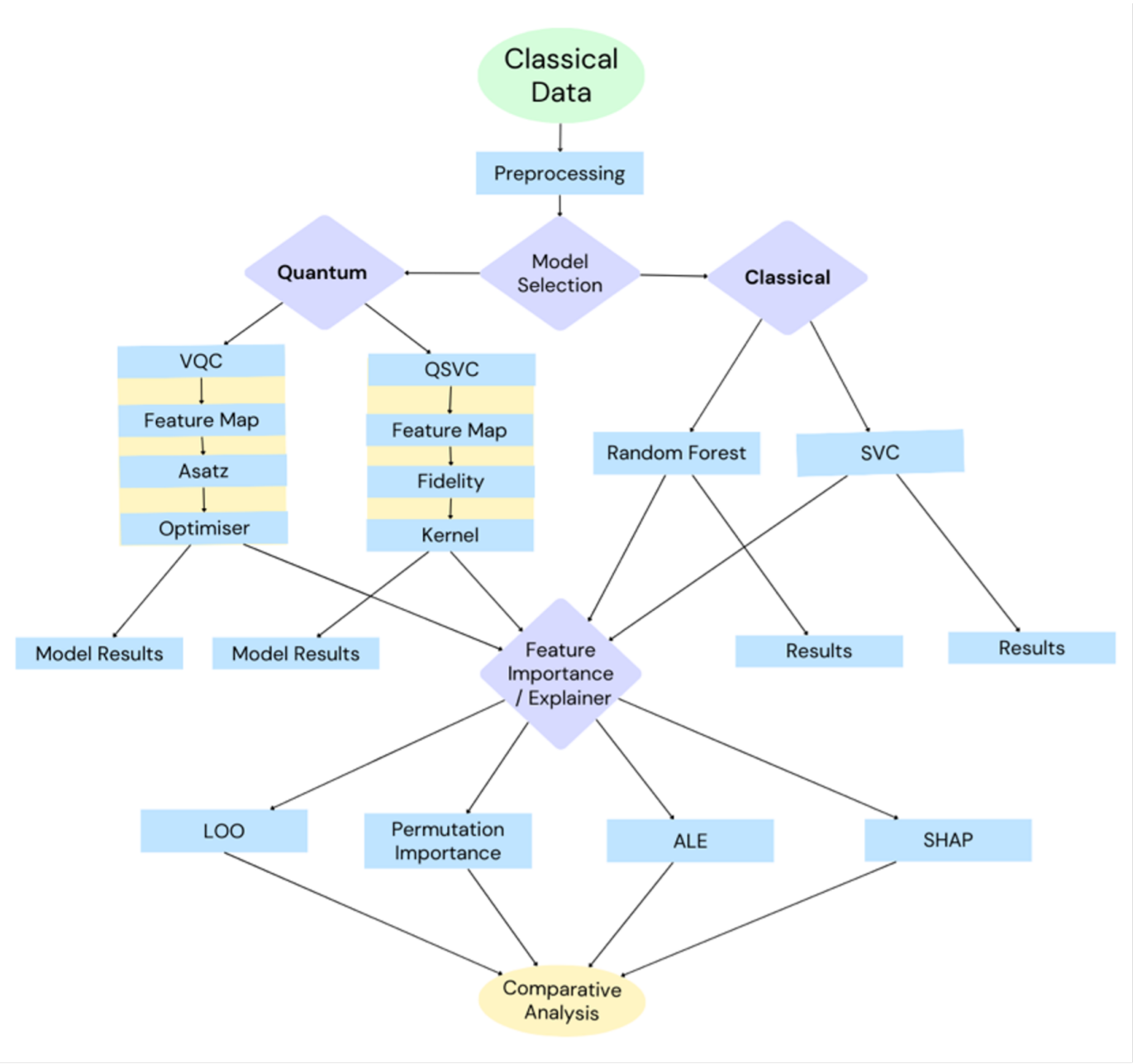


Fig. 2: Data Flow of the article, centring around the application of Feature Importance and Explainability.

the average change in the model's predictions over the intervals of the feature's values. [10]

The main software and package dependencies can be found in table I.

III. Theoretical Background

A. Supervised Machine Learning

Supervised ML consists of first gathering the labelled data, which the ML model will be trained and tested on. For classification tasks, this often consists of collecting data and filling in a number of feature vectors and a classifying label. Examples of this could be spam mail, where the features could consist of time, sender, and content keywords, among other possible predictors, and the classification label could be a simple 1 = spam, 0 = not spam. Another example could be a collection of photos, along with labels, classifying the image subjects, for example, differentiating cats and dogs. The feature vector in this regard would be the values of the image pixels' RGB and opacity values. [2]

There are a number of ML algorithms, but the one used for this article was the SVM.

1) Support Vector Machines: Support Vector Machines are a powerful ML tool that can be used for both regression and classification but is more often used in classification tasks. An

TABLE I: article Git Dependency

Package Name	Version	License
attrs	21.4.0	MIT
brotlipy	0.7.0	MIT
configparser	6.0.1	MIT
cryptography	37.0.1	Apache-2.0 and others
cython	0.29.32	Apache-2.0
dl	0.1.0	BSD-2-Clause
docutils	0.18.1	BSD-2-Clause
htmlparser	0.0.2	BSD-2-Clause
importlib-metadata	7.0.1	Apache-2.0
ipython	8.12.3	BSD-2-Clause and BSD-3-Clause
ipywidgets	7.6.5	BSD-2-Clause and BSD-3-Clause
jinja2	3.1.3	BSD-2-Clause and BSD-3-Clause
jnius	1.1.0	MIT
keyring	23.4.0	MIT
lockfile	0.12.2	MIT
lxml	4.9.1	BSD-2-Clause and BSD-3-Clause
matplotlib	3.5.2	PSF-2.0
numpy	1.23.5	BSD-2-Clause
ordereddict	1.1	MIT
pandas	1.4.4	BSD-2-Clause and BSD-3-Clause
pillow	10.2.0	HPND
pillow	9.2.0	HPND
protobuf	4.25.3	BSD-3-Clause
pyopenssl	22.0.0	Apache-2.0
pyopenssl	24.0.0	Apache-2.0
qiskit	0.44.2	
qiskit-aer	0.12.2	
qiskit-ibmq-provider	0.20.2	Apache-2.0
qiskit-terra	0.25.2.1	
railroad	0.5.0	MIT
scikit-learn	1.4.0	BSD-2-Clause and BSD-3-Clause
seaborn	0.11.2	BSD-2-Clause and BSD-3-Clause
sphinx	5.0.2	BSD-2-Clause
thread	0.1.3	BSD-2-Clause and BSD-3-Clause
toml	0.10.2	MIT
tornado	6.1	Apache-2.0
xmlrpclib	1.0.1	
zipp	3.8.0	MIT

SVM works by finding an optimal hyperplane that separates the data points with the best probability of correctly separating classes where there is overlap by maximising the margins of the hyperplane.

The separating hyperplane $f(x)$, defined by:

$$f(x) = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \ldots + \beta_p X_{pi}$$

for a given point (x_i, y_i) is so that:

$$f(x_i) = \begin{cases} > 0, & \text{if } y_i = 1 \\ < 0, & \text{if } y_i = 0 \end{cases}$$

And we can use this as a classification rule for the data; classifying x^* with respect to $(sign)f(x^*)$

The objective of the SVM is to calculate the optimal hyperplane to separate the data defined by

$$M = argmax_{\{\beta,\epsilon,m\}}\{y_i(\beta_0+\beta_1 x_{1i}+\ldots+\beta_p x_{pi}) \geq m(1-\epsilon_i\}$$

Subject to:

$$\sum \beta_j^2 = 1, \quad \epsilon_i \geq 0, \quad \sum_{i=1}^{N} \epsilon_i \leq c$$

For some tuning parameter $c > 0$, also known as a 'budget' which is the allowed rate for margin violation, and ϵ_i or 'slack variable', where:

$$\epsilon_i > 0 \quad \text{point is classified on the wrong side of the margin,} \tag{1}$$

$$\epsilon_i > 1 \quad \text{point is classified on wrong side of the hyperplane.} \tag{2}$$

In real life, data is often non-linearly separable, and the formula for the hyperplane needs to be expanded to account for non-linearity by incorporating functions known as kernels on the predictors, as seen in figure 3.

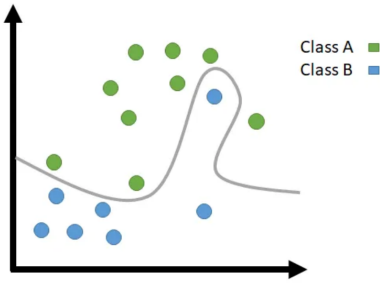


Fig. 3: Non-Linear Support Vector Machine. Image adapted from [11].

This introduction of kernel function is known as the 'kernel trick', which updates the formulas for the hyperplane as follows:

$$M = argmax_{\{\beta,\epsilon,m\}}\{y_i(\beta_0+\sum_{j=1}^{P}(\beta_j x_{i_i}+\alpha_j x_{j_i}^2) \geq m(1-\epsilon_i\}$$

For all i, subject to:

$$\sum \epsilon_i \leq c, \quad \epsilon_i \geq 0, \quad c > 0, \quad \sum \beta_j^2 + \sum_j \alpha_j^2 = 1$$

We can then establish that an SVM uses kernel functions

$$f(x) = \sum_{h=1}^{H} \alpha_h K_h(x, x') + \alpha_0$$

$$= \alpha_0 + \sum_{i \in \delta} \alpha_i k(x, x_i)$$

Where δ is the set of Support Vectors. [12]

2) Random Forest: A random forest is an ensemble model that samples from a number of different decision trees and aggregates the output of the trees. This is due to the fact that while growing a tree, the optimal cutoff is on a split-by-split basis. This greedy approach can result in a sub-optimal tree. This strategy aims to reduce this effect, reduce variability and to preserve a low bias. In order to aggregate the results from

the trees requires resampling. For example: using a bootstrap resampling scheme:

$$\hat{f}_B^*(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_b^*(x)$$

Where B represents the number of bootstrap samples and $\hat{f}_b^*(x)$ represents an estimate from each bootstrap sample. The notation $\hat{f}_B^*(x)$ denotes the averaged estimate across all bootstrapped samples.[13]

B. Quantum Machine Learning

QML can enhance machine learning by leveraging quantum mechanics to identify patterns in data that classical computing struggles with. Research suggests that quantum computers might excel in tasks like pattern recognition and optimization due to their ability to process complex, counter-intuitive patterns. Although some literature indicates that quantum-enhanced machine learning could unlock new paradigms in data analysis and problem-solving, the biggest advantage of quantum computing is the speed-up that can be expected, particularly through quantum computers' ability to handle large matrix operations. [14][15]

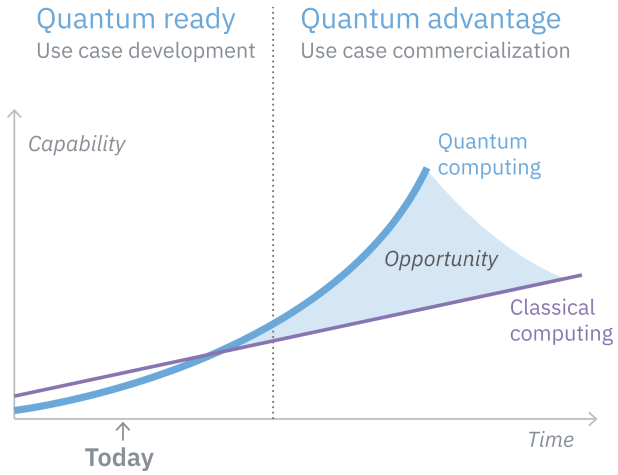


Fig. 4: Quantum Advantage. Image adapted from [16].

The core advantage of quantum computing lies in its ability to process information in ways that classical systems cannot match, offering exponential speed-ups for certain types of problems. For example, quantum algorithms have been demonstrated to outperform classical counterparts in oracle-based problems, showing that a significant quantum advantage emerges even in existing noisy systems.[17] Furthermore, quantum technology can revolutionize how we learn from experiments, as shown in a study where quantum machines learned from exponentially fewer experiments than required by conventional means, leading to dramatic reductions in the number of necessary experiments.[15] Quantum machine learning also benefits from the ability to process atypical patterns produced by quantum systems, offering potential speed-ups in learning tasks.[14] Additionally, the use of quantum-enhanced feature spaces in machine learning can provide advantages in solving classification problems where classical feature spaces become computationally prohibitive [18].

1) The Qubit & Superposition: As mentioned in the introduction, quantum computing utilises qubits which differ from classical bits by being able to exist as both zero or one simultaneously in a state called superposition. This means that n bits can represent n^2 possible states, but qubits can be in 2^n states simultaneously. Superposition allows operations to be applied on all the possible states simultaneously, and combined with the likes of interference effects which can cancel out 'wrong' results can provide results that cannot be achieved with classical computing. [19]. Quantum computing also relies on linear algebra to mathematically represent states as vectors. For example, the classical bit's states of zero or one would be represented as the vectors (using Dirac notation):

$$|1\rangle = \begin{bmatrix}0\\1\end{bmatrix} \text{ and } |0\rangle = \begin{bmatrix}1\\0\end{bmatrix}$$

In quantum computing, these certain states would be referred to as the two 'pure states'. A qubit, on the other hand, would be in a combination of the two states simultaneously, in a phenomenon known as superposition taking on coefficients α and β which represent the probabilities of being in their respective states. A qubit in a state $|\psi\rangle$ would be written as;

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \quad \alpha, \beta \in C$$

Where C represents complex numbers; numbers that have an imaginary component i where $i = \sqrt{-1}$. It is not possible to measure α or β, as the observation or measurement of a qubit will cause it to collapse into one of its pure states. However since α and β are still probabilities their total magnitude must be equal to one.

$$|\alpha|^2 + |\beta|^2 = 1$$

[20]

A superpositioned state can be achieved by passing qubits through a unitary operation known as a Hadamard gate.

- Applied to $|0\rangle$, the Hadamard gate produces the state $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, an equal superposition state where the probabilities of measuring the qubit in state $|0\rangle$ or $|1\rangle$ are both 50%.

- Applied to $|1\rangle$, it produces the state $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, another equal superposition state, but with a phase difference between the $|0\rangle$ and $|1\rangle$ components.

[19]

To better understand the state of a qubit we use what is called a 'Bloch Sphere', depicted in figure 5. A qubit's state can be represented geometrically on a Bloch sphere described with the formula:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle$$

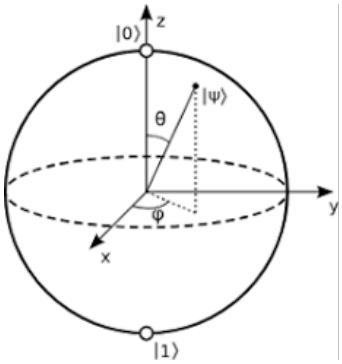


Fig. 5: 3D representation of a qubits' possible states.

In this representation, $|0\rangle$ and $|1\rangle$ are the standard basis states of the qubit. The angles θ and ϕ are real numbers, corresponding to the point on the Bloch sphere, where θ ranges from 0 to π and ϕ ranges from 0 to 2π. The state $|\psi\rangle$ is thus a point on the surface of the sphere, where the north pole corresponds to $|0\rangle$ and the south pole to $|1\rangle$. [21]

2) Variational Quantum Classifier: A VQC represents an integration of quantum computing principles into machine learning. The process of implementing a VQC involves encoding classical data to a quantum feature space via a feature map, and then using this new quantum data in the quantum circuit. This allows for classical data to be used in a quantum circuit. A VQC uses variational quantum algorithms which follow hybrid quantum-classical schemes, involving parameterised circuit and gates with parameters optimised through a classically-based optimisation loop to minimise some cost function. These steps are repeated in a quantum-classical hybrid loop that eventually terminates when the classical optimization has found optimal parameters. The typical cost function is a measurement between the actual outputs and the desired outputs for training data. [19] [22]

3) Quantum Support Vector Machines: Quantum Support Vector Machines are the quantum extension to SVMs discussed earlier by offering a natural and efficient framework for linear algebra operations, including those required for SVMs. QSVMs exploit quantum algorithms for linear systems of equations and quantum kernel estimation to calculate the inner products in a high-dimensional feature space more efficiently. This approach leverages the concept of quantum feature maps and kernels to implicitly perform these calculations in a Hilbert space represented by quantum states. What this suggests is that QSVMs' quantum kernels are expected to do better than classical if they are hard to estimate classically.[19]

The quantum kernel is created by mapping a classical feature vector $\vec{x}$ to a Hilbert space using a quantum feature map $\phi(\vec{x})$.

$$K_{ij} = |\langle \phi(\vec{x}_i) | \phi(\vec{x}_j) \rangle|^2$$

where K_{ij} is the kernel matrix, $\vec{x}_i, \vec{x}_j$ are n dimensional inputs $\phi(\vec{x})$ is the quantum feature map $|\langle a|b\rangle|^2$ denotes the overlap of two quantum states a and b.[24]

C. Encoding Classical Data

One of the first steps in building the QML model is encoding (or 'embedding) the classical data into a quantum state. This step allows us to utilise phenomena like superposition and entanglement on the embedded classical data. [19] there are a number of ways of implementing this;

- **Basis Encoding:** Basis (or computational) encoding maps each bit of classical binary data to the corresponding quantum state. A classical bit value of 0 or 1 is represented by the quantum state $|0\rangle$ or $|1\rangle$ respectively. [25]
- **Amplitude Encoding** Where data vector $\vec{x}$ in $\mathbb{R}^N$ is normalised and then encoded into the amplitudes of a quantum state $|\psi\rangle$, represented as:

$$|\psi\rangle = \sum_{i=0}^{N-1} x_i |i\rangle ,$$

  where $|i\rangle$ denotes the computational basis states. This method allows for encoding N-dimensional data into $\log_2(N)$ qubits. [26]
- **Angle Encoding:** Utilizes the values of classical data to define the angles in quantum rotations about the Bloch Sphere [19]. For instance, a real number θ is encoded as $|\psi(\theta)\rangle = R(\theta)|0\rangle + R(\theta)|1\rangle$, letting R represent the rotation procedure. For example a rotation about the Z axis would be represented as: The rotation matrix around the Z-axis is given by:

$$R_z(\theta) = e^{-i\theta Z/2} = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}$$

  [25]

1) ZZ-Feature Map: The ZZFeatureMap, is a second-order Pauli-Z evolution circuit,[27] The Pauli Expansion circuit is a data encoding circuit that transforms input data $\vec{x} \in \mathbb{R}^n$, where n is the number of feature dimensions, as

$$U_\Phi(\vec{x}) = \exp\left(i \sum_{S \in I} \phi_S(\vec{x}) \prod_{i \in S} P_i \right).$$

Here, S is a set of qubit indices that describes the connections in the feature map, I is a set containing all these index sets, and $P_i \in \{I, X, Y, Z\}$. Per default the data-mapping ϕ_S is

$$\phi_S(\vec{x}) = \begin{cases} x_i & \text{if } S = \{i\} \\ \prod_{i \in S}(\pi - x_i) & \text{if } |S| > 1 \end{cases}$$

[18]

The construction of the ZZFeatureMap involves applying a series of controlled rotation gates that correlate qubit states in a manner that reflects the underlying structure of the classical data.

7 represents the circuit of the ZZFeatureMap where where ϕ is a classical non-linear function, which defaults to $\phi(x) = x$ if $\phi(x, y) = (\pi - x)(\pi - y)$.[27]

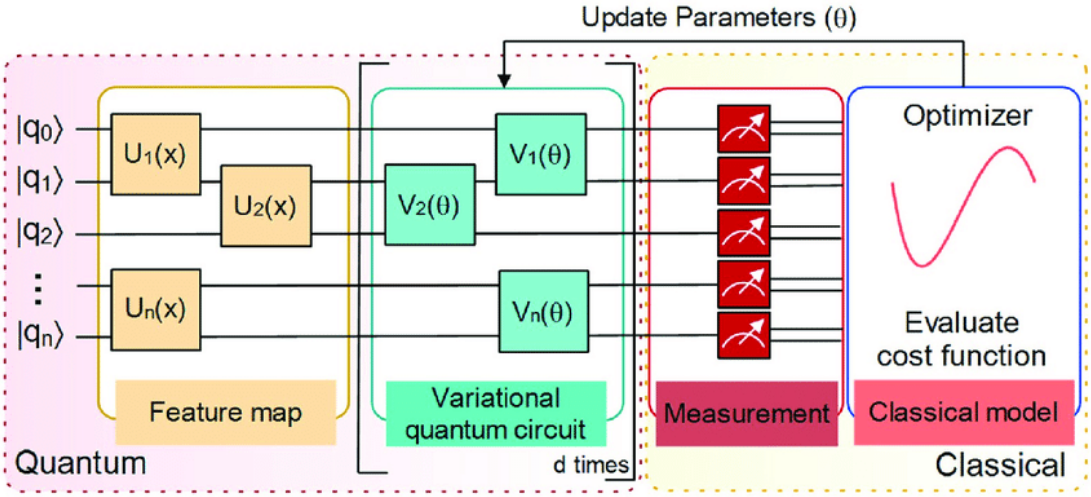


Fig. 6: Schematic representation of a variational quantum circuit (VQC). Image from [23]

.

Fig. 7: 3 qubits and 1 repetition and linear entanglement circuit.

D. Feature Importance & Explainability

Feature importance plays a pivotal role in the domain of ML, serving as a bridge to understanding the inner workings of black box predictive models, which are often hidden from the user. This concept highlights the relative significance of each input feature in contributing to the model's ability to make accurate predictions. The importance of being able to understand a model spans various aspects of development and application, emphasizing the necessity for interpretability, transparency, and trust in machine learning outcomes.

In many critical sectors such as healthcare, finance, and more, decisions made by machine learning models can have profound implications. [6] For example in healthcare, a model might be used to predict patient outcomes based on various clinical parameters, and this outcome needs to be fully understood by not only the doctor but also to be explained to the patient. Understanding which features most significantly influence the predictions can provide insights into diseases, patient management, and treatment planning. Similarly, in finance, models predicting credit risk can benefit from clear insights into which factors most influence a person's likelihood of qualifying for a loan, aiding in fair and transparent decision-making.

The emphasis on feature importance thus directly ties into the goal of XAI. XAI seeks to peel back the layers of complex ML models to offer human-understandable explanations for their predictions.

With the increasing deployment of ML models in sensitive and impactful domains, regulatory bodies are mandating greater transparency and explainability in automated decision-making systems. For instance, the European Union's General Data Protection Regulation (GDPR) introduces the right to explanation, where individuals can ask for explanations of automated decisions that affect them. Feature importance metrics can help in fulfilling such regulatory requirements by providing a basis for explaining decisions made by ML models. [28]

From a technical perspective, understanding feature importance is integral to refining and optimizing ML models. By identifying and focusing on the most features, data scientists can streamline models, reducing their complexity without sacrificing performance. This process of feature selection helps in mitigating overfitting, enhancing the model's ability to generalize to unseen data. Moreover, it can lead to more efficient models by reducing the dimensionality of the data, thereby lowering computational costs and improving runtime efficiency.

For machine learning solutions to be widely adopted, especially in high-stakes domains, it is crucial for end-users and stakeholders to trust the decisions made by these systems. Feature importance metrics facilitate this trust by offering transparency in the decision-making process. When users understand why a model makes a certain prediction, their confidence in the system's reliability and fairness should increase. This trust is paramount, especially in critical applications, where the cost of errors is high.

Feature importance and explainability are cornerstones of ethical and effective machine learning practice. It not only fosters trust, transparency, and ethical decision-making but also contributes to the technical robustness and efficiency of predictive models. As machine learning continues to permeate

various sectors of society, the role of feature importance in ensuring responsible AI cannot be overstated. This understanding is achieved through various methods, including:

- **Model Interpretability and Transparency**: By understanding which features significantly influence the model's output, stakeholders can gain insight into the model's decision-making process, leading to greater trust.
- **Improved Model Performance**: Identifying and focusing on important features can lead to more efficient models by reducing dimensionality, which can, in turn, lower computational costs and mitigate the risk of overfitting.
- **Feature Engineering**: Knowing which features are important can guide the feature engineering process, where new features are created and redundant or irrelevant features are removed.
- **Domain Insight**: Feature importance can reveal unexpected relationships between features and the target variable, providing new insights into the domain area being studied.
- **Regulatory Compliance**: In many regulated industries, the ability to explain decisions made by machine learning models is a legal requirement. [5] [6] .

1) Leave-One-Out (LOO) Feature Importance: Leave-one-out feature importance is an intuitive approach where the importance of a feature is determined by the impact on model performance when that feature is left out of the training process. In this method, the model is trained multiple times, each time leaving out a different feature. The change in model performance (such as accuracy or F1 score) indicates the importance of the omitted feature. If the performance drops significantly without a particular feature, it suggests that the feature is important for the model to make accurate predictions. However, this is a very time-consuming method as it requires retraining a model for each feature to be tested.

2) Permutation Importance: Permutation importance ranks the importance of a feature by shuffling the values of the column in question and recalculating the accuracy of the model. The permutation importance algorithm 1 shows how it is implemented. Permutation Importance breaks the relationship between the feature and the true outcome, thus the change in model error after permutation is indicative of the feature's predictive power. The main advantage of permutation importance is that it can be applied to any model and is less likely to be biased towards features with a high number of categories.[7]

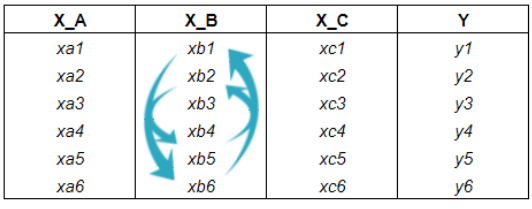


Fig. 8: How permutation importance works. Image from [29]

Algorithm 1 Permutation Importance

Require: fitted predictive model m, tabular dataset D (training or validation)

Compute the reference score s of the model m on data D
for each feature j in columns of D **do**
  for each repetition $k = 1, \ldots, K$ **do**
    Randomly shuffle column j of dataset D to generate a corrupted version of the data named $\widetilde{D}_{k,j}$
    Compute the score $s_{k,j}$ of model m on corrupted data $\widetilde{D}_{k,j}$
  end for
  Compute importance i_j for feature f_j defined as: $i_j = s - \frac{1}{K}\sum_{k=1}^{K} s_{k,j}$
end for

3) SHAP Values: The SHAP method uses game theory to explain a machine learning model's output. Through the use of the traditional Shapley values from game theory and their related extensions, it links local explanations for optimal credit allocation.SHAP values give an overall measure of feature importance but also show the direction and magnitude of a feature's effect on individual predictions [8]. This method assigns each feature an importance value for a particular prediction by comparing what a model predicts with and without the feature. The calculation considers all possible combinations of features, which ensures fair attribution since it accounts for interaction effects between features. The formula can take the form:

$$\mathbb{E}[f(X) \,|\, do(X_S = x_S)]$$

This tells us how the model would behave if the inputs were changed, and also because it is much easier to compute as computing SHAP values are generally NP-hard [9].

4) Accumulated Local Effects: Accumulated Local Effects plots calculate the local impact of a feature by evaluating the change in predictions over finely segmented windows of the feature space. This approach allows for a more accurate representation of the feature's effect, taking into account the natural interactions and dependencies among features. ALE plots achieve this by focusing on the differences in model predictions within these intervals, thus isolating the specific contribution of each feature to the prediction outcome. ALE plots average the changes in the predictions and accumulate them over the grid. This algorithm was introduced to improve upon the Partial Dependency Plots method, which can introduce problems when features are correlated [10].

$$\hat{f}_{S,ALE}(x_S) = \int_{z_{0,S}}^{x_S} \mathbb{E}_{X_C|X_S=x_S}\left[\hat{f}^S(X_S, X_C)|X_S = z_S\right] dz_S - \text{constant}$$

$$= \int_{z_{0,S}}^{x_S} \left(\int_{X_C} \hat{f}^S(z_S, X_C) d\mathbb{P}(X_C|X_S = z_S)\right) dz_S - \text{constant}$$

The ALE method calculates the differences in predictions by replacing the feature of interest with grid values z. The

difference in prediction is the effect the feature has for an individual instance in a certain interval. The sum on the right adds up the effects of all instances within an interval, which appears in the formula as the neighbourhood $N_j(k)$. We divide this sum by the number of instances in this interval to obtain the average difference of the predictions for this interval. This average in the interval is what the term *Local* in the name ALE covers. The sum symbol on the left means that we accumulate the average effects across all intervals. The uncentered ALE value of a feature that lies in the third interval, for example, is the sum of the effects of the first, second, and third intervals. This is what is reflected by the word *Accumulated* in ALE.

This effect is centred so that the mean effect is zero.

$$\hat{f}_{j,ALE}(x) = \hat{f}'_{j,ALE}(x) - \frac{1}{n}\sum_{i=1}^{n} \hat{f}'_{j,ALE}\left(x_j^{(i)}\right)$$

[10][30]

IV. Related Works

There are several papers and resources explore the implementation of QML algorithms, including guides provided through IBM Qiskit [22] and papers such as 'Quantum variational multi-class classifier for the iris data set' [31], This paper confirmed a method implemented to check optimising combinations of ansatzs and optimisers. However, there is only a handful of papers that explore the implementation of feature importance and explainability for QML.

There are many papers and resources about machine learning, feature importance, and quantum computing and quantum machine learning, however, there is an apparent gap in combining these topics. The first paper to compare feature importance between quantum and classical machine learning models was a paper titled 'Study of feature importance for quantum machine learning models' [32] was conducted by researchers at IBM using ESPN Fantasy Football data and claims to be the first paper exploring feature importance in QML. However, this paper only investigated two methods of feature importance; Permutation importance and ALE [32]. This article expands on this paper by providing a much more in-depth report on the implementation of QML and by both exploring more methods of feature importance and also venturing into methods of explainability.

There are fewer papers exploring the area of explainable QML; 'eXplainable AI for Quantum Machine Learning' [33] Implements a variation of SHAP on basic classifiers, ranging from a single qubit classifier two a four qubit classifier, using the binary 'bars and stripes' data. However, this paper does not make any comparisons to the results from a classical counterpart.

Last year, a paper published 'Explainable heart disease prediction using ensemble-quantum machine learning approach' [34] utilised various QML methods on UCI's Heart Disease data set, with a binary classification of diagnosis or not. This paper implemented SHAP predictions but does not make clear comparisons between the classical and quantum results concerning their SHAP values or comparable metrics.

What this article does is implement classical and quantum machine learning models on the multi-class Iris dataset, which is a well-known data set used to classify flowers, and implement several methods of feature importance and explainability techniques and provide some direct comparisons between the results.

V. Implementation

We have simulated the quantum models with the Iris data set with Qiskit's SDK.

A. The Iris Dataset

The Iris dataset, introduced by Ronald Fisher in 1936, is a foundational dataset used in statistical learning and machine learning. It comprises 150 samples from three species of Iris flowers, each described by four features: sepal length, sepal width, petal length, and petal width. The dataset's simplicity yet capability to demonstrate the effectiveness of various algorithms has made it a staple for teaching purposes. And so this dataset was chosen to be used in this article. [36]
For a clear breakdown, of the Iris dataset, refer to the table. IV.

B. Data Preprocessing

The data was normalised with scikit-learns MinMaxScaler, where features are scaled to fall from zero to one. This step was taken to ensure better comparability between the classical and quantum models, as this step was necessary to encode the data for the quantum classifiers. All the models were trained and scored on an 80% - 20% train-test split. The breakdown of the test group can be found in table III.

Plot 9 shows a pairs plot of the Iris dataset, which visualises the relationship between the features. The consistent separation of the setosa data points suggests that it will be more easily classified than versicolor and virginica, which have several overlapping points. It is also worth noting the linear structure between peal length and width which shoes the correlation between the two features, which can indicate that one of the features may be redundant.

1) VQC Implementation: The first and most basic QML model implemented was the VQC, which could be described as the quantum equivalent of a simple linear classifier. A ZZFeatureMap was used on the dataset to use the classical data in the quantum algorithm. However, to maximise the performance of the model we also needed to select what combination of ansatz and classical optimizer should be utilised.

VQC Optimisations: **Choices of Ansatz**

- **RealAmplitudes** circuit is a heuristic trial wave function used as Ansatz in chemistry applications or classification

TABLE II: Literature Comparison

Paper	SVM/C	RF	VQC	QSVC	ALE	LOO	Perm	SHAP
Study of Feature Importance for Quantum Machine Learning Models. [32]	Yes		Yes		Yes		Yes	
Quantum variational multi-class classifier for the iris data set. [31]			Yes					
Supervised learning with quantum-enhanced feature space. [18]	Yes		Yes	Yes				
eXplainable AI for Quantum Machine Learning. [33]	Yes		Yes					
Principles and practice of explainable machine learning. [5]	Yes	Yes					Yes	Yes
Explaining Quantum Circuits with Shapley Values (pre-print) [35]			Yes	Yes				Yes
Explainable heart disease prediction using ensemble-quantum machine learning approach [34]			Yes	Yes				Yes
This article	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

circuits in machine learning. The circuit consists of alternating layers of Y rotations and CX entanglements. The entanglement pattern can be user-defined or selected from a predefined set. It is called RealAmplitudes since the prepared quantum states will only have real amplitudes, the complex part is always 0. [37]

- **EfficientSU2** circuit consists of layers of single qubit operations spanned by SU(2) and CX entanglements. This is a heuristic pattern that can be used to prepare trial wave functions for variational quantum algorithms or classification circuit for machine learning.
  SU(2) stands for special unitary group of degree 2, its elements are 2×2 unitary matrices with determinant 1, such as the Pauli rotation gates. [38]

Choices of Optimizer

- **COBYLA** is a numerical optimization method for constrained problems where the derivative of the objective function is not known. [39]
- **SLSQP** minimizes a function of several variables with any combination of bounds, equality and inequality constraints. SLSQP is ideal for mathematical problems for which the objective function and the constraints are twice continuously differentiable. [40]

These were combined and iterated from one to four repetitions of the ansatz. As seen in 10, SLSQP and EfficentSU2 have the best overall performance, with an initial score of 90%, the SLSQP and EfficientSU2 with three and four reps would take between one and one and a half hours respectively to train on a local simulator, while the COBYLA and EfficientSU2 with three reps only took a minute and a half,

TABLE III: Test Case Distribution

Species	Label	Test Count
Setosa	0	7
Versicolor	1	9
Virginica	2	14

TABLE IV: Iris Dataset Summary

Characteristic	Detail
Number of Instances	150 (50 in each of three classes)
Number of Attributes	4 numeric, predictive attributes and the class
Attribute Information	Sepal length in cm Sepal width in cm Petal length in cm Petal width in cm
Classes	Iris-Setosa Iris-Versicolour Iris-Virginica
Missing Attribute Values	None
Class Distribution	33.3% for each of 3 classes
Creator	R.A. Fisher
Donor	Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
Date	July, 1988

with a bootstrapped accuracy of 87%. Due to the necessity of high numbers of repetitions in the experiments, this model was selected for the experiments.

2) QSVC Implementation: The Quantum Support Vector Classifier is the other quantum model implemented in this article, which is the quantum equivalent of the Support Vector Classifier. To build the QSVC, we need to specify a quantum kernel. The FidelityQuantumKernel class in Qiskit provides a quantum kernel implementation based on the principle of state fidelity. This takes in a feature map, for which we again use the ZZFeatureMap and a fidelity instance; ComputeUncompute which uses the sampler primitive to calculate the state fidelity of two quantum circuits following the compute-uncompute method. [24]

VI. Results

Table V provides the classification report generated from each of the models; the SVC showcases high precision across all classes, achieving perfect precision and recall for class 0 (setosa). Its performance remains strong for classes 1 (Versicolor) and 2 (virginica), with high f1-scores and overall accuracy of 93%.
The VQC had decent precision for classes 1 and 2, with scores of 90% and 92% respectively, with a high f1-score for class 1,

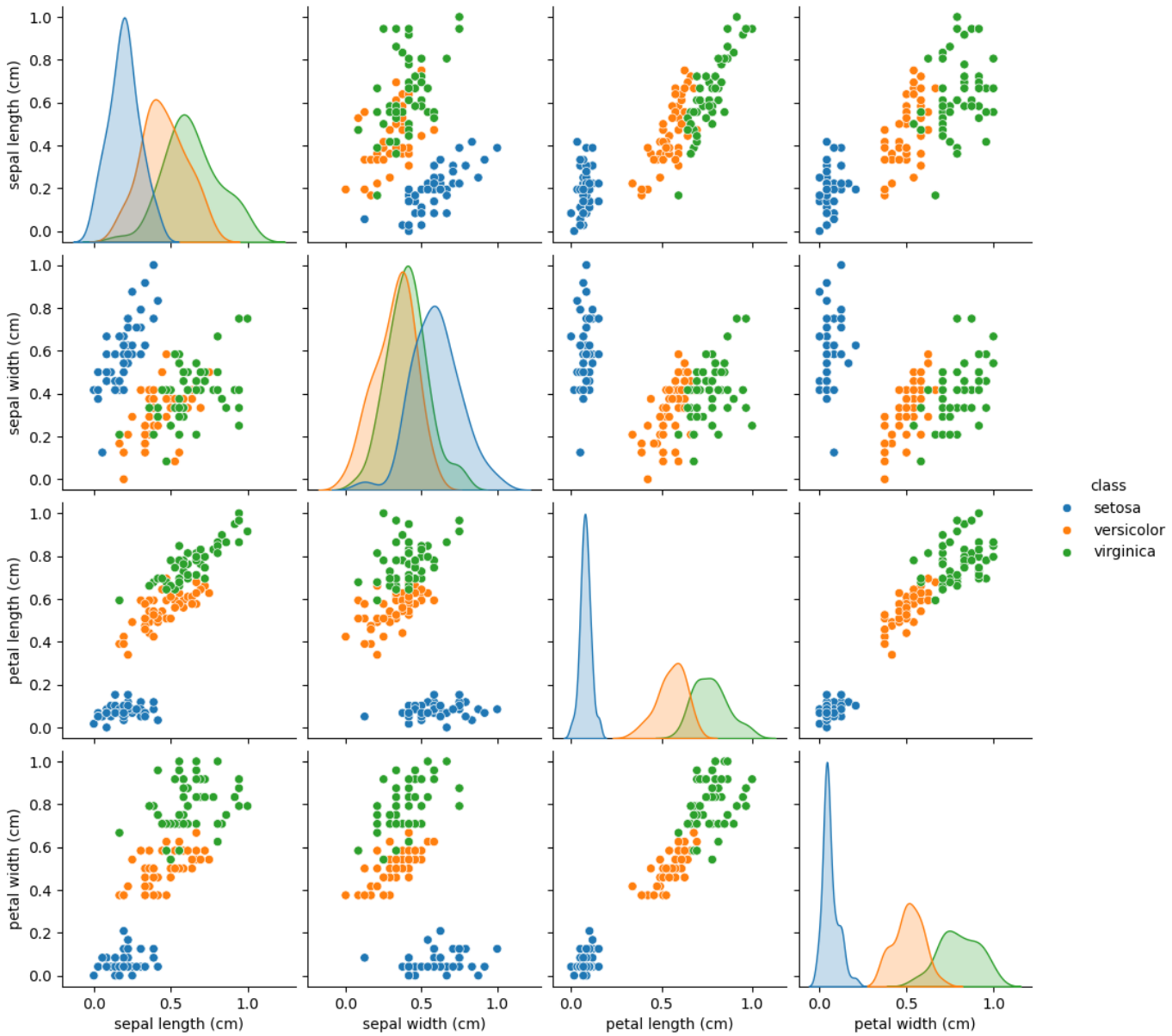


Fig. 9: Iris Pairs Plot.

but across almost all metrics, VQC appears to be the weakest classifier.

The QSVC presents excellent precision and perfect recall for classes 0 and 2, and a recall of 86% for class 1. The f1-scores are notably high, particularly for class 2, with an overall accuracy reaching 97%. This model demonstrates the best performance among the other classifiers, indicating its possibly superior capability to manage and balance dataset complexities.

The Random Forest classifier model exhibits 100% precision for classes 0 and 2 but slightly lower precision for class 1. While it achieves perfect recall for class 0 and class 1, its recall for class 2 drops, reflecting some difficulties in consistently identifying this class. The f1-scores are high for class 0 but moderate for classes 1 and 2, culminating in an overall accuracy of 90%. Although the model performs well, it shows specific weaknesses that could be mitigated through further optimization.

Overall, the QSVC stands out as the superior model, offering the highest accuracy and robust performance across all evaluated metrics, making it the preferred choice when quantum resources are available. The SVC follows closely behind, providing a highly effective and reliable alternative without the need for quantum enhancements. Both the SVC and QSVC exhibit strong capabilities in handling the classification challenges presented by the Iris dataset, whereas the RF, despite its strengths, shows areas for potential improvement.

To compare the models' overall accuracy against the test

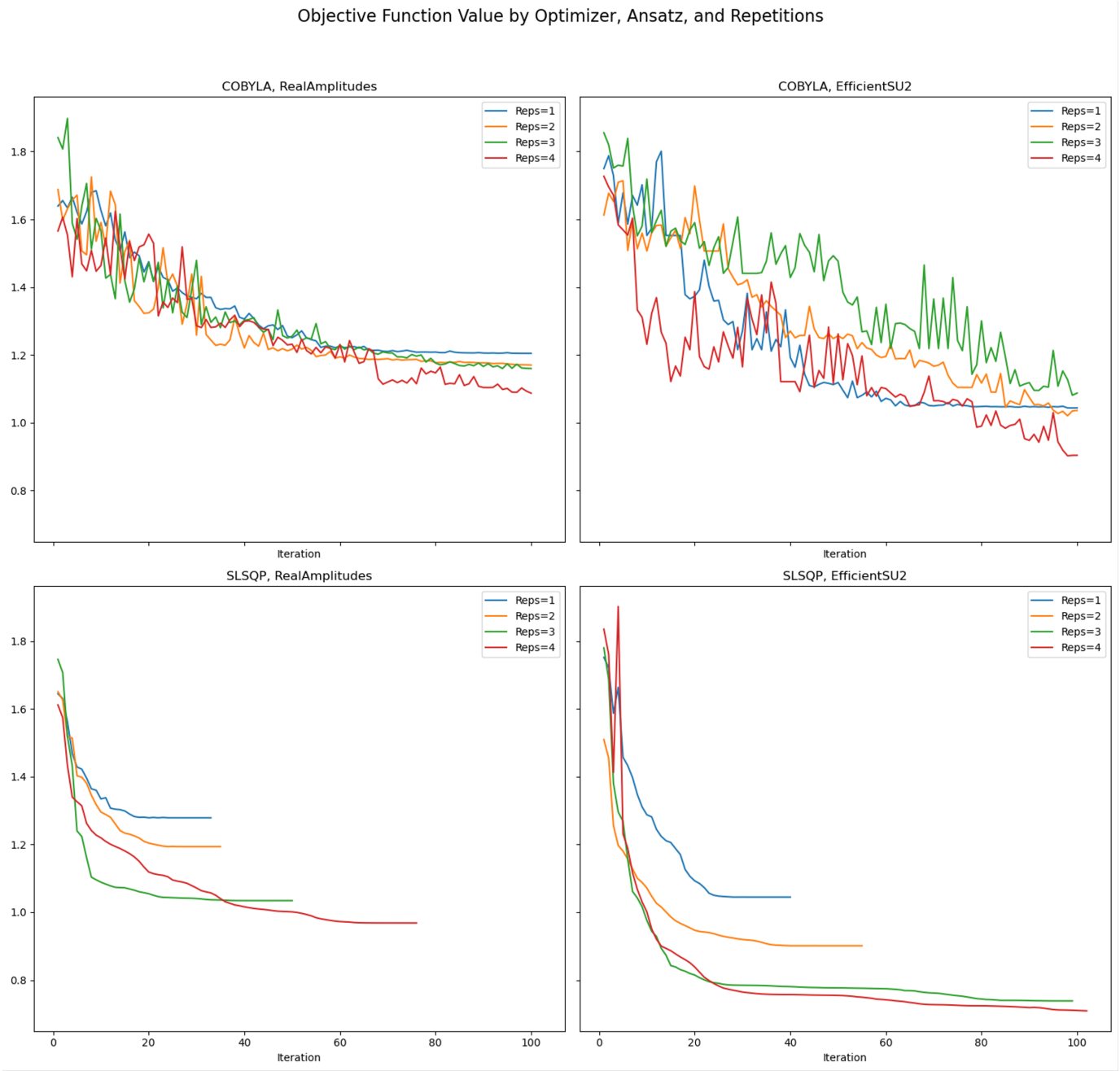


Fig. 10: Combinations of Optimiser and Ansatz for the VQC.

data, a bootstrap implementation was used using 1000 bootstrap resamples from the test data, as seen in table VI. The QSVC and SVC had average bootstrap results of 97% and 93% respectively, outperforming the random forest classifier and VQC. The box-plot 11 shows how the distributions of the results overlap, which can indicate that these models could be suitable for comparison.

For a clearer idea of how the models performed with respect to the individual test case classes, we can look at the confusion matrix in 13, which shows how the four models accurately classified all the test setosa (class 0) samples. This was expected and was also a handy sanity check as the setosa flower is clearly separable from the other two, which can be seen in the pairs plot 9. The most interesting observation is that the QSVC could accurately classify all 14 test instances of virginica (class 2). The QSVC also misclassified one of the versicolor (class 1) samples as virginica while both the SVC and Random Forest correctly classified all seven samples. The VQC had the poorest results by comparison, misclassified two of the versicolor samples, one as setosa and one as virginica. A breakdown of the misclassification can be found in table VII.

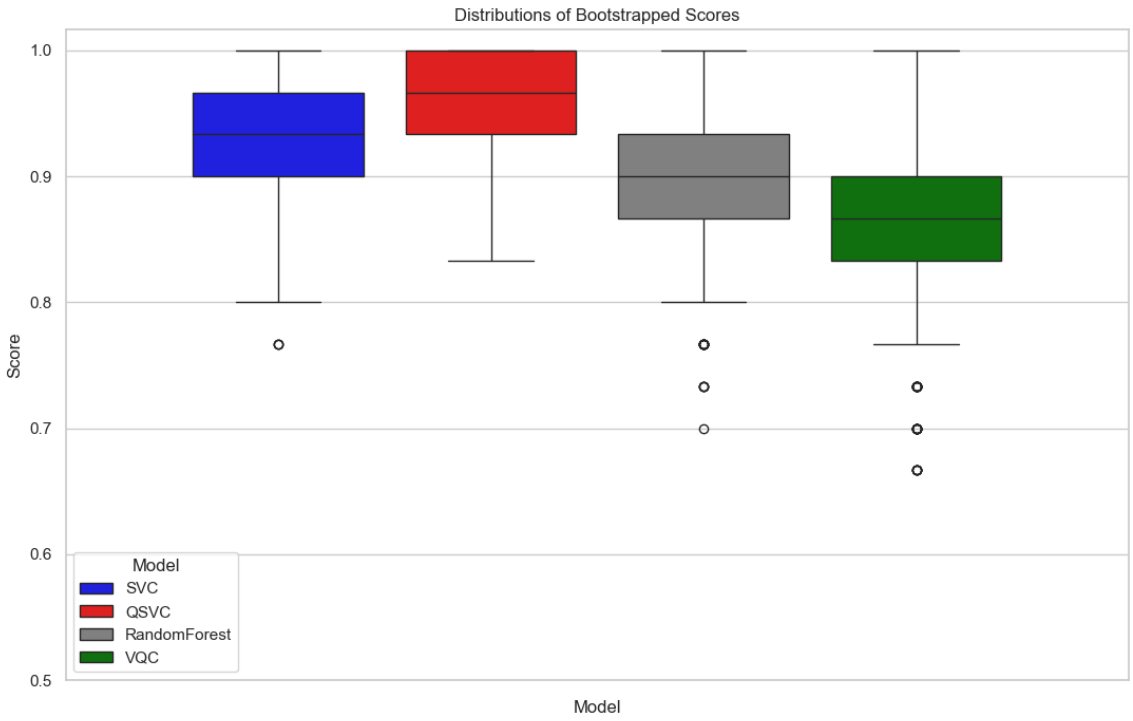


Fig. 11: Comparison of bootstrapped accuracy scores.

TABLE V: Summary of Classification Results by Model

Model	Metric	Class 0	Class 1	Class 2	Overall
VQC	Precision	0.90	0.71	0.92	
	Recall	1.00	0.71	0.86	
	F1-Score	0.95	0.71	0.89	
	Accuracy				0.87
SVC	Precision	1.00	0.78	1.00	
	Recall	1.00	1.00	0.86	
	F1-Score	1.00	0.88	0.92	
	Accuracy				0.93
QSVC	Precision	1.00	1.00	0.93	
	Recall	1.00	0.86	1.00	
	F1-Score	1.00	0.92	0.97	
	Accuracy				0.97
RF	Precision	1.00	0.70	1.00	
	Recall	1.00	1.00	0.79	
	F1-Score	1.00	0.82	0.88	
	Accuracy				0.90

TABLE VI: Bootstrapped Accuracy Results

Classifier	Mean Accuracy	25th Percentile	75th Percentile
SVC	0.932233	0.900000	0.966667
QSVC	0.965567	0.933333	1.000000
Random Forest	0.900333	0.866667	0.933333
VQC	0.864833	0.833333	0.900000

TABLE VII: Misclassification Details

Test Sample	Label	Misclassified As
4	2	SVC: 1, Random Forest: 1, VQC: 1
10	1	QSVC: 2, VQC: 2
15	1	VQC: 0
16	2	Random Forest: 1
17	2	SVC: 1, Random Forest: 1, VQC: 1

From table VII we can see that there are five of the thirty test cases were misclassified, with sample 4, which has a true label of 2 (virginica) misclassified as 1 (versicolor) by all models except the QSVC. For a view of what points were misclassified, refer to the plot 12, which highlights the five misclassified points. We can see that four of the points occur where there are some overlaps of the classes. We can see in this plot that point 15 (highlighted in green), which was misclassified by the VQC, seems to be clearly grouped with other versicolor samples. This suggests that there is still further optimisation that could be implemented in the VQC.

VII. Result Analysis and Discussion

A. Feature Importance

Three methods of feature importance were implemented; Leave One Out, Permutation Importance, and Accumulated

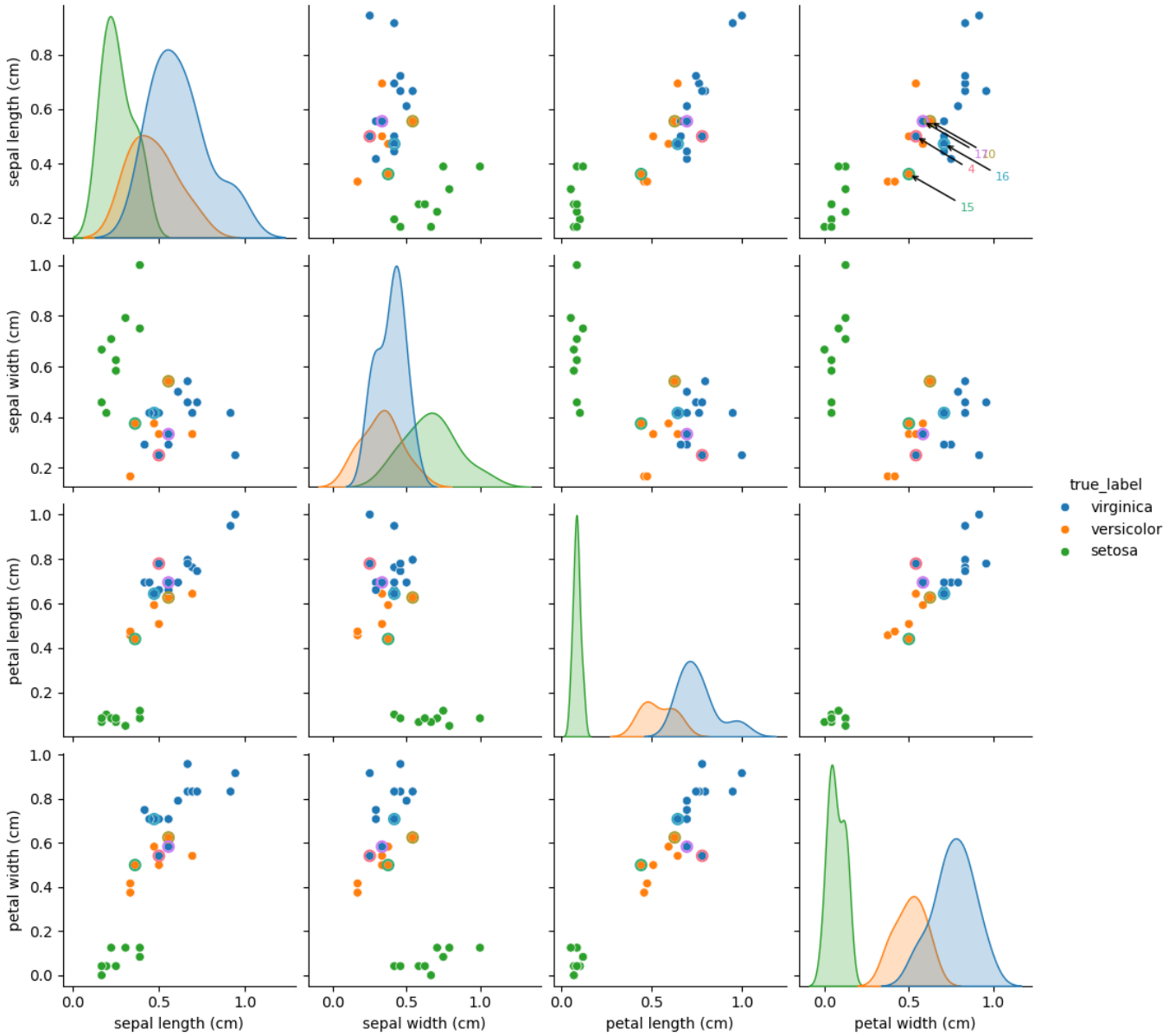


Fig. 12: Pairs plot of the test data points, highlighting the misclassified points. For misclassification breakdown refer to table VII

Local Effects. We will compare the results on a model by model basis, and an overall summary of each method results.

1) Leave One Out: To calculate LOO feature importance, we rerun the model omitting the feature of interest, and check how the performance of the model is affected.

SVC Leave-One-Out: The effects of the feature omissions in the SVC can be seen in the plot 14, and in the table VIII. We can see from the results that when either petal length or petal width is missing, the accuracy of the model increases by 3.3%. This may indicate that petal length or petal length could be poor predictors for the SVC. This counterintuitive increase in test scores upon removing 'Petal Width' and 'Petal Length' could indicate an over-reliance on these features or it could be the removal of an interference effect due to the high correlation between these two feature, suggesting that only one of them are really needed in the model. Accuracy remains unchanged when the sepal length or width is omitted. This suggests that the sepal length or width may not be a critical feature for the model, possibly because other features provide similar or sufficient information for making accurate classifications.
Due to the minimal differences, this analysis suggests that SVC regards all of the features to a similar importance.

Random Forest Leave-One-Out: The effects of the feature omissions in the Random Forest can be seen in the plot 16, and in the table IX. We can see from the results that when either

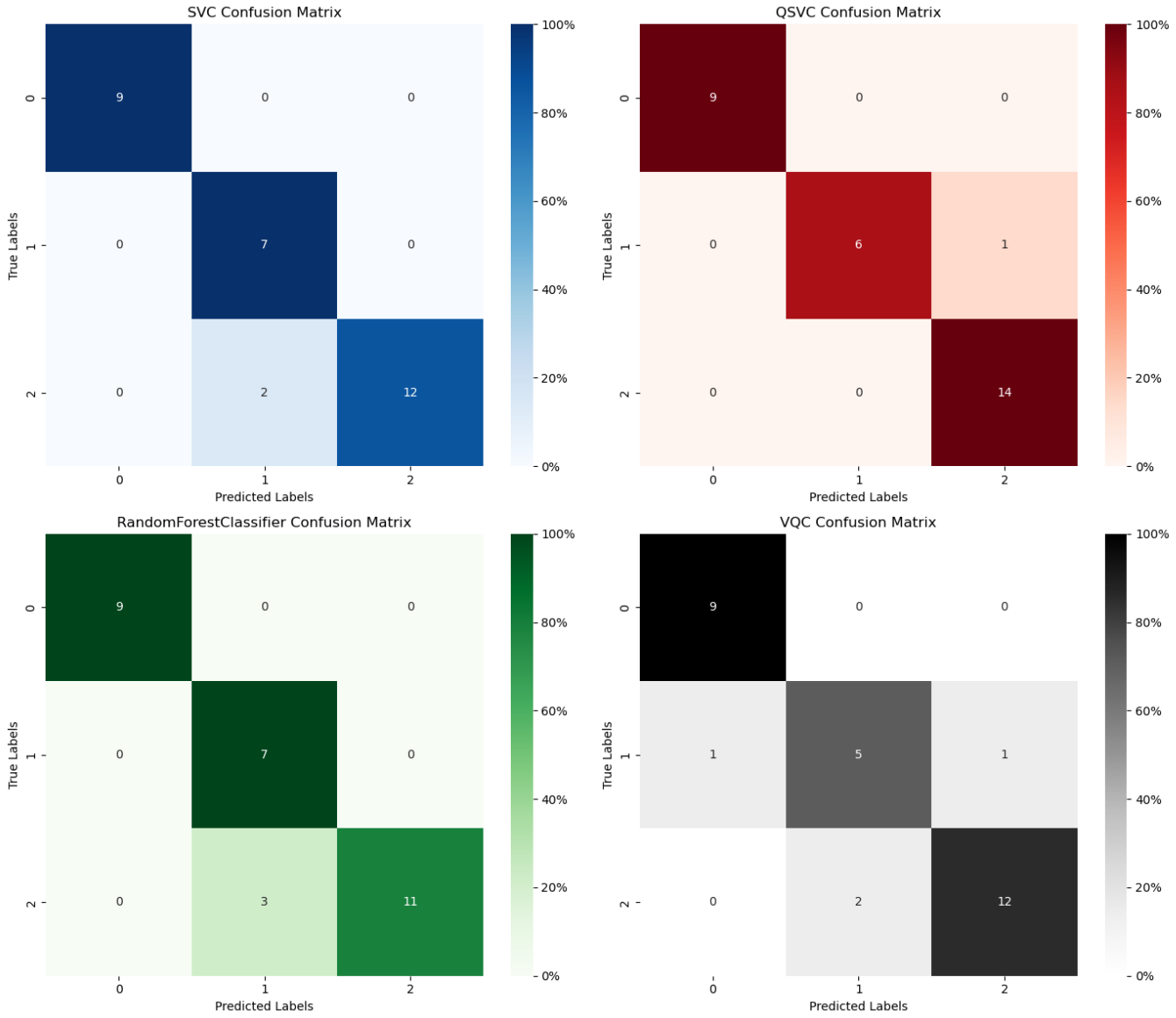


Fig. 13: Confusion matrices of models.

TABLE VIII: SVC LOO Results

Scenario	Accuracy Score
Full Feature	0.90
Missing Sepal Length	0.90
Missing Sepal Width	0.90
Missing Petal Length	0.93
Missing Petal Width	0.93

petal length is missing, the accuracy of the model increases by 3.3%. This may indicate that petal length could be a poor predictor compared to the others. This is interesting due to the high correlation between petal length and petal width, which would suggest that if petal length has an effect, petal width would cause a similar effect. Accuracy remains unchanged when the sepal length or width is omitted. This suggests that the sepal length or width may not be a critical feature for the model, possibly because other features provide similar or sufficient information for making accurate classifications.

Due to the minimal differences, this analysis suggests that the Random Forest regards all of the features to similar importance like the SVC possibly assigning petal length with the lowest importance.

TABLE IX: Random Forest LOO Results

Scenario	Accuracy Score
Full Feature	0.90
Missing Sepal Length	0.90
Missing Sepal Width	0.90
Missing Petal Length	0.93
Missing Petal Width	0.90

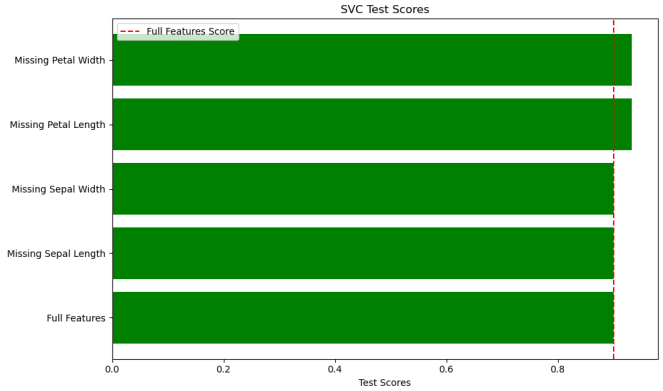


Fig. 14: SVC Leave One Out Feature Importance.

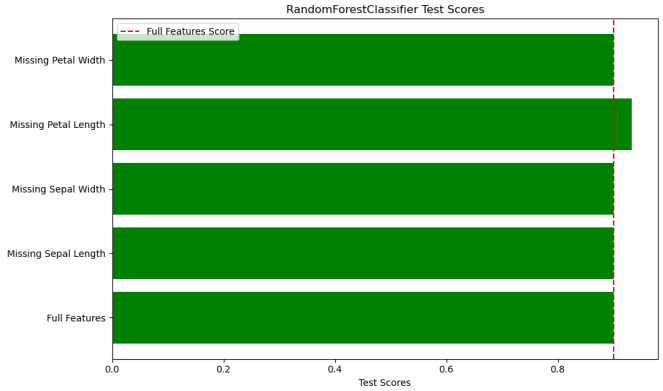


Fig. 15: Random Forest Leave One Out Feature Importance.

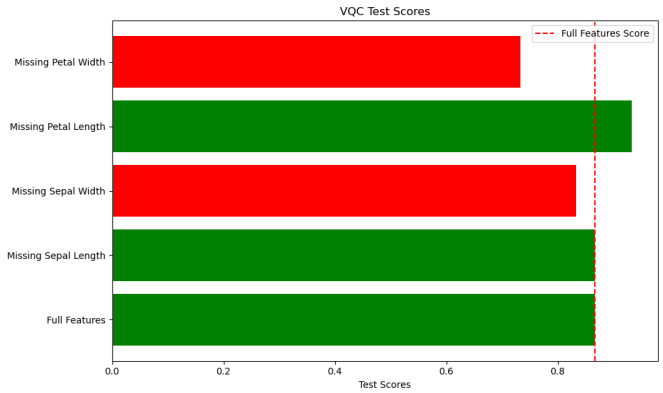


Fig. 16: VQC Leave One Out Feature Importance.

VQC Leave-One-Out: The effects of the feature omissions in the VQC can be seen in the plot 16, and in the table X. The LOO results are much different to the SVC and Random Forest. The full-featured model has an accuracy of 87%, but this time the removal of petal width caused a drop to 73%, a 16% decrease from our full model, a much bigger difference to the 3.3% difference we found with the SVC and no change in the Random Forest classifier. This causes the effect of removing petal length to be somewhat paradoxical, increasing the model's performance by 6.9%. Due to the highly correlated nature of petal length and width, they would be expected to have similar predictive effects in a model, however, they have opposite effects on the model's performance.

We can also see that the effect of removing sepal width, which had no effect in either the SVC or Random Forest, caused a 4.6% decrease in the VQC's accuracy. These results demonstrate that the VQC has highly different levels of feature importance, with petal width being considered the most important, as its absence causes the biggest decrease in accuracy, followed by sepal width for the same reason, then sepal length and petal length being the least important as its absence improves the models' accuracy.

TABLE X: VQC LOO Results

Scenario	Accuracy Score
Full Feature	0.87
Missing Sepal Length	0.87
Missing Sepal Width	0.83
Missing Petal Length	0.93
Missing Petal Width	0.73

QSVC Leave-One-Out: The effects of the feature omissions in the VQC can be seen in the plot 17, and in the table XI. We can see that the QSVC had similar accuracy to the VQC, but ended up with highly different results, more in line with what we observed in the SVC and Random Forest, the removal of individual features had very little overall effect on the model's accuracy. What has remained consistent across all the results was the removal of petal length slightly increased the model's accuracy, in this case by 3.5%. There was also an identical effect with the removal of sepal length, indicating that sepal length introduces some noise into the model.

According to this LOO analysis, we can see that QSVC acts somewhat similarly to the classical models, regarding all the models with similar importance.

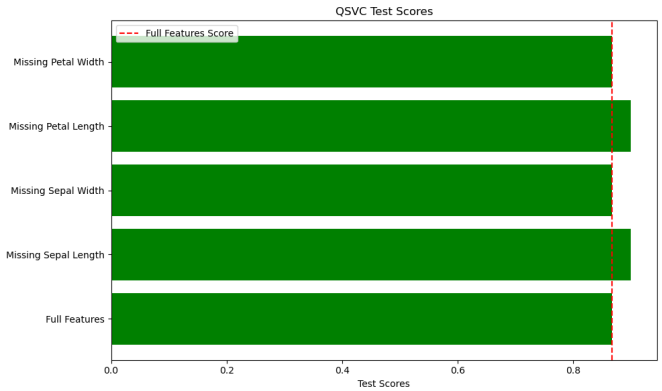


Fig. 17: QSVC Leave One Out Feature Importance.

LOO Summary: Figure 18, shows a full comparison of LOO feature importance across all the models. The VQC model exhibits a highly different approach to assessing feature importance, particularly diverging from the other models in its

TABLE XI: QSVC LOO Results

Scenario	Accuracy Score
Full Feature	0.87
Missing Sepal Length	0.90
Missing Sepal Width	0.87
Missing Petal Length	0.90
Missing Petal Width	0.87

valuation of Petal Width. The VQC was also the only model to demonstrate a decrease in performance at the removal of any features. The removal of a feature in the other three models either did not affect accuracy or increased it. Interestingly, the QSVC, while also a quantum model, seems more aligned with the classical SVC and RF models in terms of feature impact, except with Sepal width, whose removal increases accuracy. Both quantum models demonstrate a sensitivity to Sepal features, yet they differ in which specific Sepal features they find more important. The QSVC shows a marked increase in importance for Sepal Length, a trend not observed in the VQC model. Meanwhile, the VQC assigns considerable importance to Sepal Width, unlike the QSVC. This difference demonstrates the unintuitive ways in which quantum models process feature importance compared to the classical models.

2) Permutation Importance: Permutation importance differs from LOO by instead of removing the feature entirely, the feature values will be repeatedly shuffled to break the relationship between the feature and the class. This method will allow us to get a gauge of the variation of the importance.

SVC Permutation Importance: We can see in the plot 19 that petal width and petal length have the biggest impact when their values are permuted, and sepal width seems to have little to no effect on the model accuracy. The permutation of sepal length has no effect whatsoever. The results of petal length and petal width being permutated are extremely interesting as they appear to demonstrate the opposite effect in the LOO analysis, seen in the plot 14. This result makes more intuitive sense as petal length and width are considered to have strong predictive power for the Iris dataset. However, this may indicate that the model may be better off with just petal width rather than both petal width and petal length. Another possible cause is the Permutation Importance artificially introduces noise in the data so that it would be highly unlikely that a model's performance would increase when a feature is permuted.

Random Forest Permutation Importance: We can see in the plot 20 that petal width and petal length have the biggest impact when their values are permuted, while sepal length and sepal width seem to marginally increase the model's accuracy. These results are also quite different compared to the LOO analysis in plot 15, where the removal of petal length causes a slight increase in the model's accuracy. This result closely aligns with the SVC permutation importance results, which also follow the intuition of petal length and width being strong predictors. We could assume the same reasons mentioned in the SVC Permutation analysis VII-A2 for the difference seen in this result and the LOO.

VQC Permutation Importance: The results of Permutation Importance, in plot 21 shows that the VQC has a much more balanced distribution of the features' importance, with sepal length having the second highest importance after petal width, which differs from the two classical models who both had a very similar ranking of petal width and length being the most importance with sepal length and width having little to none. The size of the whiskers on the plots and the high interquartile ranges further show how similarly the VQC considers the features. The permutation results still differ from the LOO analysis. Where the removal of petal length increased model performance, and removing sepal length did not affect performance.

QSVC Permutation Importance: We can see that again in plot 22 that the QSVC model follows a similar distribution of feature importance to the classical models, with petal length and petal width causing the biggest decrease in model accuracy when their values are permuted, and sepal width and sepal length causing slightly bigger effects but still much lower than the petal features. We can see that the LOO results demonstrated no effects with the removal of petal width and sepal width, and removing petal length and sepal length causes a slight increase in model accuracy.

Permutation Importance Summary: The plot 23 we can see that the VQC has a fundamentally different approach to feature importance compared to all the other models, including the QSVC, with almost equal weighting across all the features. For the other three models, we can a clear preference for petal length and petal width over sepal length and sepal width, with the lower wicks of the petal features being above the top wicks of the sepal features. The QSVC does seem to place slightly more importance on the sepal features than the classical models, but one should note that depending on the permutation of sepal width, the median effect was zero, with some permutations resulting in a better performance, but tending to worse, indicated by the high whisker.
Overall we can see that according to permutation importance analysis, petal features have a higher ranking importance than sepal. This seems to follow LOO results, which shows that petal features have a larger effect than the sepal counterparts, but seemingly in an opposite direction (18).

3) Accumulated Local Effects Feature Importance: ALE computes feature importance differently to the LOO and Permutation Importance methods, rather than using model accuracy, ALE uses probabilities to measure how much the probability of making a classification changes concerning how the value of the feature changes. This can be aggregated and used to get an overall score of a model's feature importance, which will be analysed briefly. Due to the probabilistic nature of ALE, the VQC model is unable to produce a probability vector, as this functionality was deprecated with Qiskit 1.0 [41]. Thus the QSVC model was used as the quantum comparison. For a clearer comparison, we will compare the SVC against its quantum counterpart.

Plot 24 shows that both models follow a similar distribution of importance, however the QSVC results with higher ALE

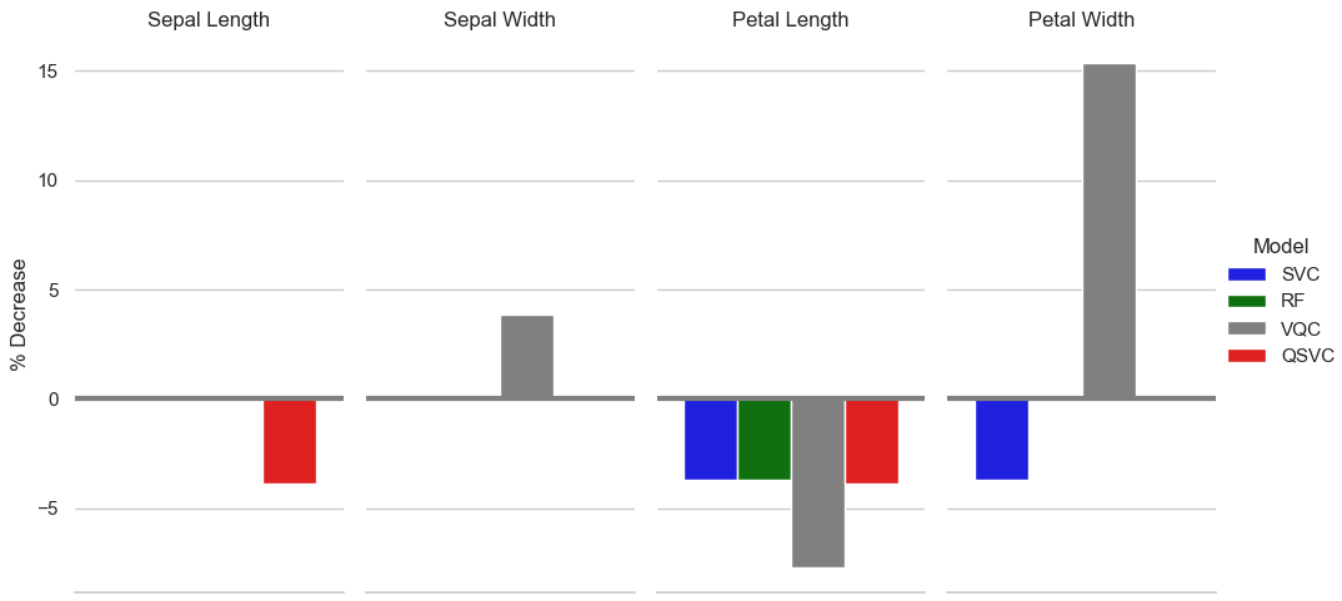


Fig. 18: Overall LOO results.

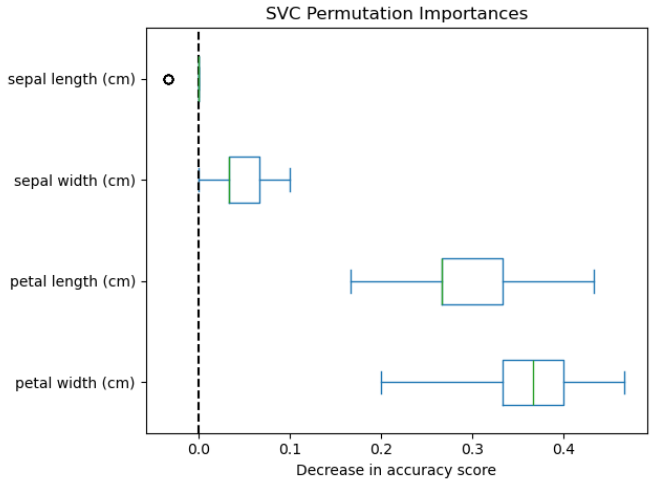


Fig. 19: SVC Permutation Importance.

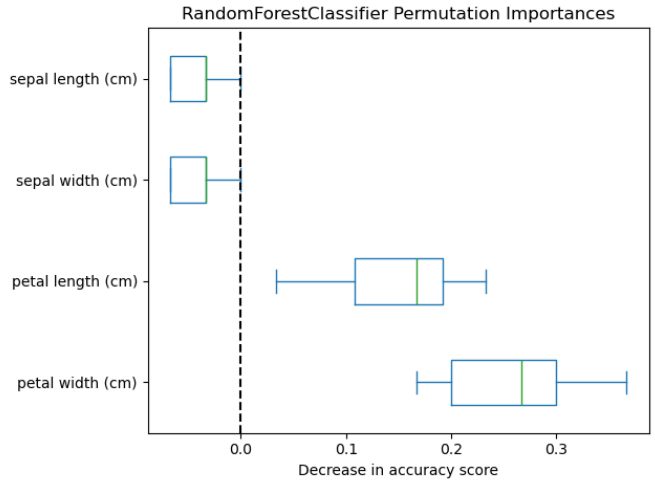


Fig. 20: Random Forest Permutation Importance.

influence in three out of the four features, with the largest increase in the importance of petal length. But overall the ALE values of both the SVC and QSVC are highly similar.

A better way to utilise and understand ALE values is in plots 25 and 26. These plots show the ALE values for SVC and QSVC respectively. We can interpret the values as the increase or decrease in the probability of predicting the class of interest. We can see that petal length and width are important features due to how much the ALE values change for each of the values. For example, we can see that the probability of predicting Virginica in both SVC and QSVC increases as petal length increases. We can also see in the QSVC that sepal width has more of an impact in the model prediction as values increase, while the same has very almost no effect in the SVC model.

VIII. Explainability

While feature importance can be useful for a general overview of what are the most important features in a model, more often than not we would like to see more specific explanations for a particular classification or individual result. The two methods used for this are Accumulated Local Effects and SHAP.

1) ALE For Class Explanation: We have shown in section VII-A3 how ALE can be used to show how features affect a model's prediction, this can be inverted so that we can see how the different models differ in predicting a class. Given how we

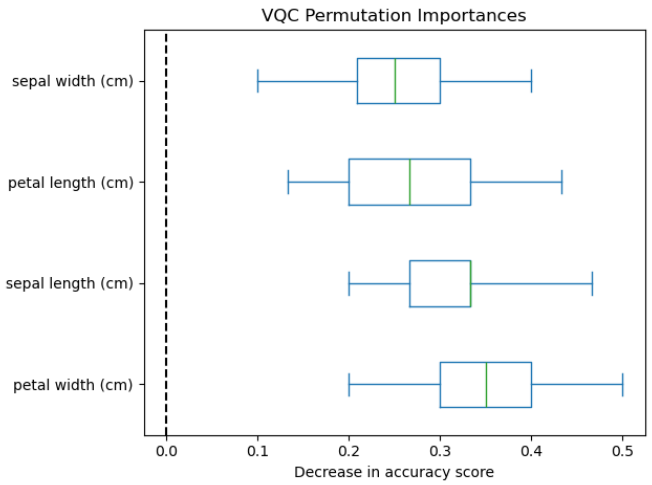


Fig. 21: VQC Permutation Importance.

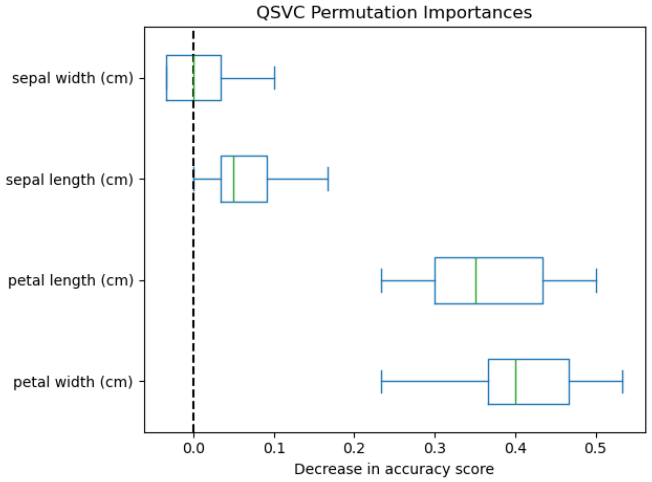


Fig. 22: QSVC Permutation Importance.

can see that the Versicolor and Virginica classes overlap in the pairs plot 9, we will investigate how ALE values can explain how the different models differ in predicting these classes.

Versicolor Prediction Model Comparison: In plot 27 we can see how the ALE values for each feature differ by model when the class of interest is Versicolor. We can see that all three models follow similar trends for each of the features, however, we can see that a higher value of sepal width resulted in the QSVC increasing its probability of predicting the sample as Versicolor, while the same value had zero impact in the SVC and Random Forest models. A very interesting observation is also made concerning sepal length. While a high value of sepal length indicated a higher probability of being Versicolor, a higher value of sepal length decreased the probability of making that same classification. However, we can see the magnitude of these changes is only a fraction of the decision when compared to the ALE values for petal length and petal width, with the sepal features giving ALE values in the region of 0 to 0.05 compared to the petal features who's effects are in the range of 0 to 0.4, which strongly decrease the probability of predicting a Versicolor class.
Plot 27 show that the three models are most likely to predict that a case is Versicolor when the normalised values of petal length and width are approximately between 0.25cm and 0.65cm, with ALE scores decreasing at either side of this interval.

Virginica Prediction Model Comparison: How the ALE values fluctuate across the features and models is shown if plot 28. Similarly to the Versicolor analysis, we can see that the QSVC's ALE values increase with sepal length and width, but this time the effect is greatly magnified with the same pattern in petal length and width. What we could infer from this visualisation is that for high values of all four features, the QSVC model has a higher probability of predicting Virginica.

When the two plots 27 and 28 are viewed together we really see how petal width and length differentiate the model's predictions between the two classifications. With all three models significantly decreasing as petal measurements increase in predicting Versicolor and at the same time, we can see the ALE values for Virginica increasing. This could also help us further understand where the model may make mistakes, particularly when an instance of petal length and width is in approximately the 0.6 to 0.7 regions.

We can see in this analysis how much more information can be inferred from the data by ALE by both classical and quantum classification models when analysis goes beyond the generality of feature importance analysis. Such analysis could also be made using SHAP values, which expands again on the ALE values by proving clear breakdowns of individual results.

2) SHAP For Case-Specific Predictions: SHAP uses game theory coalition values to estimate the marginal contribution from a feature to a model's prediction. This is a method that can be used to quantify how features in a model contribute to individual prediction, rather than focusing on a feature importance method like Permutation Importance, which has its merits when we want to get an idea of how a model would typically work, however, these generalisations cannot quantify the effects in individual cases.

The table VII, and the plot 12 show us that five out of the thirty test points are misclassified. We can see that point 4, which has a true label of Virginica was misclassified as Versicolor (label 1) by all models except the QSVC, this makes this an ideal candidate to use for further investigation. The feature values for point 4 can be found in table XII.

SVC SHAP Values: Figure 29 is a waterfall plot of the SHAP values for sample 4. We can see that the expected output, or base value is 1.1, which would fall into the Versicolor class, we can see that Feature 3 (petal width) causes the largest contribution to this prediction, however the push is almost completely offset by feature 2 (petal width),

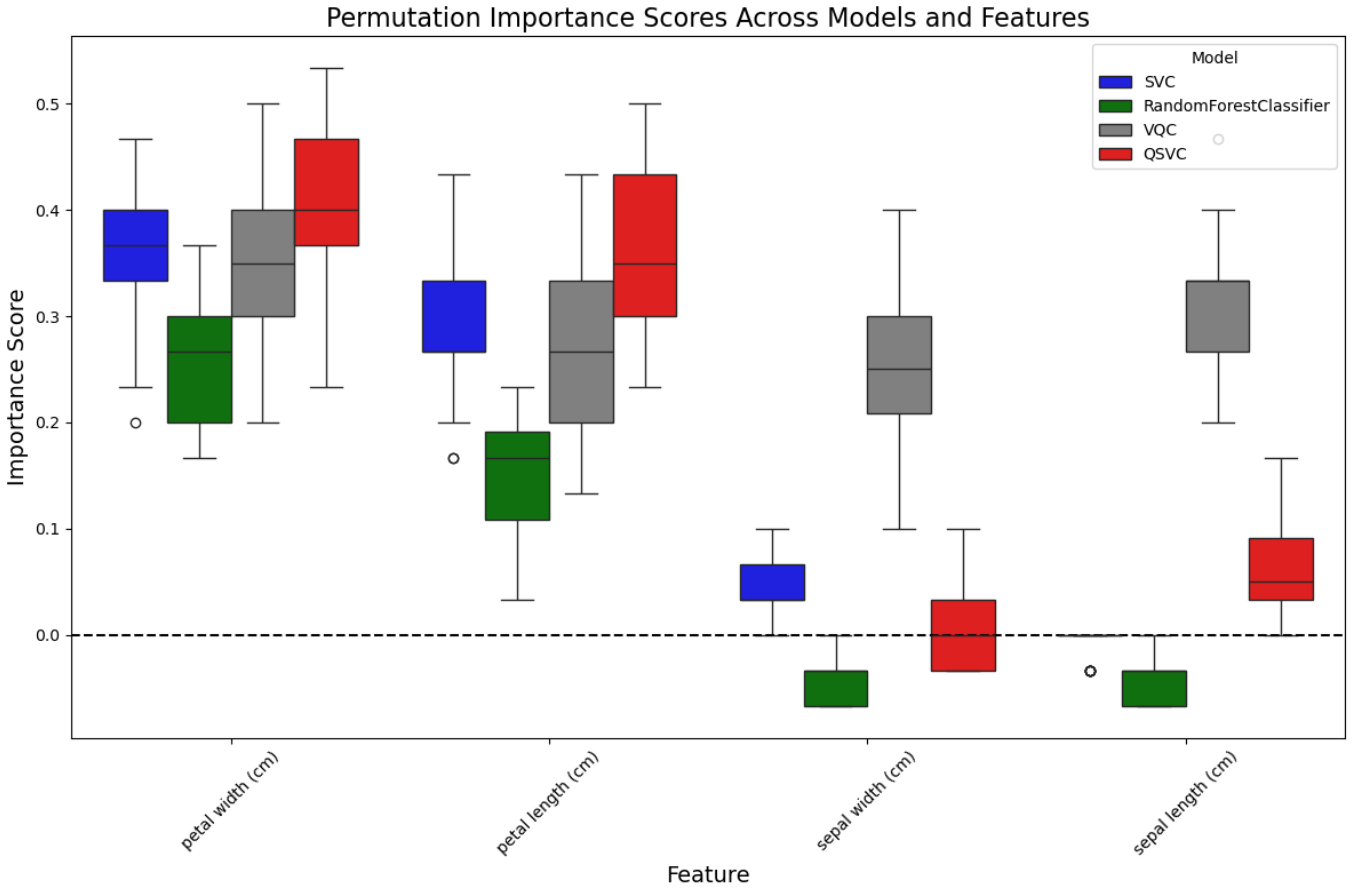


Fig. 23: Overall Permutation Importance.

Feature	Value
0 (Sepal Length)	0.5
1 (Sepal Width)	0.25
2 (Petal Length)	0.78
3 (Petal Width)	0.54

TABLE XII: Features of the test point 4

this confirms what has been discussed previously, being petal width and length being the most important features in the SVC. The SHAP values quantify their importance in making this particular prediction. We can see the effects of sepal length and width are minimal in comparison. The contrasting effects by features two and three that the model uses could be explained that the model could be using the effect of petal width and length to cancel each other out in order to stay, on the base classification and align with the classification of 1, Versicolor.

Random Forest SHAP Values: ForWith respect to the Random Forest classifier, we can see that it tended to go towards a setosa classification, again we can see that petal length strongly pushes to the right, however in this model we can see that it is sepal width is a highly important feature for this case, which differs if we look at an aggregation of all the SHAP values in plot 31, where petal sepal width almost has no importance overall.

Incorrect classification aside, this shows how much SHAP values can demonstrate how important it is to be able to understand a models functionality on a case-by-case basis rather than using a general overview of a features importance.

QSVC SHAP Values: In our QSVC we can see that the model decided that all the feature values indicated a Virginica classification, with petal length and width being the two most important features by far, with sepal length only being assigned a SHAP value of 0.02 and sepal width not contributing at all in this classification. It is also an interesting thing to note that none of the feature values in the QSVC suggested that the prediction should change in the other direction.

3) Summary of Analysis:

Feature Importance: We have explored methods of feature importance across different models using three distinct methodologies: Leave One Out, Permutation Importance, and Accumulated Local Effects. Each method offered an insight into how features affect model performance and accuracy. We have then applied these methods to QML models.

LOO analysis highlighted the differential impact of features across models. For instance, SVC and Random Forest models

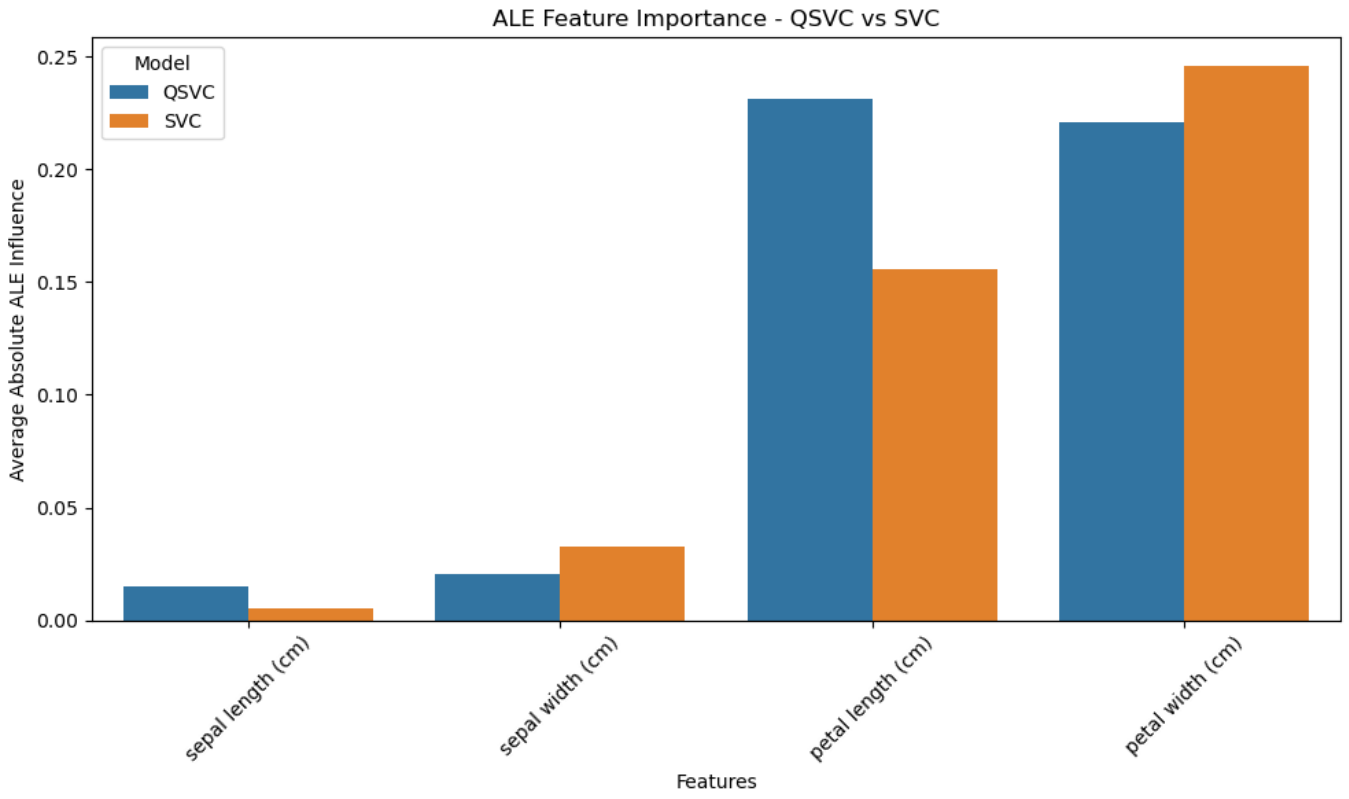


Fig. 24: ALE Importance.

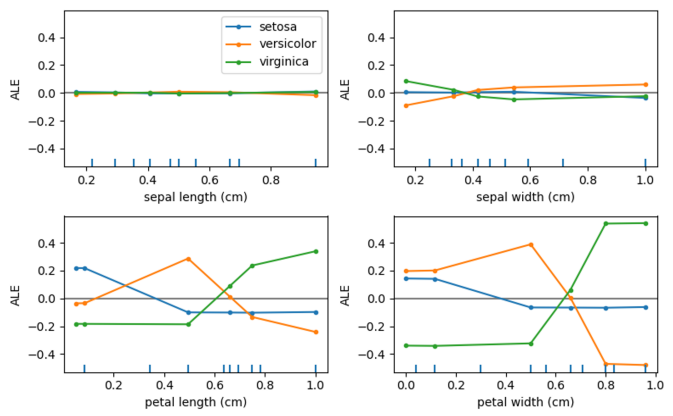


Fig. 25: SVC ALE.

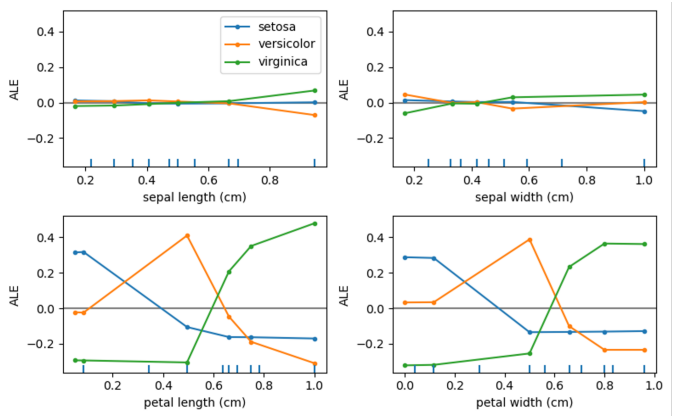


Fig. 26: QSVC ALE.

showed an increase in accuracy upon the removal of petal features, suggesting a possible over-reliance or redundancy. In contrast, the VQC model exhibited significant performance drops, especially with the removal of petal width, indicating a high dependency on this feature. This disparity underscores the different ways models prioritize or handle feature dependencies.

Permutation importance further emphasized the significance of petal features in classical models, consistent with LOO results but provided additional details on feature interactions and their impact on model accuracy. The QSVC model's feature importance distribution appeared more aligned with classical models, however, we can observe some variations in sepal features' impact.

ALE uses probabilities to gauge a feature's impact on the model, however, due to constraints in the implementation of the VQC, this model had to be omitted from this method. The ALE plots demonstrated how the QSVC follows the trends seen in the classical models, but displays a higher sensitivity to the changes across all the features with almost no flat lines

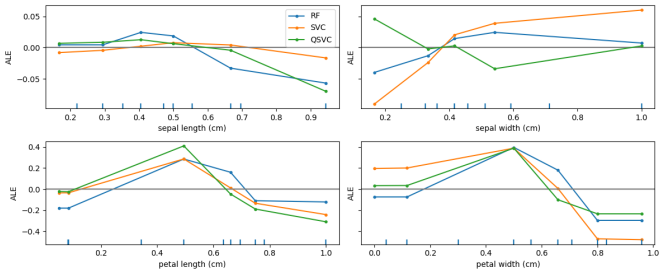


Fig. 27: Versicolor ALE Values.

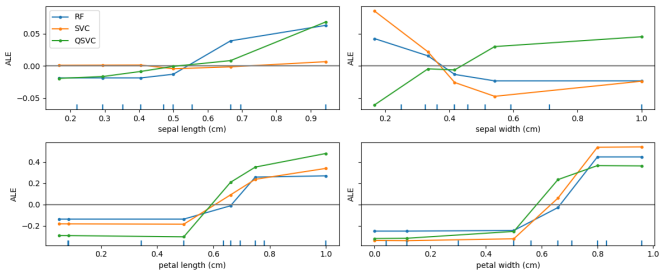


Fig. 28: Virginica ALE Values.

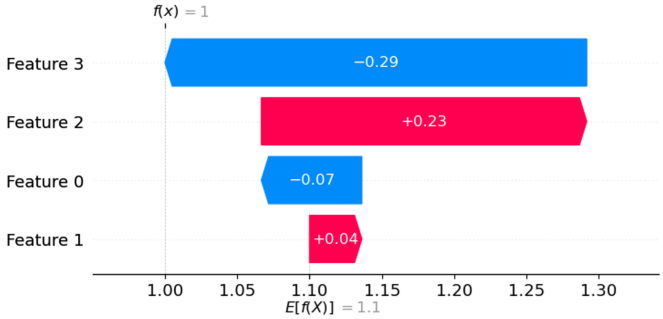


Fig. 29: SVC SHAP

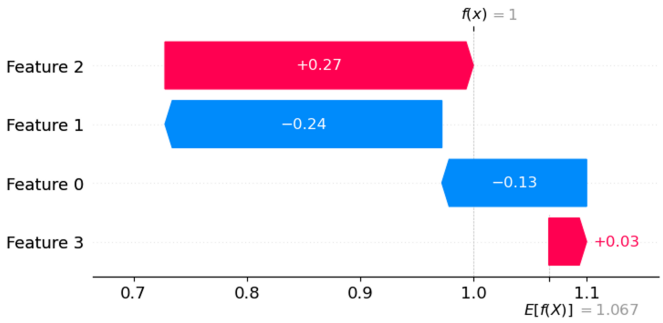


Fig. 30: Random Forest SHAP

in any of the ALE plots 27 28.

Model Explainability

Model explainability was addressed through ALE and SHAP analyses, focusing on how specific features influence individual predictions. ALE analysis demonstrated the

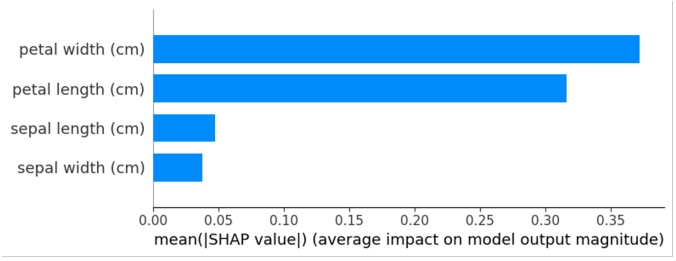


Fig. 31: Random Forest SHAP Feature Importance

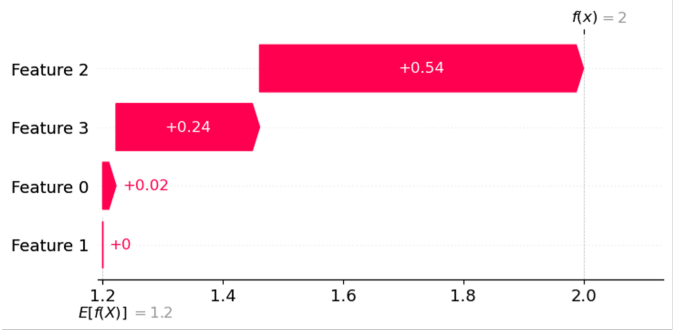


Fig. 32: QSVC SHAP

variable influence of features across different classes, providing insights into how models differentiate between the classes Versicolor and Virginica based on feature values. This analysis allows us to uncover a better understanding of how the decision-making processes in both classical and quantum models work.

SHAP values offered deeper insights into the contribution of individual features to specific predictions, highlighting the interplay of features in models like QSVC, which successfully identified the Virginica class in test point 4, which was misclassified by the other three models.

IX. Conclusion

In this article we investigated different methods of feature importance (LOO, Permutation Importance, ALE) and explainability (ALE, SHAP) and applied these methods to quantum machine learning models, comparing the results to their classical counterpart. In this comparison, we have set a small milestone in this field of explainable quantum machine learning. This article displays that these methods can indeed be applied to QML models, and so are potential candidates as tools in the unexplored field of explainable quantum machine learning, or XQAI. It was also a highly interesting takeaway in that we haven't necessarily seen a reliable example of the quantum advantage in our QML models, probably due to the small scale of the data, but there are still applications to be explored through QML, for example, researching gradients and making quantum software ready for machine learning applications [42].

It is also important to remember that these are methods that were designed, tested, implemented and are used with almost

exclusively classical machine learning tools in mind, meaning that there is likely a need for not only amending these tools but perhaps a need to come up with completely new quantum-specific methods of explainability. [35]

The plans for this article are to take the QML models and train them with more complicated and larger, real-world data. This would further enhance the necessity of applying explainability techniques. To do this we would need to further optimise the quantum models, as well as design them optimally for running on a real quantum computer. The lack of models trained on an actual quantum computer was one of the greatest limitations of this article, as a simulator was used to avoid the extra time and complexity required, which was able to work due to the small scale of the data used. Due to the necessary time and resources this would take, attempting this on a larger dataset was beyond what was believed to be a realistic task for this article's time frame. It would also be required to explore more methods of quantum machine learning and algorithms such as Quantum Boltzmann Machines, and perhaps explore the possibility of quantum-specific explainability techniques.

The notebooks containing the code, and the resources for this article can be found on the repository accessible at: https://github.com/LukePower01/ml-to-qml

REFERENCES

[1] Andertoons, "Shakespeare's cat: "to be or not to be" - schrödinger's cat," 2024. [Online]. Available: https://andertoons.com/cat/cartoon/8046/shakespeares-cat-to-be-or-not-to-be-schrodingers-cat-both

[2] A. Burkov, *The Hundred-Page Machine Learning Book*, 2019.

[3] M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," *Contemporary Physics*, vol. 56, no. 2, Oct 2014.

[4] D. J. Wineland, "Nobel lecture: Superposition, entanglement, and raising schrödinger's cat," *Reviews of Modern Physics*, vol. 85, no. 3, p. 1103–1114, Jul 2013.

[5] V. Belle and I. Papantonis, "Principles and practice of explainable machine learning," *Frontiers in Big Data*, vol. 4, Jul 2021.

[6] E. Horel, "Towards explainable ai: feature significance and importance for machine learning models," Ph.D. dissertation, Stanford University, 2020.

[7] [Online]. Available: https://scikit-learn.org/stable/modules/permutation_importance.html

[8] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," 2017.

[9] [Online]. Available: https://shap-lrjball.readthedocs.io/en/latest/

[10] C. Molnar, "Interpretable machine learning," Aug 2023. [Online]. Available: https://christophm.github.io/interpretable-ml-book/ale.html

[11] Mazen Ahmed, "Non-linear support vector machines explained," 2021. [Online]. Available: https://linguisticmaz.medium.com/support-vector-machines-explained-ii-f2688fbf02ae

[12] E. Wolsztynski, "Support vector machines."

[13] ——, "Random forest."

[14] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, 2017.

[15] H.-Y. Huang, M. Broughton, J. C. S. Chen, J. Li, M. Mohseni, H. Neven, R. Babbush, R. Kueng, J. Preskill, and J. R. McClean, "Quantum advantage in learning from experiments," *Science*, vol. 376, no. 6598, 2022.

[16] IBM, "Quantum advantage." [Online]. Available: https://www.ibm.com/thought-leadership/institute-business-value/public/static/images/Quantum-Report/Figure3.svg

[17] D. Diego, M. P. da Silva, C. A. Ryan, A. W. Cross, A. D. Córcoles, J. A. Smolin, J. M. Gambetta, M. J. Chow, and B. R. Johnson, "Demonstration of quantum advantage in machine learning," *npj Quantum Information*, vol. 3, no. 1, 2017.

[18] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, 2019.

[19] Qiskit, "Platypus/notebooks/summer-school/2021 at main · qiskit/platypus." [Online]. Available: https://github.com/Qiskit/platypus/tree/main/notebooks/summer-school/2021#

[20] S. Pattanayak, *Quantum Machine Learning with python: Using CIRQ from Google Research and IBM qiskit*. Apress, 2021.

[21] I. Glendinning, "The bloch sphere," in *QIA meeting. Vienna*, 2005.

[22] Qiskit, "Building a quantum variational classifier using real-world data," 2021. [Online]. Available: https://medium.com/qiskit/building-a-quantum-variational-classifier-using-real-world-data-809c59eb17c2

[23] A. Elbeltagi, "Vqc schematic." [Online]. Available: https://www.researchgate.net/figure/Schematic-representation-of-a-variational-quantum-circuit-VQC-VQC-is-created-by_fig1_358006357

[24] "Quantum kernel machine learning - qiskit machine learning 0.7.2." [Online]. Available: https://qiskit-community.github.io/qiskit-machine-learning/tutorials/03_quantum_kernel.html

[25] M. Rath and H. Date, "Quantum data encoding: A comparative analysis of classical-to-quantum mapping techniques and their impact on machine learning accuracy," 2023.

[26] [Online]. Available: http://ndl.ethernet.edu.et/bitstream/123456789/73371/1/320.pdf

[27] Qiskit, "Ibm quantum documentation," 2021. [Online]. Available: https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.ZZFeatureMap

[28] A. Das and P. Rad, "Opportunities and challenges in explainable artificial intelligence (xai): A survey," *arXiv preprint arXiv:2006.11371*, 2020.

[29] "Permutation importance." [Online]. Available: https://www.google.com/url?sa=i&url=https%3A%2F%2Ftowardsdatascience.com%2Ffeature-importance-with-neural-network-346eb6205743&psig=AOvVaw2sVSobbrj_S1EHpceIQimr&ust=1712314565451000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCNDRroOzqIUDFQAAAAAdAAAAABAE

[30] D. W. Apley and J. Zhu, "Visualizing the effects of predictor variables in black box supervised learning models," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 82, no. 4, pp. 1059–1086, 2020.

[31] I. Piatrenka and M. Rusek, "Quantum variational multi-class classifier for the iris data set," in *International Conference on Computational Science*. Springer, 2022, pp. 247–260.

[32] A. Baughman, K. Yogaraj, R. Hebbar, S. Ghosh, R. U. Haq, and Y. Chhabra, "Study of feature importance for quantum machine learning models," *arXiv preprint arXiv:2202.11204*, 2022.

[33] P. Steinmüller, T. Schulz, F. Graf, and D. Herr, "explainable ai for quantum machine learning," *arXiv preprint arXiv:2211.01441*, 2022.

[34] G. Abdulsalam, S. Meshoul, and H. Shaiba, "Explainable heart disease prediction using ensemble-quantum machine learning approach," *Intell. Autom. Soft Comput*, vol. 36, no. 1, pp. 761–779, 2023.

[35] R. Heese, T. Gerlach, S. Mücke, S. Müller, M. Jakobs, and N. Piatkowski, "Explaining quantum circuits with shapley values: Towards explainable quantum machine learning," *arXiv preprint arXiv:2301.09138*, 2023.

[36] [Online]. Available: https://scikit-learn.org/stable/datasets/toy_dataset.html

[37] [Online]. Available: https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.RealAmplitudes

[38] "Efficientsu2." [Online]. Available: https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.EfficientSU2

[39] [Online]. Available: https://docs.quantum.ibm.com/api/qiskit/0.44/qiskit.algorithms.optimizers.COBYLA

[40] [Online]. Available: https://docs.quantum.ibm.com/api/qiskit/0.19/qiskit.aqua.components.optimizers.SLSQP

[41] [Online]. Available: https://docs.quantum.ibm.com/api/qiskit/0.19/qiskit.aqua.algorithms.VQC

[42] M. Schuld and N. Killoran, "Is quantum advantage the right goal for quantum machine learning?" *PRX Quantum*, vol. 3, no. 3, Jul. 2022. [Online]. Available: http://dx.doi.org/10.1103/PRXQuantum.3.030101

Quantum Machine Intelligence (2025) 7:24
https://doi.org/10.1007/s42484-025-00253-9

RESEARCH ARTICLE

QRLaXAI: quantum representation learning and explainable AI

Asitha Kottahachchi Kankanamge Don[1] · **Ibrahim Khalil**[1]

Received: 16 November 2024 / Accepted: 6 February 2025

Abstract

As machine learning grows increasingly complex due to big data and deep learning, model explainability has become essential to fostering user trust. Quantum machine learning (QML) has emerged as a promising field, leveraging quantum computing to enhance classical machine learning methods, particularly through quantum representation learning (QRL). QRL aims to provide more efficient and powerful machine learning capabilities on noisy intermediate-scale quantum (NISQ) devices. However, interpreting QRL models poses significant challenges due to the reliance on quantum gate-based parameterized circuits, which, while analogous to classical neural network layers, operate in the quantum domain. To address these challenges, we propose an explainable QRL framework combining a quantum autoencoder (QAE) with a variational quantum classifier (VQC) and incorporating theoretical and empirical explainability for image data. Our dual approach enhances model interpretability by integrating visual explanations via local interpretable model-agnostic explanations (LIME) and analytical insights using Shapley Additive Explanations (SHAP). These complementary methods provide a deeper understanding of the model's decision-making process based on prediction outcomes. Experimental evaluations on simulators and superconducting quantum hardware validate the effectiveness of the proposed framework for classification tasks, underscoring the importance of explainable representation learning in advancing QML towards more transparent and reliable applications.

Keywords Quantum machine learning · Quantum representation learning · Interpretable machine learning · Explainable AI

1 Introduction

Machine learning has emerged as a transformative force across various industries, revolutionizing applications in numerous domains. As research advances, machine learning models have grown increasingly complex, enhancing performance and reducing interpretability. These highly intricate models, such as deep neural networks, are often called "black-box" models due to their opacity, making it nearly impossible to understand how they function internally (Bodria et al. 2023). These models are typically built directly from data, and even their developers and data scientists cannot fully explain the inner workings of the reasoning behind specific outputs generated by artificial intelligence (AI) algorithms. Therefore, in addition to focusing on performance, the explainability of machine learning models has gained prominence (Molnar 2020). Explainable artificial intelligence (XAI) seeks to clarify how machine learning models operate, their potential biases, and their overall impact. XAI enhances transparency, fairness, and accuracy, particularly in decision-making processes powered by machine learning (Minh et al. 2022). It is essential for building trust and ensuring responsible AI deployment within organizations (Ahmed et al. 2022). Although a comprehensive review of XAI is beyond the scope of this paper, we refer readers to Minh et al. (2022); Adadi and Berrada (2018), and Ahmed et al. (2022) for further details.

In addition to explainability, another rapidly evolving branch of machine learning is quantum machine learning (QML) (Biamonte et al. 2017; Schuld and Petruccione 2018). Quantum computing has experienced significant advancements in recent years, especially with the rise of noisy intermediate-scale quantum (NISQ) devices (Preskill 2018; Lau et al. 2022). Giant big data companies, pioneers in quantum computing, have made substantial progress in quantum

Ibrahim Khalil contributed equally to this work.

✉ Asitha Kottahachchi Kankanamge Don
asitha.kottahachchi.kankanamge.don@student.rmit.edu.au

Ibrahim Khalil
ibrahim.khalil@rmit.edu.au

[1] School of Computing Technologies, RMIT University, P.O. Box 3000, Melbourne 3000, Victoria, Australia

Published online: 19 February 2025

error correction—a critical factor for achieving fault-tolerant quantum computing and real quantum speedup (Wootton and Loss 2012; Bravyi et al. 2024). Among the various applications of quantum computing, QML has gained attention due to its potential for significant speedup compared to classical machine learning (Schuld and Killoran 2019, 2022; Jerbi et al. 2024).

A key element in QML is the variational quantum classifier (VQC), which utilizes quantum gates to manipulate quantum states. In contrast, a subset of these gates includes trainable variational parameters (Benedetti et al. 2019; Cerezo et al. 2021, 2022). VQCs are the backbone for many QML applications (Schuld et al. 2020; Farhi and Neven 2018). However, VQCs on NISQ devices face several challenges, including the barren plateau issue (Li and Deng 2021), circuit optimization complexity (Jia et al. 2023), and qubit encoding problems (Maheshwari et al. 2022). One of the main challenges in variational quantum classifiers (VQCs) is the requirement of n qubits to encode n features, as each feature must be embedded into a quantum state. This significantly limits the scalability and efficiency of VQCs (Thumwanit et al. 2021; Belis et al. 2021; Yano et al. 2021). This qubit-intensive nature of quantum models makes it particularly challenging to evaluate explainability on simplified real-world datasets, such as benchmark image datasets, due to the high number of features and the associated qubit requirements. To overcome this, quantum representation learning (QRL) has been proposed as an emerging solution in QML research (Frohnert and Nieuwenburg 2024; Rivas et al. 2021; Don and Khalil 2024; Kottahachchi Kankanamge Don et al. 2024), which is the focus of this study.

Despite the potential for significant speedup over classical machine learning algorithms, QML algorithms including QRL models face numerous challenges, including hardware limitations, noise in quantum systems, and the complexity of classical preprocessing (Strobl et al. 2024; Ciliberto et al. 2017; Thakare 2023). These issues often overshadow the focus on speedup and contribute to significant explainability challenges, similar to those encountered in classical deep neural networks (see Fig. 1). The counter-intuitive and highly abstract internal workings of quantum gates and layers in quantum models further exacerbate these challenges (Pira and Ferrie 2024; Heese et al. 2023). Additionally, entanglements within quantum circuits obscure the contributions of individual components to model predictions, underscoring the importance of improving interpretability (Zhang et al. 2024; Rohe et al. 2024). The QRL models consist of several components that contribute to their overall architecture. The first component, the encoder, is a dimensionality reduction quantum circuit that reduces the feature dimension after transforming classical data into quantum states. This encoder circuit comprises a quantum feature map circuit and a variational circuit. The second component is the classifier, which is a VQC that classifies the reduced feature dimensions outputted by the encoder. The classifier also contains a data encoding circuit and a variational circuit but with a different quantum circuit architecture. Both components include a measurement circuit, which converts the quantum states back into classical scalar values. These three components increase the overall complexity and reduce transparency (Cerezo et al. 2021; Hur et al. 2022; Choudhury et al. 2021). And, the first two components make it difficult for quantum machine learning researchers and users of QRL to understand how

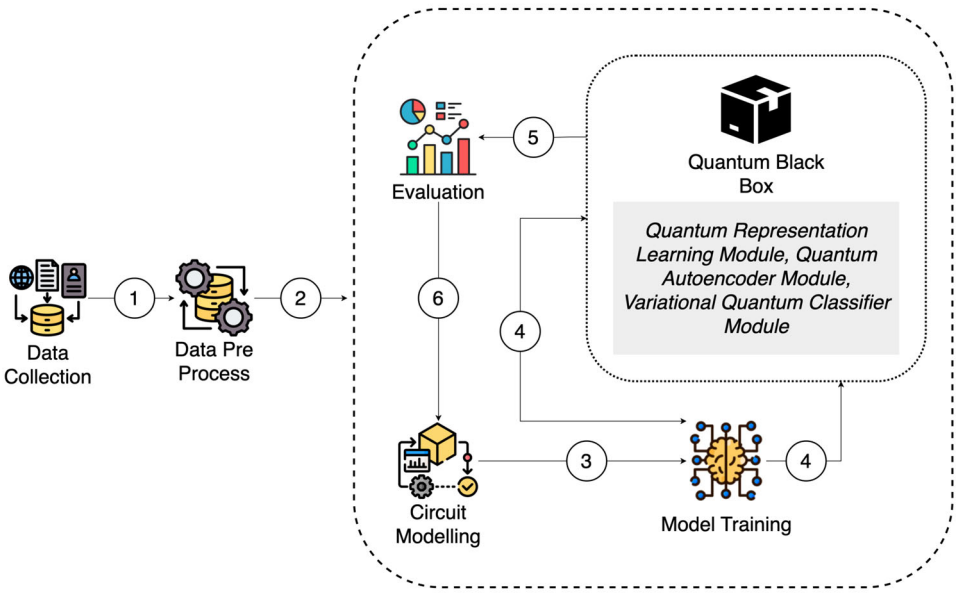


Fig. 1 QRL black-box model: Steps (1) and (2) involve data collection and preprocessing. In step (3), the data is used to model the quantum circuit, followed by step (4), where the quantum model is trained, forming the quantum black box. Finally, in steps (5) and (6), the model is evaluated and fine-tuned, without incorporating explanations for the predictions

predictions are made based on feature representations and quantum gates. Additionally, the measurement component adds further difficulty, as interpreting how quantum circuits are measured and how results are converted back to scalar values is grounded in quantum mechanics, which is not easily intuitive for many users.

However, addressing all transparency issues related to QRL is not a trivial task. The qubit entanglements within quantum circuits add to the complexity of quantum gate operations. Although the measurement process converts quantum states into scalar values, representing the final prediction results of the quantum machine learning model, explaining how each of the three components within a QRL model contributes to the overall prediction remains a challenge. Previous studies have presented various visualization-based explainable approaches for classical deep neural networks (Choo and Liu 2018; Liu et al. 2017; Mohseni et al. 2021; Saranya and Subhashini 2023). Despite their effectiveness in explaining classical deep learning models, adapting these techniques to quantum machine learning models poses significant challenges due to the fundamentally different internal mechanisms between quantum models and classical deep neural networks. In response to this, prior research has proposed various visualization methods specifically designed for quantum neural networks (Choo and Liu 2018; Pira and Ferrie 2024). These studies introduced visual explainability techniques to enhance understanding of various components, such as data encoding circuits, variational circuits, and measurement processes. For example, Choo and Liu (2018); Pira and Ferrie (2024), and Heese et al. (2023) proposed the use of satellite charts and augmented heatmaps to analyze single-qubit states and expectation value measurements. Although several studies propose visual explainability techniques for quantum neural networks, they do not specifically address the explainability and performance of VQC models using a QRL framework. This gap is especially notable when applying model-agnostic explainability methods to benchmark image datasets under low-qubit constraints.

To address this research gap, we propose *QRLaXAI*, a QRL framework designed to collectively explain its two main components: the quantum feature extractor and the quantum classifier, both optimized for low qubit requirements. The quantum feature extractor transforms and compresses classical benchmark image datasets into low-dimensional quantum feature vectors, enabling efficient processing. The quantum classifier makes predictions based on the compressed quantum states generated by the feature extractor. Explanations are then produced using the original image data, the classically derived quantum states from the feature extractor, and the classifier's predicted probabilities, employing two model-agnostic explainability approaches. The first is a simplified breakdown of local interpretable model-agnostic explanations (LIME), which visually explains the step-by-step results of the QRL model based on its predictions. LIME is a technique used to describe how input features influence the predictions of a machine learning model by identifying the region of an image most strongly associated with a prediction label (Štrumbelj and Kononenko 2014). The simplified LIME breakdown in this study allows users to adjust the internal configurations of LIME to generate more detailed explanations. However, this method primarily provides overall model explainability rather than explanations for individual components. The second approach applies Shapley Additive Explanations (SHAP) to assess the classically derived quantum feature importance of the complete model. SHAP is a game-theoretic approach that explains the output of any machine learning model by assigning feature importance based on contribution to the predictions (Lundberg and Lee 2017).

The remainder of this manuscript is organized as follows: Sect. 2 reviews related works. Section 3 describes the proposed architecture in detail. Section 4 presents experimental results demonstrating the effectiveness of our approach in various applications. Finally, Sect. 5 concludes with a summary and potential directions for future research.

2 Related works

In this section, we outline the prerequisites for the proposed *QRLaXAI* method. To this end, we begin by providing a brief overview of classical representation learning, explaining how its explainability is measured and the methods commonly used for this purpose. We then review existing explainability techniques for QML models. Finally, we introduce two state-of-the-art explainability methods, LIME and SHAP, and describe how they are used to measure model explainability.

QRL builds upon classical representation learning, a powerful technique for extracting meaningful representations across various domains, such as computer vision and natural language processing (Chen et al. 2020; He et al. 2020). Classical representation learning methods often excel with fewer model parameters. The state-of-the-art method, supervised contrastive learning (SCL), incorporates a contrastive loss in supervised settings, enabling the network to differentiate between positive and negative pairs and thereby improving its ability to generalize to unseen data (Khosla et al. 2020). In Khosla et al. (2020), variants of the deep neural network were used as encoders, trained using the supervised contrastive loss. During training, 128 projection units were used to produce a 2048-dimensional encoded feature vector, which was then passed through a linear classifier. The study achieved a top-1 accuracy of 81.4% on the ImageNet dataset, surpassing the previously best-reported result for this architecture by 0.8%. Several other studies have also demon-

strated the effectiveness of representation learning, achieving remarkable accuracy (Wang and Qi 2023; Zheng et al. 2021; Hassani and Khasahmadi 2020).

Since classical representation learning has demonstrated promising results, several studies have explored the explainability and interpretability of these models using different techniques. For example, Xu et al. (2022) proposed an SCL-based detection model that utilizes contrast in the representation space to detect novel and unseen deepfake attacks. They further investigate the explainability of the learned features, advocating for score fusion with a deep neural network to enhance variability. Another recent work (Sammani et al. 2024) focused on visual explainability in contrastive learning, which is essential due to its reliance on interacting inputs and data augmentation. This study developed methods for interpreting the model's learning process by analyzing pairs of images. Additionally, the study on self-explaining deep models (Sarkar et al. 2022) introduced an approach that generates concept-based explanations during training, removing the need for post hoc explainability techniques. Despite the advancements, these approaches still exhibit limitations, such as a lack of in-depth feature attribution, over-reliance on task-specific methods, and limited human-centric evaluation and causal explanations. Several studies have proposed model-agnostic explainability methods to explain classical representation learning methods (Zhang et al. 2022; Shahariar et al. 2023; Yuan et al. 2024).

Various studies on explainable classical representation learning have employed diverse explainability techniques, with two main approaches to creating interpretable machine learning models, as outlined by Molnar (2020). The first approach involves designing an intrinsically interpretable predictive model, such as rule-based algorithms, or utilizing a black-box model with a surrogate model to provide explanations. The second approach is known as post hoc interpretability, which can be categorized into global models that explain the average behavior of the black-box model, and local models that explain individual predictions. Several tools are available for creating post hoc interpretable models, most of which are model-agnostic, meaning they can be applied regardless of the underlying algorithm. According to Molnar (2020), the most common post hoc interpretability techniques include partial dependence plot (PDP), accumulated local effects (ALE), individual conditional expectation (ICE), local interpretable model-agnostic explanations (LIME), and Shapley Additive Explanations (SHAP). PDP is a global, model-agnostic interpretation method that shows how individual features contribute to the prediction results of a black-box model (Hastie et al. 2009). However, PDP becomes less effective when models have more than two features, as the plots become harder to interpret. ALE, another global model-agnostic method, provides an alternative to PDP, but can still be biased when variables are highly correlated (Apley and Zhu 2020). While ALE uses a different methodology, it also struggles to capture complex interactions between multiple features, as it only considers a pair of features at a time.

Unlike PDP and ALE, ICE is a local, model-agnostic interpretation method that, while similar to PDP, plots the contribution for each individual instead of the average contribution. However, ICE shares PDP's limitation of struggling with models that have more than two features; as the number of features increases, ICE explanations become difficult to interpret (Adadi and Berrada 2018). Given that our work deals with image datasets, limiting evaluations to a small number of features is not feasible, leading us to discard both PDP, ALE global interpretable methods and the ICE local interpretable method. Next, LIME, as its name suggests, is a local, model-agnostic interpretation method. According to Ribeiro et al. (2016), LIME works by generating a new dataset consisting of perturbed samples and the corresponding predictions of the underlying model for a new observation. An interpretable model is then fitted to this new dataset, weighted by the proximity of the perturbed samples to the original observation. In the case of image classification tasks, LIME identifies the region of an image most strongly associated with a prediction label. SHAP, another local model-agnostic interpretation method, is based on the Shapley value (Lundberg and Lee 2017). Several studies (Štrumbelj and Kononenko 2014; Shrikumar et al. 2019; Datta et al. 2016; Bach et al. 2015; Lipovetsky and Conklin 2001; Lundberg and Lee 2017) have shown that the Shapley value, derived from game theory, explains the prediction by distributing the "gains" (prediction shift from a baseline) among the input features, showing how each feature contributes positively or negatively to the prediction.

In prior research, LIME and SHAP have demonstrated remarkable effectiveness in explaining classification tasks. For instance, Lin et al. (2021) show that LIME effectively highlights patterns in virus datasets and outperforms six other XAI methods by accurately identifying trigger regions. Similarly, Dieber and Kirrane (2020) show LIME's potential for enhancing the interpretability of tabular machine learning models, as demonstrated through performance assessments and user studies. Furthermore, Jeyakumar et al. (2020) suggest that LIME was preferred for text sentiment classification tasks due to its annotation-based explanations. SHAP, on the other hand, has excelled in explaining models for image and text classification tasks. An empirical evaluation (Hailemariam et al. 2020) comparing LIME and SHAP revealed that SHAP slightly outperforms LIME in terms of identity, stability, and separability across deep neural network models on image classification datasets. Several studies have utilized LIME and SHAP together to interpret black-box models for various machine learning tasks, including image classification (Gaspar et al. (2024); Alabi et al. (2023); Ahmed et al.

(2024); Vimbi et al. (2024); Panati et al. (2022)). Therefore, in this study, we decided to leverage LIME and SHAP-based explainability methods to evaluate the explainability of our quantum-supervised contrastive learning model for image classification tasks.

3 Methodology

The *QRLaXAI* method's explainable QRL model consists of a feature extractor and a classifier. The feature extractor is a quantum autoencoder trained with a supervised contrastive learning loss function, while the classifier is a VQC designed for multi-class tasks and optimized using the cross-entropy loss function. After training the feature extractor on image data, its output is used to train the VQC. As shown in Fig. 2, the trained QRL model generates model-agnostic LIME and SHAP-based visual and analytical explanations, including a feature analysis based on its predictions for image data across each dataset.

3.1 Quantum model

Feature extractor The explainable QRL model consists of two main components: the feature extractor and the classifier. The feature extractor employs a quantum autoencoder architecture (Romero et al. 2017), with a quantum data augmentation module (Chalumuri et al. 2022) as its head and a quantum projection head (Khosla et al. 2020) as its tail. This setup enables the transformation of classical image data into quantum data, leveraging quantum gates and entanglements to capture complex features that classical methods may overlook. The output is a reduced-dimensional feature vector representing quantum-transformed information trained using the supervised contrastive loss function.

The quantum data augmentation module, serving as the head of the feature extractor (see Eq. 9), uses 6 qubits-matching the number of latent qubits in the quantum autoencoder-to encode pixel values through $RZ(\theta)$ rotation gates, where θ corresponds to pixel values. Initially, all qubits are set to the $|0\rangle$ state, and the $RZ(\theta)$ gates adjust the qubits based on the image data. By encoding image data into quantum states, this module initiates the quantum processing pipeline, which distinguishes this method from purely classical approaches.

The quantum encoder processes these quantum states through multiple layers, applying quantum gates like $RY(\theta)$ and CX, which introduce entanglement and non-linearity to the feature space. This operation results in compressed quantum feature representations, highlighting the quantum system's ability to handle high-dimensional data efficiently. It begins with an input layer containing an 8-qubit quantum state, which includes the quantum state, a reference state, an auxiliary qubit, and a classical register for measurement. The next layer is a parameterized circuit (see Eq. 11), which applies alternating $RY(\theta)$ rotations and CX entanglements. In the bottleneck layer, 2 qubits initialized in the $|0\rangle$ state, known as "trash states," are excluded, representing successful compression of the input state. The remaining 6 latent qubits capture the core features of the input.

The quantum data augmentation circuit transforms images, which are then fed into the quantum encoder for further processing. At the end of the quantum autoencoder, the projection head—a simple fully connected quantum neural network layer with 64 units—generates the feature vector. Unlike previous studies (Khosla et al. 2020; Don and Khalil 2024) that use more complex configurations, we retain the projection head during classifier training to reduce computational complexity when generating feature vectors. The quantum autoencoder, data augmentation module, and pro-

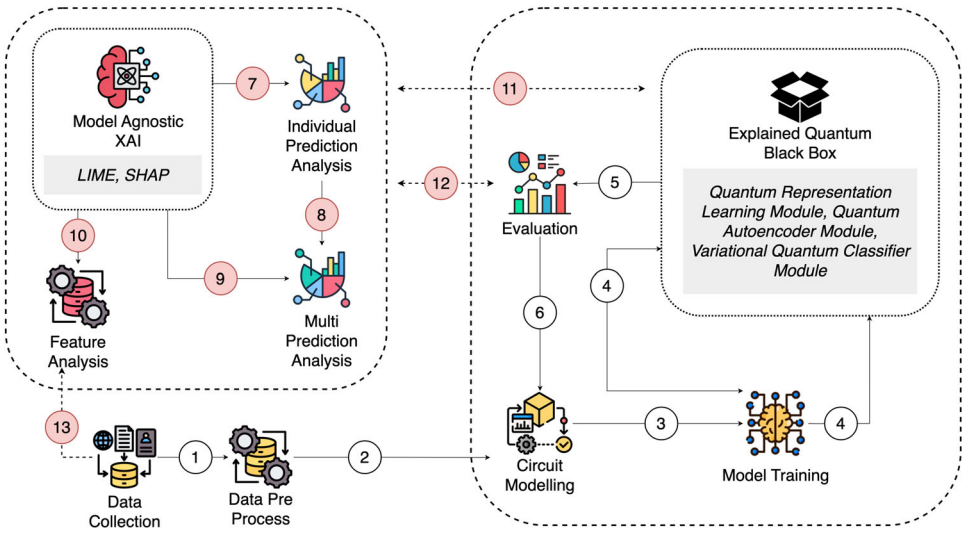


Fig. 2 *QRLaXAI* explainability framework: Steps (7), (8), and (9) generate model-agnostic LIME and SHAP explanations, including individual and multi-prediction explanations. Step (10) involves feature analysis, using the original data from step (13) and QRL model predictions from step (11). Step (12) fine-tunes the QRL model based on the explanation results and re-evaluates model explanations using the fine-tuned QRL model

jection head are all trained using the supervised contrastive loss function (Khosla et al. 2020), which encourages similar data points to cluster in the representation space while separating dissimilar points, thus enhancing classification performance.

The supervised contrastive loss function is defined as follows:

$$L_S(\omega) = -\sum_{\beta \in K} \log \left(\frac{\exp(Z_i \cdot Z_j / T)}{\sum_{n \in S} \exp(Z_i \cdot Z_n / T)} \right) \tag{1}$$

where Z_i and Z_j are the encoded states in the feature space, T is the temperature scaling parameter, and the loss is computed over positive pairs β compared to all instances S. The objective is to minimize this loss, bringing positive pairs closer in the feature space while pushing negative pairs further apart.

The optimization process uses the COBYLA optimizer (IBM 2024), a gradient-free algorithm that iteratively refines linear approximations of the objective function, making it suitable for managing complex constraints and tuning quantum parameters.

Classifier The second component of the QRL model is the VQC (Farhi and Neven 2018; Schuld et al. 2020), which classifies the features extracted by the feature extractor. By converting quantum-transformed feature vectors into a probability distribution over classes, the VQC directly integrates quantum processing into the classification task. The VQC used in this study employs the Special Unitary Group Degree 2 (SU2) variational circuit (see Eq. 12), a hardware-efficient design with 6 qubits that processes the 64 features produced by the feature extractor.

The VQC is trained using the cross-entropy loss function and optimized with the COBYLA algorithm. The cross-entropy loss function is defined as follows:

$$L_C(\omega) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} p_{ij} \log q_{ij} \tag{2}$$

where p_{ij} represents the true label for sample i in class j, q_{ij} is the VQC output for sample i in class j, and $N = 50$ and $C = 10$ represent the number of samples and classes, respectively.

3.2 Explainability framework

3.2.1 LIME breakdown

In this study, we apply the LIME (Ribeiro et al. 2016) method to interpret the QRL model's predictions on individual images. LIME provides local explanations for complex, black-box models by approximating them with interpretable models around specific predictions. As shown in Algorithm 1, given an image $x \in \mathbb{R}^{n \times m}$, where $n \times m$ represents the image's pixel dimensions, LIME generates an explanation by creating perturbed versions of the input image and fitting a locally weighted linear model to approximate the classifier's predictions near the original image.

Algorithm 1 LIME (Ribeiro et al. 2016).

function LIME(f, x, D, K)
  Input:
    f: Classifier model
    x: Data point (image) to be explained
    D: Distribution from which to sample perturbations
    K: Number of perturbed samples to generate
  Output: Surrogate model $\xi(x)$
  $Z \leftarrow$ Sample K points from D ▷ Perturbed samples generated via binary masking
  for $i = 1 \rightarrow K$ **do**
    Generate a binary mask $z_i \in \{0, 1\}^{n \times m}$ ▷ Random pixel removal for perturbation
    Generate perturbed image $x_i' = x \odot z_i$ ▷ Apply the mask
    Compute prediction $g_i = f(x_i')$ ▷ Prediction of f on perturbed sample x_i'
  end for
  Compute Locality and Fit Surrogate Model:
  for each perturbed sample z_i **do**
    Compute proximity weight $\pi_x(z_i) = \exp\left(-\frac{d(x,z_i)^2}{\sigma^2}\right)$ ▷ Gaussian kernel for locality with cosine similarity
  end for
  Fit a weighted linear regression model $\xi(x)$ using (z_i, g_i) pairs, weighted by $\pi_x(z_i)$ ▷ Locally weighted linear model
  return $\xi(x)$
end function

The main steps of the LIME methodology are as follows:

Image segmentation The first step involves segmenting the image into meaningful regions using the QuickShift algorithm (Vedaldi and Soatto 2008), an edge-preserving image segmentation technique that groups pixels based on color and spatial proximity. QuickShift ensures that perturbations affect coherent regions within the image, enhancing the interpretability of the resulting explanations.

Perturbation of input image In this step, we generate a set of perturbed samples by applying random binary masking to the images. For each pixel in the image, a binary mask $z \in \{0, 1\}^{n \times m}$ is created, where a value of 0 masks the pixel and a value of 1 retains it. Applying these masks to the original image x produces a set of perturbed images X', where only a subset of pixels is preserved. This random pixel removal serves as the perturbation strategy (see Eq. 3).

$$X' = \{x \odot z_i : z_i \in \{0, 1\}^{n \times m}, i = 1, \ldots, k\} \tag{3}$$

where k is the number of perturbed samples, and $\odot$ represents element-wise multiplication.

Prediction on perturbed samples The classifier $f(x)$ (see Eq. 4) is then applied to each perturbed sample $x' \in X'$, generating a set of predictions. We use the QRL model to compute these predictions, denoted as follows:

$$f(x') = \{f(x'_i) : x'_i \in X', i = 1, \ldots, k\} \tag{4}$$

These predictions provide the basis for fitting an interpretable model.

Locally weighted linear model To approximate the behavior of the black-box model f near the original image x, we fit a weighted linear regression model $g(z)$ to the predictions. This model is trained on the binary masks z and their corresponding predictions $f(x')$.

The weight of each sample depends on its proximity to the original image, computed using a Gaussian similarity kernel (see Eq. 5). The proximity between a perturbed image and the original image is calculated using cosine similarity, which measures the angle between two non-zero vectors:

$$K(x, x') = \exp\left(-\frac{d(x, x')^2}{\sigma^2}\right) \tag{5}$$

where $d(x, x')$ represents the cosine similarity distance between the original image x and the perturbed sample x', and σ is a scaling factor controlling the width of the kernel.

Explanation generation The explanation is derived by interpreting the coefficients of the linear model $g(z)$ (see Eq. 6). The magnitude and sign of each coefficient indicate the contribution of each pixel to the classifier's prediction. The weighted linear regression coefficients β_i reflect pixel importance:

$$g(z) = \beta_0 + \sum_{i=1}^{n \times m} \beta_i z_i \tag{6}$$

where β_i represents the importance of pixel i in the explanation, and β_0 is the bias term. Pixels with the highest absolute β_i values are identified as the most influential in the classifier's decision. By analyzing the linear approximation $g(z)$, we generate a visual explanation highlighting the pixels that most influence the model's prediction. In this study, we select 50 representative samples from each dataset and generate 150 perturbations per sample to compute the visual LIME explanation for each QRL model prediction, as outlined in QRL-LIME (Algorithm 2). This process involves 150 evaluations per sample, providing a clear visual representation of feature importance.

Algorithm 2 QRL-LIME.

function QRL- LIME(f_{QRL}, x, D, $K = 150$)
  Input:
    f_{QRL}: Quantum Representation Learning model
    x: Data point to be explained
    D: Distribution from which to sample perturbations
    K: Number of perturbed samples to generate ($K = 150$)
  Output: List Γ of surrogate models
  $\Gamma \leftarrow$ Empty list to store surrogate models
  for $i = 1$ to K **do**
    $D_i \leftarrow$ Generate perturbed quantum data locally around x
    $\gamma_i \leftarrow$ LIME(f_{QRL}, x, D_i, K) ▷ Apply LIME to generate a surrogate model (see Alg. 1)
    Add γ_i to Γ
  end for
  return Γ
end function

It is important to note that LIME provides local explanations, meaning the explanation is valid only in the vicinity of the specific instance analyzed. The accuracy of the explanation depends on the number of perturbed samples k and the choice of distance metric in the similarity kernel, both of which warrant further investigation in future studies.

3.2.2 Shapley values

In this section, we extend the SHAP method, based on Shapley values (SVs), to the QRL model. SVs are commonly used to assess the importance of each feature in a model's prediction by treating individual features as players in a coalition game (Hart 1989), with the model's output serving as the value function that quantifies each feature's impact.

Features as players and value function Given an image $x \in \mathbb{R}^N$, where N is the number of features (or dimensions), each feature $i \in \{1, \ldots, N\}$ is treated as a player in the coalition game. The value function $v(S)$ (see Eq. 7), which evaluates the importance of a coalition of features $S \subseteq \{1, \ldots, N\}$, is determined by the output of the classification model f based on these features:

$$v(S) = v(S; x, f) \tag{7}$$

where f is the classification model of interest. This formulation enables the calculation of each feature's contribution by assessing how the model's output changes when the feature is added or removed from the coalition.

Computing Shapley values The Shapley value ϕ_i (see Eq. 8) for each feature $i \in \{1, \ldots, N\}$ is computed by averaging the marginal contributions of the feature across all possible coalitions S that exclude feature i. The Shapley value for

feature i is defined as follows:

$$\phi_i = \sum_{S \subseteq \{1,\ldots,N\} \setminus \{i\}} \frac{|S|!(N - |S| - 1)!}{N!} [v(S \cup \{i\}) - v(S)] \tag{8}$$

where $|S|$ represents the size of the coalition S, and $N!$ is the total number of coalitions. This equation ensures that the Shapley values fairly distribute the model's output contribution among all features.

Approximating Shapley values Calculating exact Shapley values is computationally expensive due to the factorial number of coalitions that require evaluation. To address this, we use the Kernel SHAP approximation (Covert and Lee 2021), which applies a locally linear surrogate model to approximate the value function, significantly reducing computational complexity. The expectation $E[f(X)|x_S]$, where X represents all features not included in coalition S, is estimated by sampling from the dataset. In this study, similar to the LIME approach, we select 50 samples from each dataset to represent feature values and generate 500 perturbation samples to estimate the Shapley values for each prediction. This results in 500 times 50 model evaluations.

3.3 Dataset selection and preparation

Selection This study utilizes four datasets: MNIST (Deng 2012), FMNIST (Xiao et al. 2017), KMNIST (Clanuwat et al. 2018), and CIFAR10 (Krizhevsky 2009), each containing ten classes. Due to quantum hardware constraints, only a subset of classes and samples from each dataset is selected for experimentation. Additionally, images are resized to a 16×16 pixel resolution to fit these limitations. A random sampling method is used to select classes and data samples.

- **Class selection:** Each dataset includes 10 classes. During the random selection process, n classes (where $1 < n < 10$) are initially selected for the experiments, with this procedure repeated five times. In each iteration, previously selected classes are excluded to allow new selections. Due to quantum hardware constraints, we do not evaluate all possible class combinations to prevent bias.
- **Data sample selection:** MNIST, FMNIST, and CIFAR10 contain 60,000 samples each, while KMNIST has 70,000 samples. Given the high cost and inefficiency of processing these volumes on quantum hardware, we select a smaller, manageable set of samples for experimentation, ensuring class balance.

Preparation To train the QRL model, we create incremental data batches, each containing 50 samples from the relevant classes. Starting with two classes, we add one class at a time until reaching 10 classes, resulting in a total of 9 data batches for each dataset. Our primary evaluations focus on the QRL model trained with the first two-class data batch in each dataset. Specifically, the first two classes are digits 0 and 1 for MNIST, T-shirt/top and Trouser for FMNIST, labels 0 and 1 for KMNIST, and airplane and automobile for CIFAR10. For simplicity, we refer to these two classes as "0" and "1" throughout the study. During the multi-class experiments, we evaluate all 9 QRL models trained across the 10 classes in each dataset.

3.4 Experimental setup

To ensure reliable performance, the experimental setup was configured to include both classical counterparts and quantum simulators. These systems were equipped with a 16 GiB NVIDIA T4 Tensor Core GPU to support machine learning inference and graphics-intensive tasks effectively.

For QRL model training, the Amazon Braket IonQ Harmony device (Amazon 2024; IonQ 2024) was utilized. This quantum processor, comprising 11 qubits, operates on a universal gate-model ion-trap architecture. The Universal Gate Set includes quantum gates capable of approximating any unitary transformation to a desired precision. The set consists of Pauli-X, Y, and Z gates, along with the CNOT gate, making it suitable for constructing any quantum circuit. In the ion-trap architecture, qubits are formed using the electronic states of confined ions, which interact via the Coulomb force (Kielpinski et al. 2002). These ions, isolated and trapped to preserve quantum coherence, encode qubit states in their electronic or hyperfine energy levels. Laser beams manipulate the internal energy levels of the trapped ions, enabling single-qubit gate operations. The experiments were implemented and executed using the IBM Qiskit library (Treinish 2023). Access to the IonQ Harmony device was facilitated via the Amazon Braket Provider (Amazon 2024), which allowed the conversion of IBM Qiskit circuits for compatibility with the Harmony platform.

3.5 Code and dataset availability

The code used in this study, along with detailed instructions for reproducing the experiments, is available at the following repository: https://github.com/AsithaIndrajith/qrlaxai. All datasets used in this study—MNIST, FMNIST, KMNIST, and CIFAR10—are publicly available and can be accessed through their respective official sources. The repository also includes preprocessing scripts for preparing the datasets as

described in this paper. By providing access to the code and datasets, we aim to ensure transparency and facilitate further exploration and validation of our proposed framework.

3.6 Quantum gate explainability analysis

This section provides a theoretical explanation of the QRL model's feature extractor, detailing the structure and function of its quantum circuits and the roles of each gate within them.

3.6.1 Feature extractor

The feature extractor in the QRL model includes a quantum data augmentation circuit, a quantum autoencoder (QAE), and a projection head. Together, these components encode classical data into enhanced quantum representations, optimized for quantum-based classification.

Data augmentation circuit The data augmentation circuit applies sequences of SX and R_Z rotations to each qubit, entangled through $CNOT$ gates between adjacent qubits. Its structure is given by the following:

$$\mathrm{DA} = \left(\bigotimes_{i=1}^{n} SX \cdot R_Z\left(\frac{\pi}{2}\right) \cdot SX \cdot R_Z\left(\frac{\pi}{2}\right)\right) \cdot \left(\prod_{k=1}^{n-1} \mathrm{CNOT}(q_k, q_{k+1})\right) \tag{9}$$

The SX gate, or the square root of the X (Pauli-X) gate, is applied to each qubit q_i, placing it into a superposition and enabling each qubit to represent multiple values simultaneously. Each qubit then undergoes an $R_Z\left(\frac{\pi}{2}\right)$ rotation, which shifts its phase around the Z-axis by $\pi/2$, embedding additional information into the quantum state. By combining SX and R_Z operations, the circuit creates a complex representation of the input data. To introduce entanglement, $CNOT$ gates are applied between adjacent qubits q_k (control) and q_{k+1} (target), establishing dependencies that capture interaction patterns and enhance the data-augmented quantum state.

Autoencoder circuit The autoencoder circuit combines an amplitude encoding (AmpEnc) layer with a swap test to evaluate reconstruction quality. The circuit is structured as follows:

$$\mathrm{QAE} = (AE) \cdot \mathrm{Barrier} \cdot \left(H(q_{n+1}) \cdot \prod_{j=1}^{m} \mathrm{CSWAP}(q_{n+1}, q_j, q_{j+m}) \cdot H(q_{n+1})\right) \cdot \mathrm{M}(q_{n+1}) \tag{10}$$

In the amplitude encoding layer, each qubit q_i undergoes a rotation around the Y-axis using the $R_Y(\theta_i)$ gate, parameterized by θ_i. This transformation maps classical data onto the quantum state by modifying each qubit's probability amplitude, an essential step for encoding data into quantum form. Following this, $CNOT$ gates entangle adjacent qubits, creating correlations that capture complex data relationships.

A barrier separates the encoding layer from the swap test, maintaining a logical separation between the two stages.

The swap test begins with a Hadamard gate, $H(q_{n+1})$, applied to an auxiliary qubit q_{n+1}, placing it in superposition to enable controlled swap operations on two sets of qubits. Next, $CSWAP$ gates are applied between pairs of qubits q_j and q_{j+m}, with q_{n+1} serving as the control qubit. These gates swap the states of paired qubits only when the auxiliary qubit is in a particular state, allowing a comparison of two quantum states. After the $CSWAP$ operations, a second Hadamard gate, $H(q_{n+1})$, completes the swap test.

The circuit concludes with a measurement (M) on q_{n+1}, assessing the similarity between the quantum states generated by the encoding layer, effectively verifying the fidelity of the compressed quantum data.

The AmpEnc circuit (IBM 2024a) in the QAE applies parameterized R_Y rotations and entangling $CNOT$ gates, as described by the following:

$$\mathrm{AmpEnc} = \left(\bigotimes_{i=1}^{n} R_Y(\theta_i)\right) \cdot \left(\prod_{k=1}^{n-1} \mathrm{CNOT}(q_k, q_{k+1})\right) \cdot \left(\bigotimes_{i=1}^{n} R_Y(\theta_{i+n})\right) \tag{11}$$

Each qubit is initialized in $|0\rangle$ and undergoes R_Y rotations followed by $CNOT$ gates, transforming the input into an entangled quantum state. This step encodes classical data while preserving information critical for the autoencoder's compression.

3.6.2 Classifier

The VQC in the QRL model uses a SU2 variational circuit (IBM 2024b), which consists of multiple repeated layers. Each layer includes a combination of rotation gates and entangling gates, specifically R_Y, R_Z, and $CNOT$ gates. These layers enable the VQC to capture complex patterns in quantum states, making it suitable for classification tasks.

The SU2 variational circuit is structured as follows:

$$\mathrm{SU2} = \left(\bigotimes_{i=1}^{n} R_Y(\theta_i) R_Z(\theta_{i+n})\right) \cdot \left(\prod_{k=1}^{n-1} \mathrm{CNOT}(q_k, q_{k+1})\right) \tag{12}$$

The R_Y and R_Z gates are parameterized rotation gates applied to each qubit. The $R_Y(\theta_i)$ gate rotates the qubit state around the Y-axis by an angle specified by θ_i, adjusting the quantum representation based on learnable parameters. Following this, the $R_Z(\theta_{i+n})$ gate rotates the qubit state around

the Z-axis, using parameters θ_{i+n}. Together, these gates allow each qubit to explore a broader range of orientations on the Bloch sphere, enhancing the model's ability to capture complex data relationships.

After each qubit undergoes R_Y and R_Z rotations, $CNOT$ gates are applied between adjacent qubits, creating entanglement. A $CNOT$ gate is a two-qubit gate where the first qubit acts as the control and the second as the target. If the control qubit is in state $|1\rangle$, the $CNOT$ gate flips the target qubit's state; otherwise, the target remains unchanged. This entanglement establishes dependencies between the states of neighboring qubits, allowing the circuit to capture interactions between data features.

By repeating these layers, the SU2 circuit iteratively refines the quantum state, optimizing it for classification. The adjustable parameters θ_i are tuned through a training process to minimize classification errors, effectively enabling the circuit to learn from data and perform accurate predictions.

4 Experiments and evaluations

4.1 Evaluation on MNIST, KMNIST, and FMNIST

We performed three types of experiments on the MNIST, KMNIST, and FMNIST datasets. First, we generated LIME visual explanations to interpret prediction results. Second, we utilized SHAP to analyze prediction explanations across all three datasets. For these two experiments, the QRL model, trained with a standardized data preparation step (see Sect. 3.3), achieved accuracies of 90%, 90%, and 75% on MNIST, FMNIST, and KMNIST, respectively, with explanations provided by LIME and SHAP. Finally, we conducted ablation studies to evaluate the impact of various QRL model components on explainability, examining variations in the variational circuits, the number of quantum gates within these circuits, and the effect of varying class counts on the explanations generated by LIME and SHAP.

4.1.1 LIME visual explanation

Image segmentation for LIME explanation The first step in generating LIME explanations involved segmenting the images into coherent regions to facilitate meaningful perturbations. This segmentation was performed using the quick-shift algorithm, an edge-preserving image segmentation technique that groups pixels based on color and spatial proximity. By applying quick-shift, we ensured each segmented region represented a visually consistent part of the image, enhancing the interpretability of the perturbations applied in subsequent steps.

Figures 3, 4, and 5 show two correctly predicted sample images from the 0th and 1st classes in each dataset, illustrating comparisons among the original images, generated superpixels, and segmented images with highlighted boundaries for the QRL model. For each sample, the original image is shown in the first column, the superpixel representation generated by quick-shift appears in the second column, and the segmented image with highlighted boundaries is in the third column. This segmentation approach was applied to 50

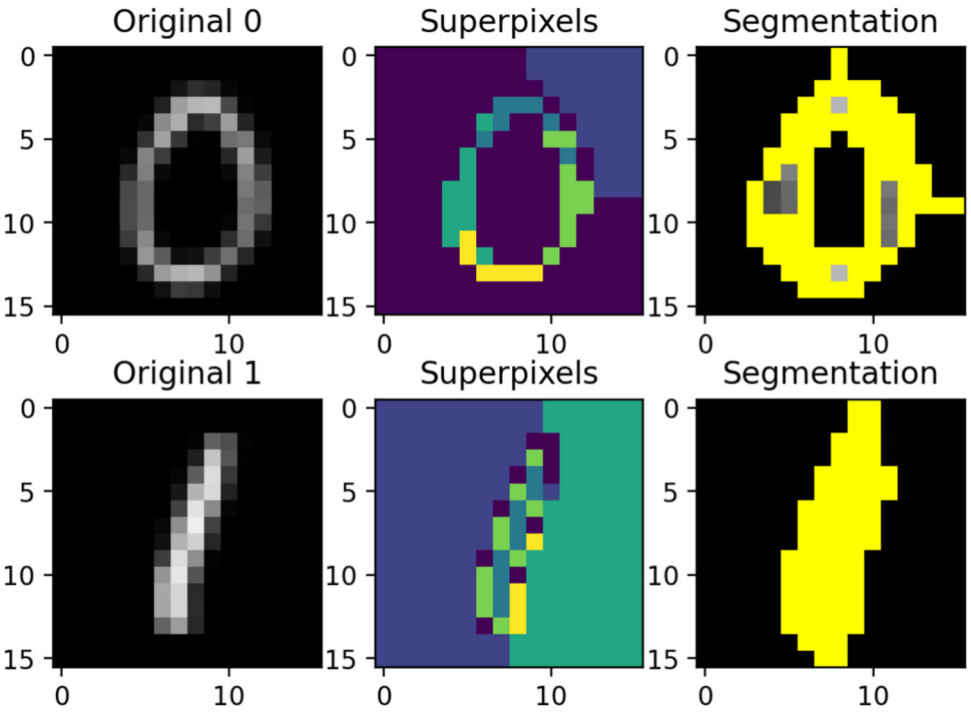


Fig. 3 MNIST image segmentation: The first column shows the original MNIST image, the second displays the quick-shift-generated superpixel representation, and the third presents the segmented image with highlighted boundaries

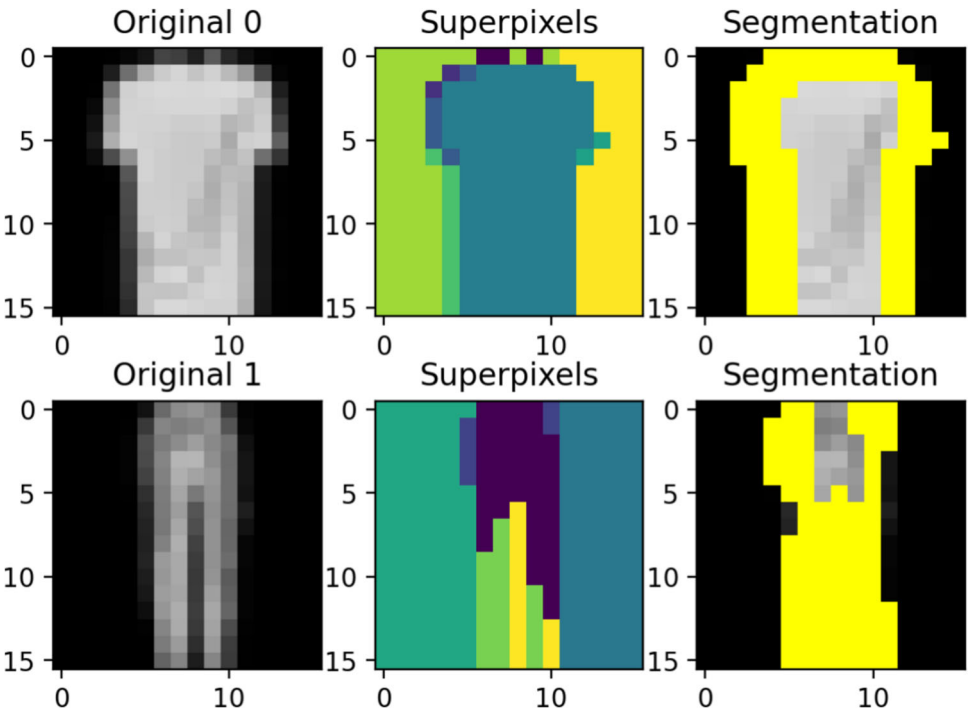


Fig. 4 FMNIST image segmentation: The first column shows the original FMNIST image, the second displays the quick-shift-generated superpixel representation, and the third presents the segmented image with highlighted boundaries

images in each dataset to evaluate the stability and effectiveness of superpixel generation across a variety of samples.

These initial segmentation results confirm that quick-shift successfully identified distinct, coherent regions within each image, allowing the LIME methodology to apply perturbations while maintaining the contextual integrity of each segmented area. This segmentation is crucial for ensuring that the explanation generated by LIME is both interpretable

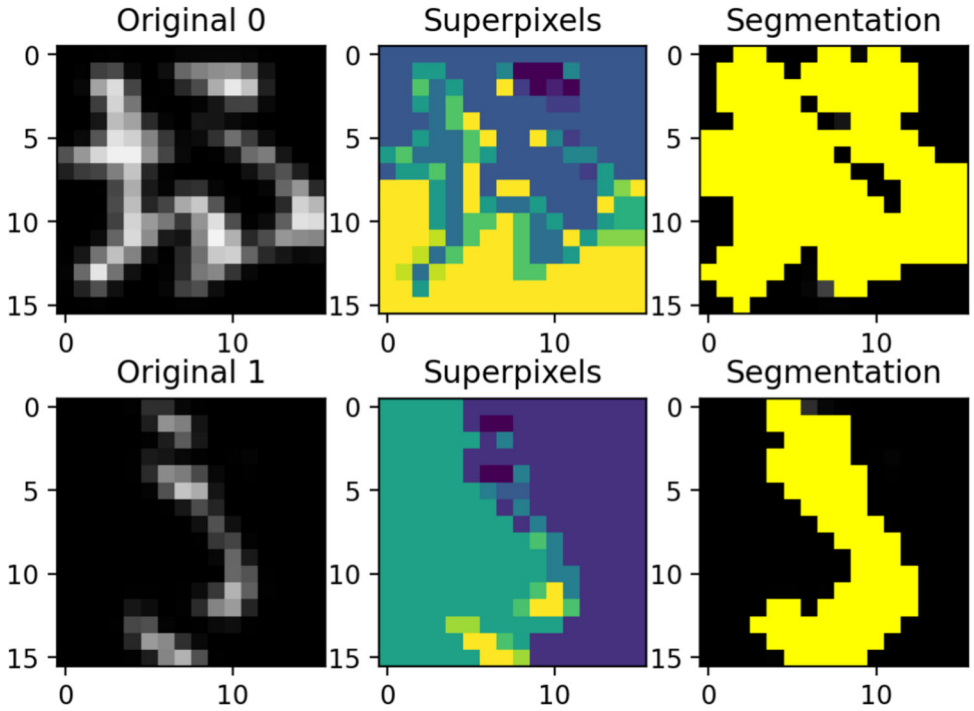


Fig. 5 KMNIST image segmentation: The first column shows the original KMNIST image, the second displays the quick-shift-generated superpixel representation, and the third presents the segmented image with highlighted boundaries

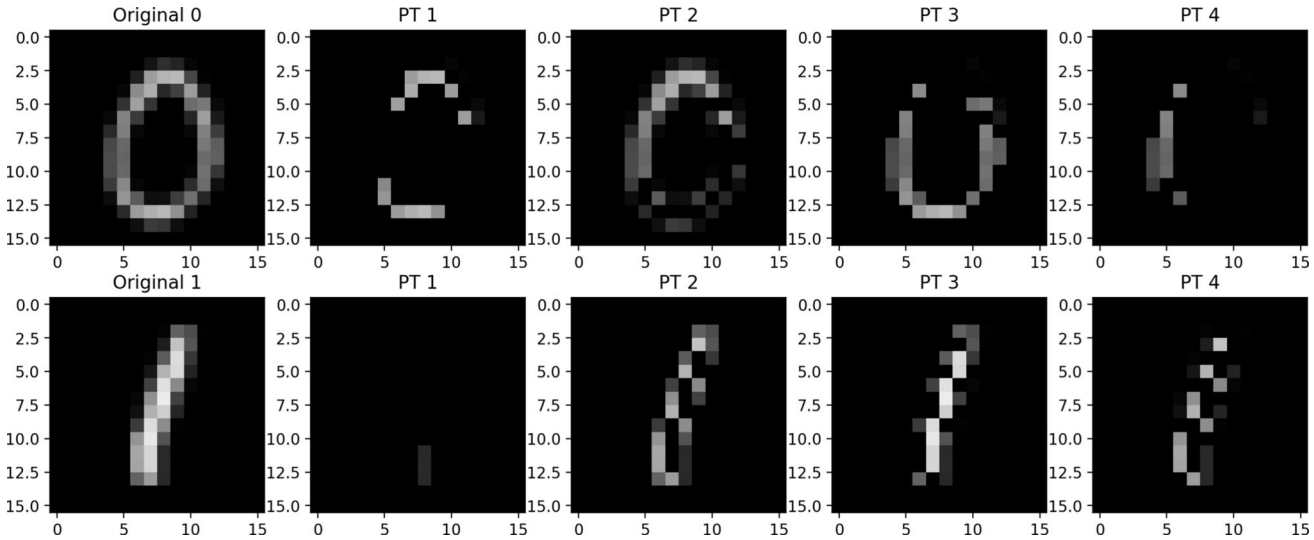


Fig. 6 MNIST perturbations: The first column shows the original MNIST image, followed by four perturbations

and visually consistent, as perturbations are applied to well-defined areas of the image rather than individual pixels.

Perturbation of input image for LIME explanation The second step in generating LIME explanations involved creating a set of perturbed images to facilitate local explanations of model predictions. To achieve this, we applied random binary masking to each image, generating distinct perturbations by selectively masking or retaining certain pixel regions. For each pixel in the image, a binary mask was created, where a value of 0 masked the pixel and a value of 1 retained it. This approach produced perturbed images in which only a subset of pixels is preserved, effectively simulating various occlusion patterns across the image.

In Figs. 6, 7, and 8, we present the same two sample images from each of the three datasets, showing the original image alongside four representative perturbations generated by this method. For each sample image, the original image is shown in the first column, followed by four perturbations in the second through fifth columns. The perturbations illustrate different subsets of visible superpixels, achieved through random binary masking. This random pixel removal constitutes the perturbation strategy, enabling the model to focus on specific image segments while disregarding others.

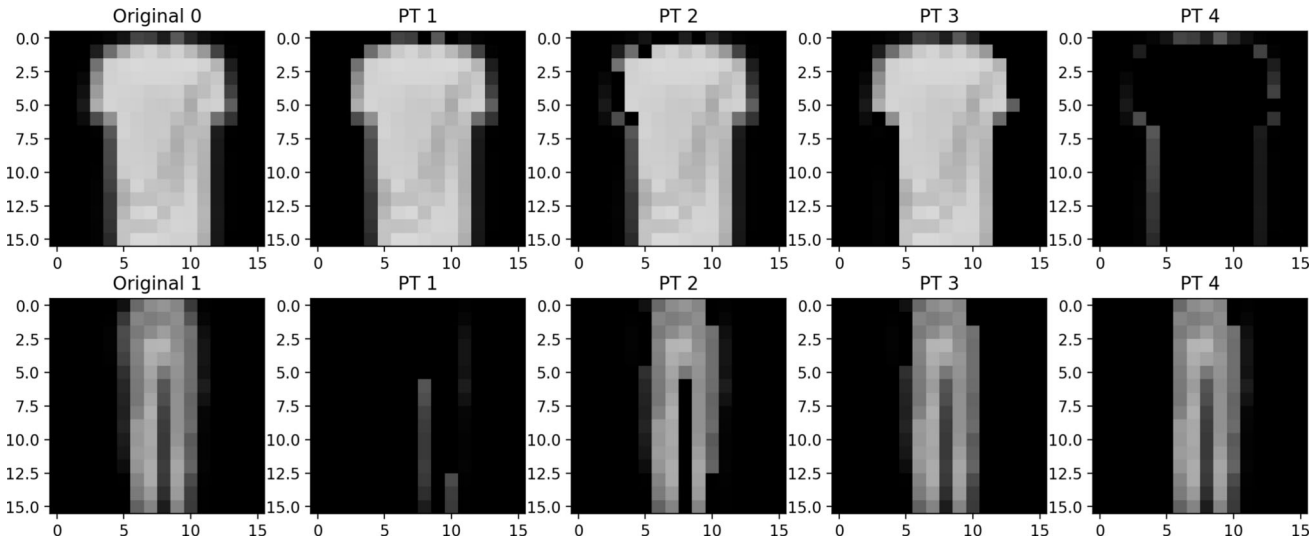


Fig. 7 FMNIST perturbations: The first column shows the original FMNIST image, followed by four perturbations

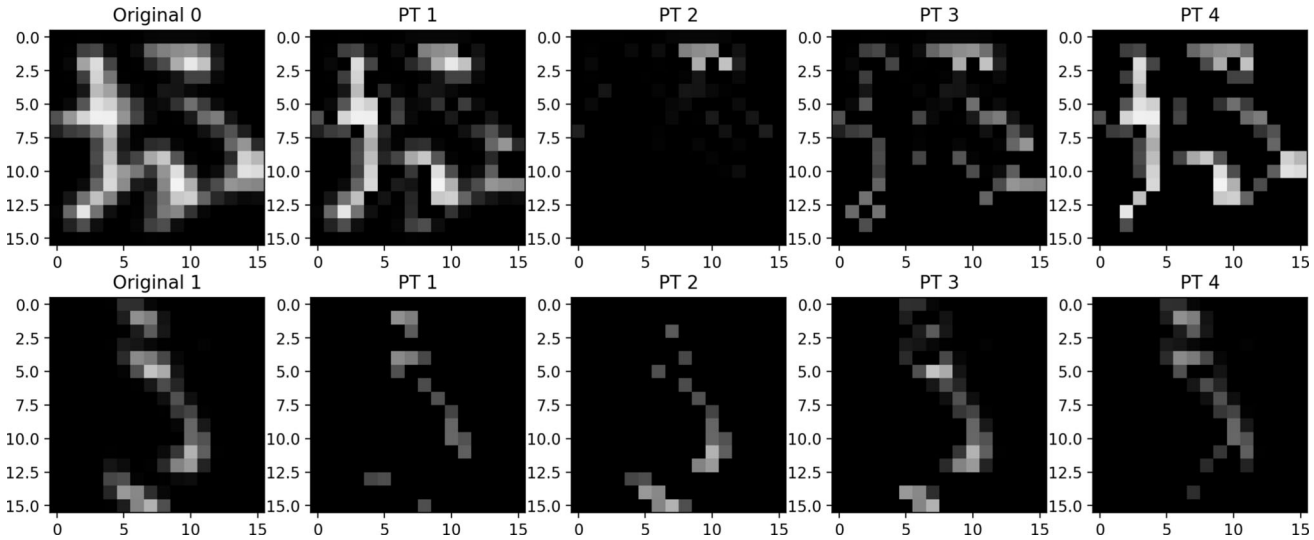


Fig. 8 KMNIST perturbations: The first column shows the original KMNIST image, followed by four perturbations

During this experiment, 150 perturbations were generated for each sample, though only four are displayed here to provide a visual example.

Key feature identification in LIME explanations The third step in the LIME explanation process involved using a locally weighted linear regression model to approximate the behavior of the QRL model around the original image. Each perturbed sample was weighted based on its proximity to the original image using a Gaussian similarity kernel, allowing the model to emphasize features closest to the original image.

Since the model achieved high accuracy on MNIST, FMNIST, and KMNIST, the LIME results indicate that the QRL model effectively learned accurate feature representations for MNIST and FMNIST images. The selected superpixel regions consistently align with visually significant areas in the original images, showing the model's ability to focus on key features contributing to its predictions. However, for KMNIST, the top superpixel regions cover fewer features than in the originals, aligning with the slightly lower accuracy achieved on this dataset. Figure 9 displays the top four superpixel regions identified by LIME across all three datasets for the same two sample images, highlighting the QRL model's robust feature-learning capability. The highlighted regions offer intuitive insights into how specific image areas influence classification decisions.

4.1.2 Shapley values explanation

In this experiment, we extended the SHAP methodology to interpret the QRL model's predictions, using Shapley val-

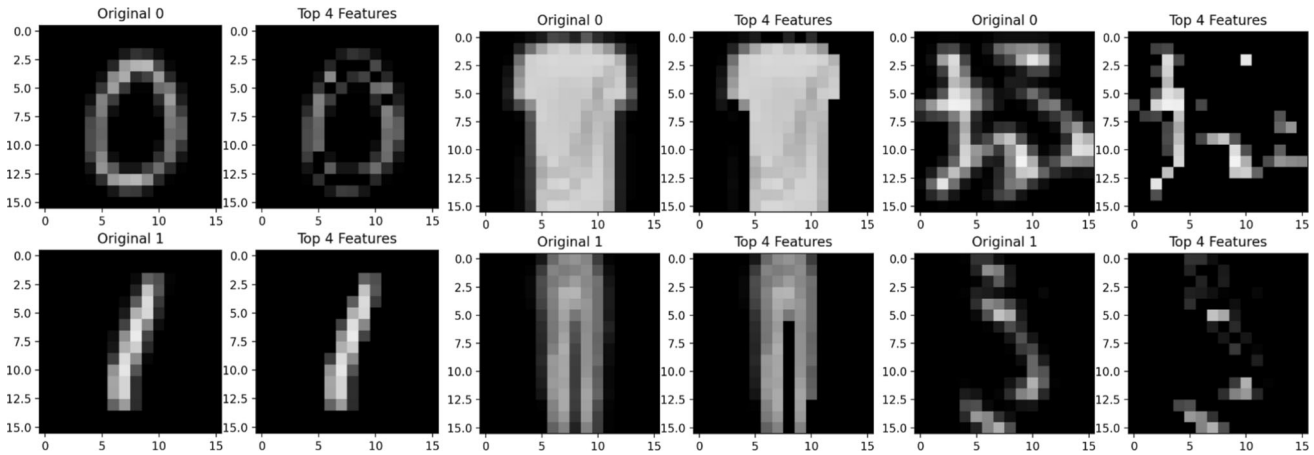


Fig. 9 MNIST, FMNIST, and KMNIST top 4 features: Displays the original image with the top four superpixel regions identified by LIME for all 3 datasets

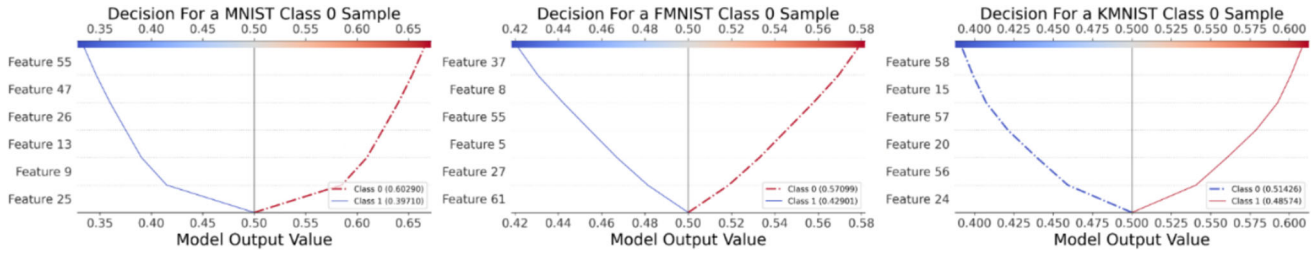


Fig. 10 Individual SHAP Explanations for MNIST, FMNIST, and KMNIST: the magnitude of each top 5 feature's effect on the class 0 sample prediction in each dataset

ues (SVs) to quantify the contribution of individual features to model outputs, as detailed in Sect. 3.2.2. SHAP values reveal how each feature influences the model's predictions, with the model's output acting as the value function that represents the combined impact of these features. After calculating the SHAP values using KernelSHAP, we visualized the results for individual and multiple predictions to understand the QRL model's decision-making process across the three datasets.

Individual prediction explanations For individual samples, we utilized a multi-output decision plot to illustrate the impact of each feature on the model's prediction for a single instance. As shown in Fig. 10, this representation visualizes the SHAP values across multiple classes, indicating how each feature influences the model's output toward or away from each class. The plot structure reveals both the magnitude and direction of each feature's effect on the prediction, facilitating the identification of features that contribute positively or negatively to specific class predictions.

Following a similar approach to the LIME explanations, we selected the same two correctly predicted sample images from the 0th and 1st classes in each dataset and visualized the top 5 features identified using hierarchical clustering based on SHAP values. Figure 10 displays the effect magnitude of each top 5 feature on the class 0 sample prediction across the datasets. The plots illustrate the QRL model's confidence in predicting each sample's class, with the deviation of each prediction from the base value representing the model's confidence level.

Multi-prediction explanations We also produced plots for multiple predictions to provide a broader view of how feature contributions vary across different samples, encompassing both correct and incorrect classifications. This representation plots model output values for different samples against feature contributions, offering a comparative perspective on feature influence across samples. The multi-prediction plot uses a distinct ordering along the feature contribution axis to emphasize different aspects of feature importance. It clusters samples by similarity in feature contributions, allowing for the identification of groups where the model exhibits consistent behavior.

We generated SHAP values for 50 samples. Figure 11 illustrates the effect magnitude of each top 5 feature on predictions for 10 samples in each dataset, with dashed lines indicating misclassifications. Although the QRL model achieved high accuracy across all three datasets (see Sect. 4.1), these plots suggest that the model's confidence level is generally moderate, with predictions deviating around the SHAP base value. Additionally, several misclassifications in the MNIST and FMNIST plots show high confidence lev-

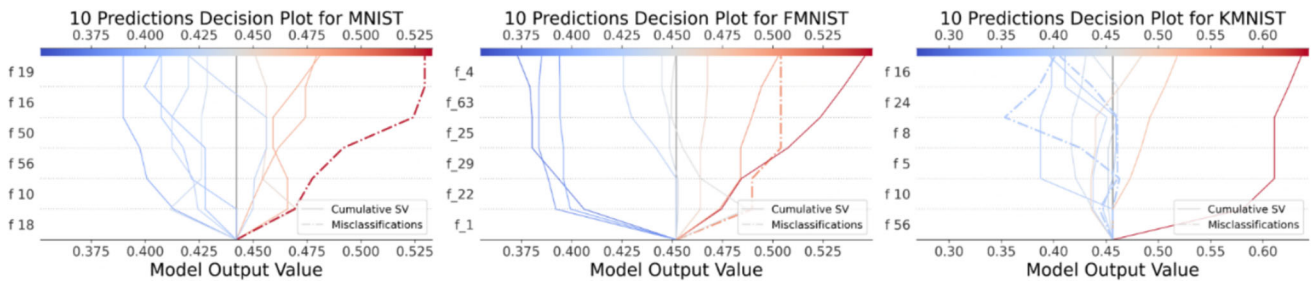


Fig. 11 Multiple SHAP explanations for MNIST, FMNIST, and KMNIST: the magnitude of each top 5 feature's effect on predictions for 10 samples in each dataset, with misclassifications highlighted

Table 1 Effect of rotational layers on QRL model accuracy: This demonstrates the impact of different numbers of rotational layers (RY-4 to RY-10) on the accuracy of the QRL model across three datasets: MNIST, FMNIST, and KMNIST

Dataset	RY-4	RY-5	RY-6	RY-7	RY-8	RY-9	RY-10
Accuracy with rotational layers (%)							
MNIST	80.0	90.0	80.0	85.0	75.0	85.0	90.0
FMNIST	75.0	85.0	80.0	75.0	65.0	90.0	85.0
KMNIST	100.0	70.0	75.0	80.0	80.0	80.0	70.0

els, pointing to potential improvement areas. This analysis demonstrates that SHAP explanations offer deeper insights into the QRL model's predictions compared to LIME. Consequently, in future experiments, we will continue to utilize multi-prediction SHAP explanations alongside LIME visual explanations, as class-wise SHAP explanations for individual predictions provide limited analytical detail for binary classification problems, where only two classes are compared.

4.2 Ablation studies

In the ablation studies, we assessed how different components of the QRL model influence explainability, with a particular focus on the classifier component. We analyzed how variations in the variational circuits, the number of quantum gates within these circuits, and the impact of multi-class classification problems affected the explanations generated by LIME and SHAP.

4.2.1 Impact of rotational layers in the QRL model

The classifier, a variational quantum circuit (VQC) (see Sect. 3.1), uses a variational circuit that allows it to learn complex data patterns by iteratively adjusting its parameters, similar to the weight optimization process in classical neural networks. In this experiment, we investigated how increasing the number of quantum gates in the variational circuit-analogous to adding hidden layers in a deep neural network-impacts the accuracy of the QRL model. As detailed in Eq. 12, the SU2 variational circuit employed in this study consists of layers of R_Y and R_Z rotation gates on each qubit, followed by $CNOT$ gates that entangle adjacent qubits. The rotational gate configuration in the QRL model's classifier was initially repeated four times and then gradually increased to ten repetitions to evaluate its impact on performance.

Table 1 shows that increasing the repetitions of rotational gates does not significantly enhance the QRL model's accuracy across the three datasets. The LIME explanations demonstrate that increasing the number of rotational gates up to ten layers allowed the classifier to consistently identify the top 4 superpixel regions in the original images, similar to the results achieved with four layers, as shown in Fig. 12. SHAP results (see Fig. 13) reinforce this finding, showing that additional rotational gates do not boost the QRL model's confidence level in its predictions. Thus, both LIME and SHAP analyses suggest that adding more rotational gates to the QRL model's classifier does not substantially improve accuracy or confidence in correct predictions, though it does increase computational complexity.

4.2.2 Effect of variational circuit architecture

Since increasing the rotation layer repetitions (R_Y) in the QRL model's classifier showed no clear impact on accuracy, and the explainability studies produced similar explanations, we further analyzed the effect of different variational circuit architectures for the classifier. Specifically, we examined the SU2 (see Eq. 12) variational circuit architecture used throughout the study, along with the Amplitude encoding cir-

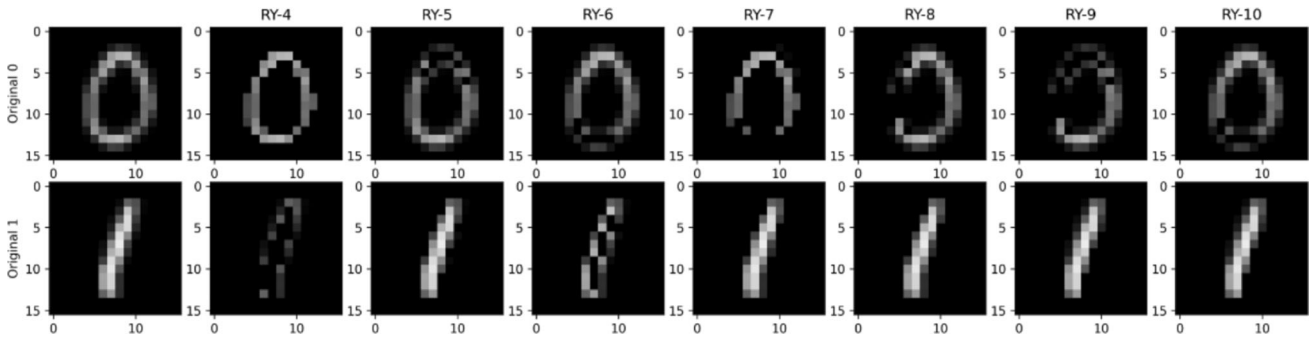


Fig. 12 MNIST top 4 features with rotational layers: The first column displays the original MNIST image, while the remaining columns illustrate the behavior of the top 4 features as rotational layers are incrementally added in the classifier

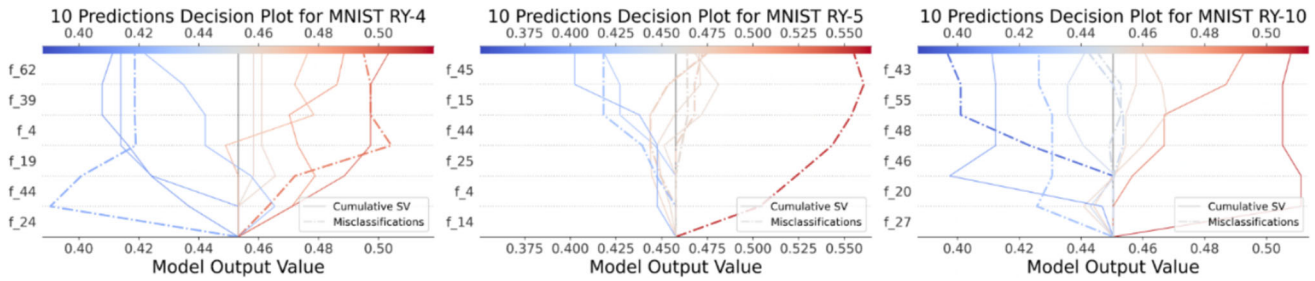


Fig. 13 Multiple SHAP explanations for MNIST with rotational layers: This visualization shows the magnitude of each top 5 feature's effect on 10 predictions for rotational layer counts of 4, 5, and 10

cuit (see Eq. 11) and the local interactions circuit (TL) (IBM 2024).

The TL circuit (see Eq. 13) employs alternating layers of R_Y and R_Z rotations, interspersed with $CNOT$ entanglement between adjacent qubits in a linear configuration.

$$\mathrm{TL} = \left(\bigotimes_{i=1}^{n} R_Y(\theta_i) R_Z(\theta_{i+n})\right) \cdot \left(\prod_{k=1}^{n-1} \mathrm{CNOT}(q_k, q_{k+1})\right) \cdot \left(\bigotimes_{i=1}^{n} R_Y(\theta_{i+2n}) R_Z(\theta_{i+3n})\right) \tag{13}$$

The QRL model's classifiers, based on SU2, AE, and TL variational circuits, achieved the following accuracies: for MNIST, the SU2, AE, and TL circuits reached accuracies of 90%, 80%, and 95%, respectively, with TL exhibiting the highest performance. On FMNIST, the accuracies were 90% for SU2, 75% for AE, and 80% for TL, with SU2 performing best. Finally, for KMNIST, SU2 achieved an accuracy of 75%, outperforming AE and TL, which scored 60% and 55%, respectively.

While the QRL model demonstrated relatively high accuracies across all datasets, the results indicate that varying the circuit architecture (SU2, AE, TL) does not significantly improve the overall accuracy of the QRL model. This finding is consistent with the limited effect observed when increasing the number of rotation layers (see Sect. 4.2.1). This observation is further supported by the LIME and SHAP explanations, as shown in Fig. 14. The LIME explanations for MNIST revealed that the AE and TL circuits captured most of the top superpixel regions similarly to the SU2 variational circuit used in the study. SHAP explanations also

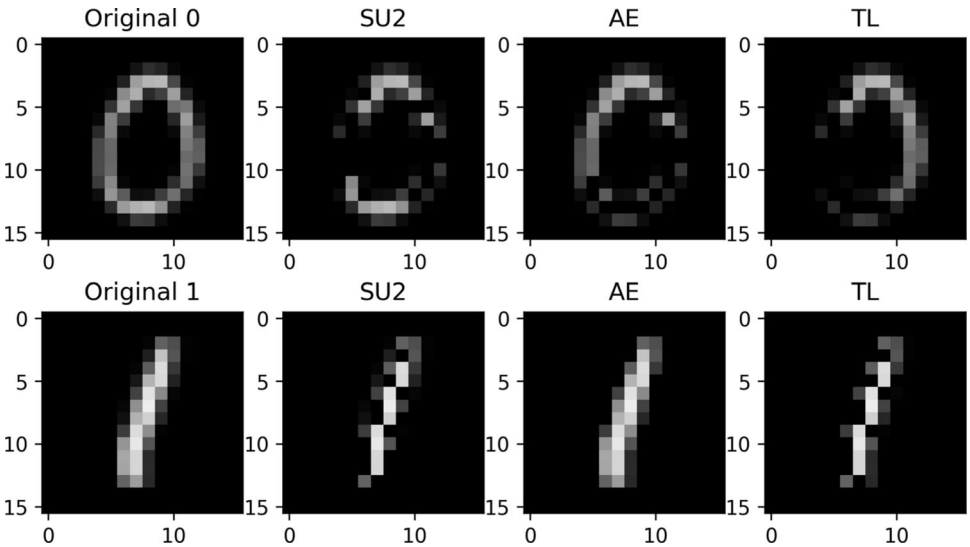


Fig. 14 Top 4 features of MNIST with different variational circuit architectures: The first column shows the original MNIST image, while the remaining columns illustrate the behavior of the top 4 features for SU2, AE, and TL circuits integrated into the classifier

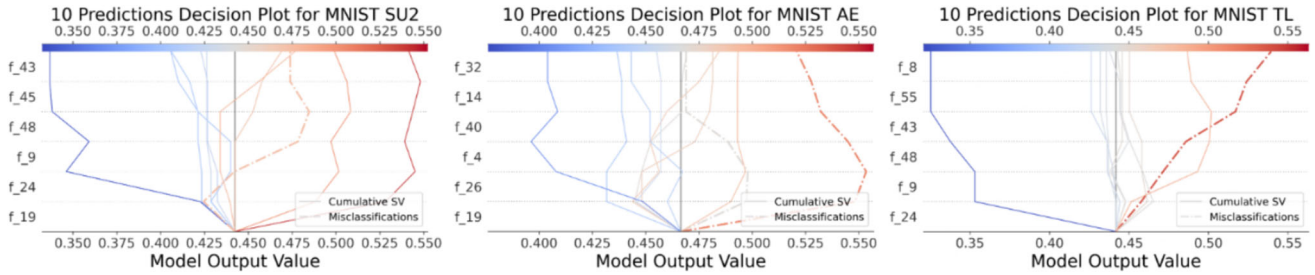


Fig. 15 Multiple SHAP explanations for MNIST with different variational circuit architectures: This visualization shows the magnitude of each top 5 feature's impact on 10 predictions for SU2, AE, and TL circuits within the classifier

verified this result, although the AE and TL circuits exhibited higher-confidence misclassifications compared to the SU2 circuit (see Fig. 15).

4.2.3 Multi-class classification

In addition to analyzing the effects of rotational layers on QRL model explainability (see Sect. 4.2.1) and the impact of variational circuit architecture (see Sect. 4.2.2), both of which focused on QRL model components in binary classification, we conducted experiments to assess the explainability of the QRL model in a multi-class setting. In this experiment, we incrementally trained the QRL model on 2 to 10 classes, following the order of the original class labels and using 50 samples per class across all three datasets.

As shown in Table 2, the accuracy of the QRL model begins to degrade after three classes. This reduction in accuracy results from the increased complexity of encoding multi-class data, which necessitates more advanced quantum state preparation and additional qubits within the VQC. Consequently, this added complexity can lead to greater inaccuracies due to the noise present in NISQ devices.

The LIME explanations reveal that as the QRL model's feature complexity increases with additional classes, the top superpixel regions identified by LIME progressively reduce in size. In Fig. 16, we observe a clear reduction in the intensity of the top 4 superpixel regions in class 1, a trend similarly observed in the other datasets.

In the SHAP explanations (see Fig. 17), the initial plot shows the QRL model's prediction confidence, with a distinct separation between correct predictions and the SHAP base value, indicating high confidence. In the subsequent plot, however, the SHAP values for predictions are closer to the SHAP base value, reflecting decreased confidence in correct predictions and an increase in misclassification instances. The final plot indicates that the QRL model often predicts misclassifications with higher confidence than correct classifications.

4.3 Evaluation on CIFAR10

We extended our experiments to assess the explainability of QRL models on a more complex dataset, CIFAR10. In this experiment, we conducted a binary classification task as outlined in Sect. 3.3. The QRL model achieved an accuracy of 65% on CIFAR10. We then applied both LIME and SHAP to interpret the trained QRL model.

In the LIME explanations, interpreting the top 4 superpixel regions identified proved challenging. Based on the results, it appears that the QRL model did not effectively capture the most important regions of the original image. In Fig. 18, despite applying the correct perturbations, LIME struggled to identify the key pixel regions using the QRL model.

The SHAP explanations, shown in Fig. 19, further illustrate this finding. The plots, which display SHAP values for 10, 20, and 50 predictions, indicate an increase in the

Table 2 QRL model's accuracy across classes: This illustrates the accuracy of classification across increasing numbers of classes (2 to 10) for three datasets: MNIST, FMNIST, and KMNIST

Dataset	2-C	3-C	4-C	5-C	6-C	7-C	8-C	9-C	10-C
Accuracy across classes 2 to 10 (%)									
MNIST	90.0	67.0	43.0	30.0	23.0	19.0	25.0	21.0	19.0
FMNIST	90.0	57.0	30.0	18.0	17.0	13.0	13.0	16.0	13.0
KMNIST	75.0	43.0	23.0	18.0	15.0	21.0	15.0	10.0	15.0

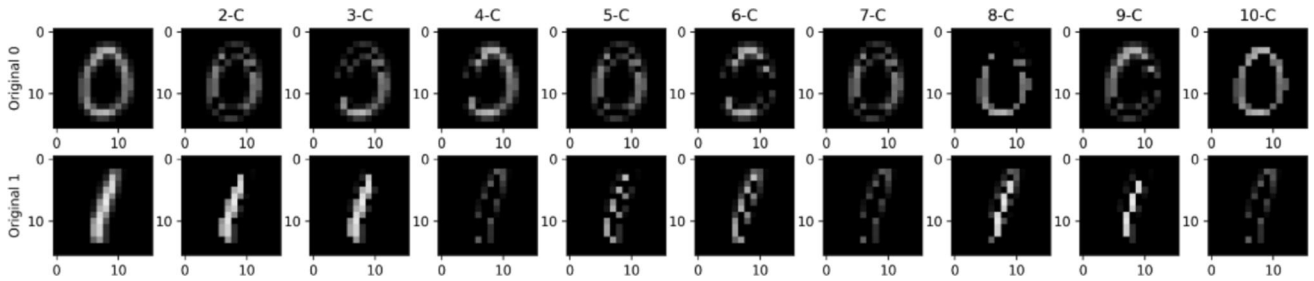


Fig. 16 MNIST top 4 features for multi-class classification: The first column displays the original MNIST image, while the remaining columns show the behavior of the top 4 features across classes 2 to 10

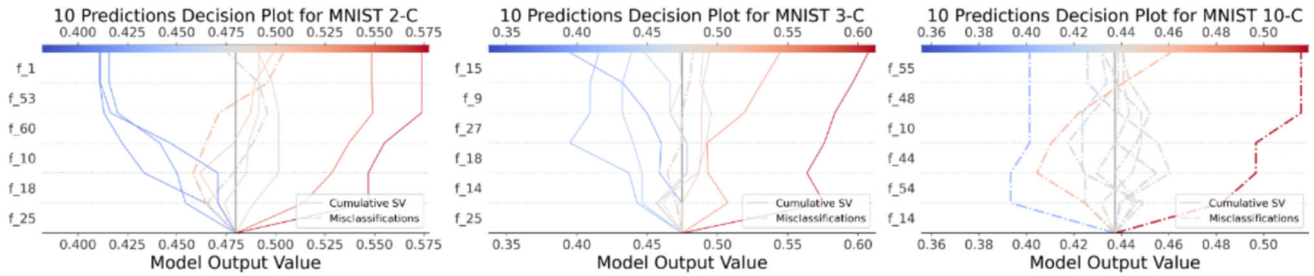


Fig. 17 SHAP explanations for MNIST in multi-class classification: This visualization shows the impact of the top 5 features on predictions for classes 2, 3, and 10

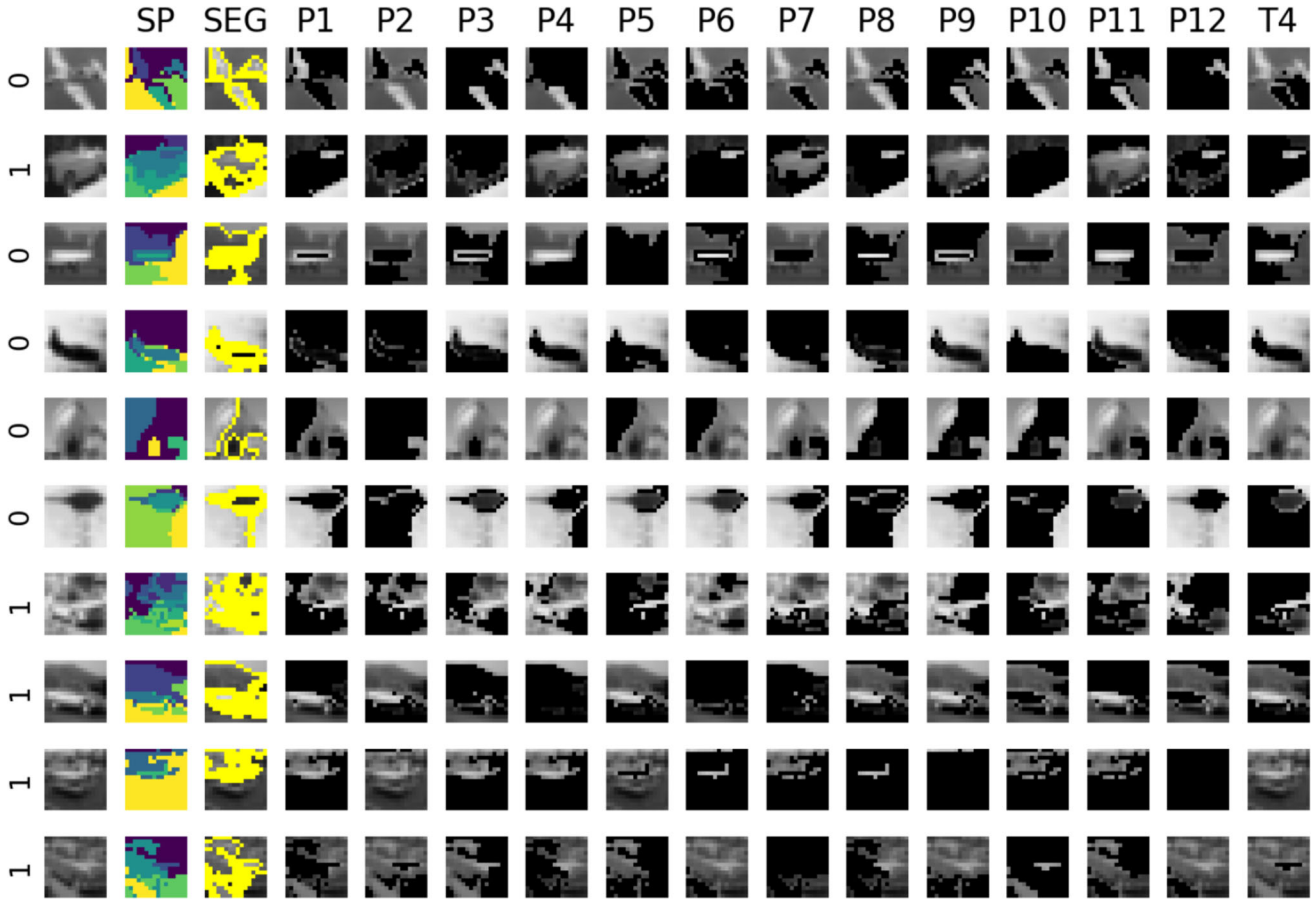


Fig. 18 Top 4 features in CIFAR10 for binary classification: The first column displays the original CIFAR10 image, the second column shows the superpixels, the third column shows the segmentation, columns 4 to 15 show the 12 perturbations generated, and the last column highlights the top 4 features identified by LIME

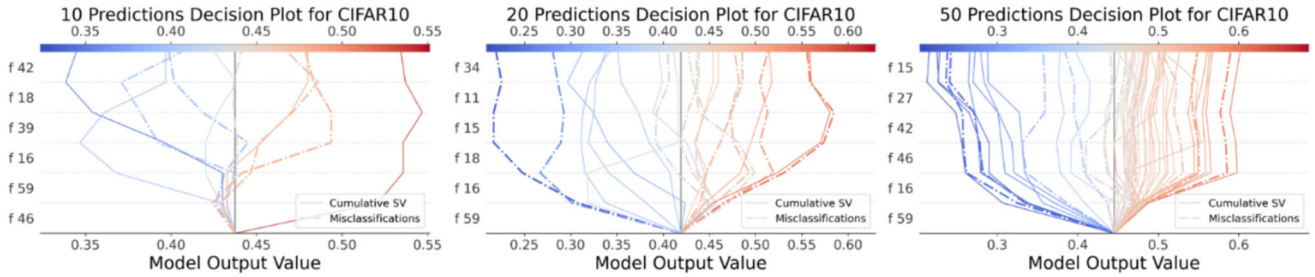


Fig. 19 SHAP explanations for binary classification: From left to right, this visualization shows the effect magnitude of each top 5 feature on 10, 20, and 50 predictions

model's confidence in misclassified predictions. The QRL model exhibited higher confidence in misclassifications than in true predictions, despite achieving an accuracy of 65%.

5 Conclusion

In this study, we introduced QRLaXAI, an explainable framework for quantum representation learning (QRL) that integrates local interpretable model-agnostic explanations (LIME) and Shapley Additive Explanations (SHAP) to enhance the interpretability of variational quantum machine learning models. The framework is designed to operate under low qubit requirements and is evaluated on benchmark image datasets. Our results demonstrate the feasibility of explaining variational quantum classifiers (VQCs) through the combined use of QRL, LIME, and SHAP. We validated QRLaXAI's effectiveness through extensive experiments on MNIST, FMNIST, KMNIST, and CIFAR10 datasets, evaluating the model's explainability across binary and multi-class tasks. The results provided insightful visual and analytical interpretations, making the decision-making process of QRL models more transparent. However, the findings revealed limitations when applied to more complex datasets, such as CIFAR10, where interpretability and accuracy were impacted. Ablation studies further highlighted the sensitivity of QRL model interpretability to quantum circuit design, particularly the number of rotational layers and variational circuit architecture. By processing and explaining quantum-transformed data derived from real-world datasets, QRLaXAI establishes a robust baseline for advancing quantum machine learning explainability research.

Despite these advancements, several limitations remain. This study does not compare QRL models against classical machine learning models, as the primary focus is on validating the reliability and explainability of QRL models. While such comparisons are valuable for understanding the unique advantages of QML interpretability, they are beyond the current scope and will be addressed in future work. We also acknowledge the inherent sensitivity of LIME and SHAP explanations to factors such as data perturbations, model variability, and sampling strategies, which can introduce uncertainties in the generated explanations. These sensitivities are further influenced by quantum noise and gate variability in quantum circuits. Although this study provides initial insights, a deeper statistical analysis, including the use of confidence intervals, uncertainty quantification, and stabilization techniques, is required to fully evaluate the robustness of these explainability methods in quantum contexts.

Future work will focus on several key directions to address these limitations and advance the field of quantum interpretability. These include exploring circuit-level and gate-level explainability to identify the contributions of individual quantum gates and layers, particularly concerning quantum phenomena such as entanglement and superposition. Additionally, we plan to evaluate QRLaXAI on pure quantum datasets, such as the Quantum Chemistry Dataset (QCD), VQE Molecule Dataset, and QAOA Results, to distinguish its capabilities from classical approaches. Comparative analyses with classical machine learning models will be conducted to assess the interpretability advantages of QML and its practical utility. Furthermore, the framework will be tested on real quantum hardware to evaluate its reliability under hardware-specific constraints, such as noise and decoherence. Finally, we aim to enhance the robustness of LIME and SHAP explanations by incorporating advanced statistical tools to quantify and mitigate uncertainties, ensuring more reliable interpretability even in noisy quantum environments.

Acknowledgements We would like to extend our acknowledgment to Robert Shen from RACE (RMIT AWS Cloud Supercomputing Hub) and Jeff Paine from Amazon Web Services (AWS) for their invaluable provision of computing resources.

Author contribution A.K.K.D. (Asitha Kottahachchi Kankanamge Don) conceptualized and developed the methodology, conducted all experiments, performed data analysis, and wrote the main manuscript text. I.K. (Ibrahim Khalil) provided guidance, supervision, and critical feedback throughout the research and manuscript preparation. All authors reviewed and approved the final manuscript.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions. This work is supported by the Australian Research Council Discovery Project (DP210102761).

Data availability No datasets were generated or analyzed during the current study.

Declarations

Conflict of interest The authors declare no competing interests.

References

(2024) Amazon: Qiskit provider for Amazon Braket. https://aws.amazon.com/blogs/quantum-computing/introducing-the-qiskit-provider-for-amazon-braket/. Accessed 12 Jan 2024

(2024) Amazon: quantum cloud computing service - Amazon Braket - AWS. https://aws.amazon.com/braket/. Accessed 14 Sept 2024

(2024) IBM: COBYLA. https://docs.quantum.ibm.com/api/qiskit/0.26/qiskit.algorithms.optimizers.COBYLA. Accessed 19 Feb 2024

(2024) IBM: TwoLocal. https://docs.quantum.ibm.com/api/qiskit/docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.realamplitudes. Accessed 15 Nov 2024

(2024) IonQ: IonQ harmony. https://ionq.com/quantum-systems/harmony. Accessed 29 Sept 2023

(2024a) IBM: RealAmplitudes. https://docs.quantum.ibm.com/api/qiskit/docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.realamplitudes. Accessed 15 Nov 2024

(2024b) IBM: EfficientSU2. https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.EfficientSU2. Accessed 12 Jan 2024

Adadi A, Berrada M (2018) Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). IEEE Access 6:52138–52160. https://doi.org/10.1109/ACCESS.2018.2870052. Accessed 10 Jun 2024

Ahmed I, Jeon G, Piccialli F (2022) From artificial intelligence to explainable artificial intelligence in Industry 4.0: a survey on what, how, and where. IEEE Trans Ind Inf 18(8):5031–5042. https://doi.org/10.1109/TII.2022.3146552. Accessed 10 Jun 2024

Ahmed S, Shamim Kaiser M, Hossain MS, Andersson K (2024) A comparative analysis of LIME and SHAP interpreters with explainable ML-based diabetes predictions. IEEE Access 1–1. https://doi.org/10.1109/ACCESS.2024.3422319. Accessed 13 Oct 2024

Alabi RO, Elmusrati M, Leivo I, Almangush A, Mäkitie AA (2023) Machine learning explainability in nasopharyngeal cancer survival using LIME and SHAP. 13(1):8984. Nature Publishing Group. https://doi.org/10.1038/s41598-023-35795-0. Accessed 13 Oct 2024

Apley DW, Zhu J (2020) Visualizing Eff predictor variables black box supervised learn models 82(4):1059–1086. https://doi.org/10.1111/rssb.12377. Accessed 13 Oct 2024

Bach S, Binder A, Montavon G, Klauschen F, Müller KR, Samek W (2015) Pixel-wise explanations non-linear classifier decis layer-wise relevance propagation. 10(7):0130140. https://doi.org/10.1371/journal.pone.0130140. Public Library of Science. Accessed 10 Oct 2024

Belis V, González-Castillo S, Reissel C, Vallecorsa S, Combarro EF, Dissertori G, Reiter F (2021) Higgs Anal Quant Classifiers 251:03070. EDP Sciences. https://doi.org/10.1051/epjconf/202125103070. Accessed 17 Oct 2024

Benedetti M, Lloyd E, Sack S, Fiorentini M (2019) Parameterized Quant Circ Mach Learn Models 4(4). IOP Publishing. https://doi.org/10.1088/2058-9565/ab4eb5. Accessed 06 Oct 2024

Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N, Lloyd S (2017) Quantum machine learning. Nature Publishing. Group. 549(7671):195–202. https://doi.org/10.1038/nature23474. Accessed 10 Jun 2024

Bodria F, Giannotti F, Guidotti R, Naretto F, Pedreschi D, Rinzivillo S (2023) Benchmarking and survey of explanation methods for black box models 37(5):1719–1778. https://doi.org/10.1007/s10618-023-00933-9. Accessed 10 Jun 2024

Bravyi S, Cross AW, Gambetta JM, Maslov D, Rall P, Yoder TJ (2024) High-threshold and low-overhead fault-tolerant quantum memory. 627(8005):778–782. Nature Publishing Group. https://doi.org/10.1038/s41586-024-07107-7. Accessed 06 Oct 2024

Cerezo M, Arrasmith A, Babbush R, Benjamin SC, Endo S, Fujii K, McClean JR, Mitarai K, Yuan X, Cincio L, Coles PJ (2021) Variational Quant Algorithm 3(9):625–644. Nature Publishing Group. https://doi.org/10.1038/s42254-021-00348-9. Accessed 06 Oct 2024

Cerezo M, Verdon G, Huang HY, Cincio L, Coles PJ (2022) Challenges Oppor Quant Mach Learn 2(9):567–576. https://doi.org/10.1038/s43588-022-00311-3. Accessed 06 Oct 2024

Chalumuri A, Kune R, Kannan S, Manoj BS (2022) Quantum–classical image processing for scene classification. 6(6):1–4. https://doi.org/10.1109/LSENS.2022.3173253 . Accessed 24 Sept 2023

Chen T, Kornblith S, Norouzi M, Hinton G (2020) A simple framework for contrastive learning of visual representations. In: Proceedings of the 37th international conference on machine learning, PMLR, ???, pp 1597–1607. https://proceedings.mlr.press/v119/chen20j.html Accessed 10 Aug 2023

Choo J, Liu S (2018) Visual analytics for explainable deep learning. IEEE Comput Graph Appl 38(4):84–92. https://doi.org/10.1109/MCG.2018.042731661. Accessed 08 Oct 2024

Choudhury S, Dutta A, Ray D (2021) Chaos and complexity from quantum neural network. A study with diffusion metric in machine learning. 2021(4):138. https://doi.org/10.1007/JHEP04(2021)138. Accessed 08 Oct 2024

Ciliberto C, Herbster M, Ialongo AD, Pontil M, Rocchetto A, Severini S, Wossnig L (2017) Quantum machine learning: a classical perspective. Proc Math Phys Eng Sci 474. https://doi.org/10.1098/rspa.2017.0551

Clanuwat T, Bober-Irizar M, Kitamoto A, Lamb A, Yamamoto K, Ha D (2018) Deep learning for classical Japanese literature. https://doi.org/10.48550/arXiv.1812.01718. Accessed 16 Nov 2024

Covert I, Lee SI (2021) Improving KernelSHAP: practical Shapley Value estimation using linear regression. In: Proceedings of The 24th international conference on artificial intelligence and statistics, PMLR, ???, pp 3457–3465. ISSN: 2640-3498. https://proceedings.mlr.press/v130/covert21a.html. Accessed 15 Nov 2024

Datta A, Sen S, Zick Y (2016) Algorithmic transparency via quantitative input influence: theory and experiments with learning systems. In: 2016 IEEE symposium on security and privacy (SP), pp

598–617. https://doi.org/10.1109/SP.2016.42. ISSN: 2375-1207. https://ieeexplore.ieee.org/document/7546525. Accessed 10 Oct 2024

Deng L (2012) The MNIST database of handwritten digit images for machine learning research [best of the web]. 29(6):141–142. https://doi.org/10.1109/MSP.2012.2211477. Accessed 24 Sept 2023

Dieber J, Kirrane S (2020) Why model why? Assessing the strengths and limitations of LIME. Accessed 13 Oct 2024

Don AKK, Khalil I (2024) Q-SupCon: quantum-enhanced supervised contrastive learning architecture within the representation learning framework. https://doi.org/10.1145/3660647 . Accessed 13 Oct 2024

Farhi E, Neven H (2018) Classification with quantum neural networks on near term processors. https://doi.org/10.48550/arXiv.1802.06002. Accessed 06 Oct 2024

Frohnert F, Nieuwenburg EV (2024) Explainable Represent Learn small Quant S 5(1). IOP Publishing. https://doi.org/10.1088/2632-2153/ad16a0. Accessed 17 Oct 2024

Gaspar D, Silva P, Silva C (2024) Explainable AI for intrusion detection systems: LIME and SHAP applicability on multi-layer perceptron. IEEE Access 12:30164–30175. https://doi.org/10.1109/ACCESS.2024.3368377. Accessed 13 Oct 2024

Hailemariam Y, Yazdinejad A, Parizi RM, Srivastava G, Dehghantanha A (2020) An empirical evaluation of AI deep explainable tools. In: 2020 IEEE globecom workshops (GC Wkshps, pp 1–6. https://doi.org/10.1109/GCWkshps50303.2020.9367541. Accessed 13 Oct 2024

Hart S (1989) Shapley value. In: Eatwell J, Milgate M, Newman P (eds) Game Theory, Palgrave Macmillan UK, ???, pp 210–216. https://doi.org/10.1007/978-1-349-20181-5sps25. Accessed 12 Oct 2024

Hassani K, Khasahmadi AH (2020) Contrastive multi-view representation learning on graphs. In: Proceedings of the 37th international conference on machine learning. ICML'20, JMLR.org, ???, vol 119, pp 4116–4126

Hastie T, Tibshirani R, Friedman J (2009) Boosting and additive trees, Springer, New York, NY, pp 337–387. https://doi.org/10.1007/978-0-387-84858-7sps10

Heese R, Gerlach TT, Mücke S, Müller S, Jakobs M, Piatkowski N (2023) Explaining quantum circuits with shapley values: Towards explainable quantum machine learning. https://doi.org/10.48550/arXiv.2301.09138, https://publica.fraunhofer.de/handle/publica/445397

He K, Fan H, Wu Y, Xie S, Girshick R (2020) Momentum contrast for unsupervised visual representation learning. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR), IEEE, ???, pp 9726–9735. https://doi.org/10.1109/CVPR42600.2020.00975. https://ieeexplore.ieee.org/document/9157636/. Accessed 10 Aug 2023

Hur T, Kim L, Park DK (2022) Quant Convolutional Neural Netw Class Data Classif 4(1):3. https://doi.org/10.1007/s42484-021-00061-x. Accessed 08 Oct 2024

Jerbi S, Gyurik C, Marshall SC, Molteni R, Dunjko V (2024) Shadows Quant Mach Learn 15(1):5676. Nature Publishing Group. https://doi.org/10.1038/s41467-024-49877-8. Accessed 13 Nov 2024

Jeyakumar JV, Noor J, Cheng YH, Garcia L, Srivastava M (2020) How can i explain this to you? an empirical study of deep neural network explanation methods. In: Advances in neural information processing systems, Curran Associates, Inc., ???, vol 33. pp 4211–4222. https://proceedings.neurips.cc/paperspsfiles/paper/2020/hash/2c29d89cc56cdb191c60db2f0bae796b-Abstract.html. Accessed 13 Oct 2024

Jia R, Yang G, Nie M, Liu Y, Zhang M (2023) Automatic optimization of variational quantum algorithm-based classifiers. In: Proceedings of the 2022 5th international conference on artificial intelligence and pattern recognition. AIPR '22, pp 1–7. Association for computing machinery, New York, NY, USA. https://doi.org/10.1145/3573942.3573943

Khosla P, Teterwak P, Wang C, Sarna A, Tian Y, Isola P, Maschinot A, Liu C, Krishnan D (2020) Supervised contrastive learning. In: Advances in neural information processing systems, Curran Associates, Inc., ???, vol 33. pp 18661–18673. https://proceedings.neurips.cc/paper/2020/hash/d89a66c7c80a29b1bdbab0f2a1a94af8-Abstract.html. Accessed 10 Aug 2023

Kielpinski D, Monroe C, Wineland DJ (2002) Architecture for a large-scale ion-trap quantum computer. 417(6890):709–711 (2002). Nature Publishing Group. https://doi.org/10.1038/nature00784. Accessed 12 Jan 2024

Kottahachchi Kankanamge Don A, Khalil I, Atiquzzaman M (2024) A fusion of supervised contrastive learning and variational quantum classifiers. IEEE Trans Consum Electr 70(1):770–779. https://doi.org/10.1109/TCE.2024.3351649. Accessed 13 Oct 2024

Krizhevsky A (2009) Learning multiple layers of features from tiny images, 32–33

Lau JWZ, Lim KH, Shrotriya H, Kwek LC (2022) NISQ computing: where are we and where do we go? 32(1):27. https://doi.org/10.1007/s43673-022-00058-z. Accessed 10 Jun 2024

Li W, Deng DL (2021) Recent advances for quantum classifiers. Sci Chin Phys Mech Astron 65(2):220301. https://doi.org/10.1007/s11433-021-1793-6

Lin YS, Lee WC, Celik ZB (2021) What do you see?: Evaluation of explainable artificial intelligence (XAI) interpretability through neural backdoors. In: Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining, ACM, ???, pp 1027–1035. https://doi.org/10.1145/3447548.3467213. Accessed 13 Oct 2024

Lipovetsky S, Conklin M (2001) Anal Regression Game Theory Approach 17(4):319–330. https://doi.org/10.1002/asmb.446

Liu S, Wang X, Liu M, Zhu J (2017) Towards Better Anal Mach Learn Models Vis Anal Perspect 1(1):48–56. https://doi.org/10.1016/j.visinf.2017.01.006. Accessed 08 Oct 2024

Lundberg SM, Lee SI (2017) A unified approach to interpreting model predictions. In: Advances in neural information processing systems, vol 30. Curran Associates, Inc., ??? (2017). https://papers.nips.cc/paperspsfiles/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html. Accessed 10 Oct 2024

Maheshwari D, Sierra-Sosa D, Garcia-Zapirain B (2022) Variational quantum classifier for binary classification: Real vs synthetic dataset. IEEE Access 10:3705–3715. https://doi.org/10.1109/ACCESS.2021.3139323. Accessed 17 Oct 2024

Minh D, Wang HX, Li YF, Nguyen TN (2022) Explainable Artif Intell Compr Rev 55(5):3503–3568. https://doi.org/10.1007/s10462-021-10088-y. Accessed 10 Jun 2024

Mohseni S, Zarei N, Ragan ED (2021) A multidisciplinary survey and framework for design and evaluation of explainable ai systems. 11(3–4). https://doi.org/10.1145/3387166

Molnar C (2020) Interpretable Machine Learning. Lulu.com, ???. Google-Books-ID: jBm3DwAAQBAJ

Panati C, Wagner S, Brüggenwirth S (2022) Feature relevance evaluation using grad-CAM, LIME and SHAP for deep learning SAR data classification. In: 2022 23rd international radar symposium (IRS), pp 457–462. https://doi.org/10.23919/IRS54158.2022.9904989. ISSN: 2155-5753

Pira L, Ferrie C (2024) Interpretability Quant Neural Netw 6(2):52. https://doi.org/10.1007/s42484-024-00191-y. Accessed 07 Oct 2024

Pira L, Ferrie C (2024) Interpretability Quant Neural Netw 6(2):52. https://doi.org/10.1007/s42484-024-00191-y. Accessed 08 Oct 2024

Preskill J (2018) Quantum computing in the NISQ era and beyond. Quant 2:79. https://doi.org/10.22331/q-2018-08-06-79

Ribeiro MT, Singh S, Guestrin C (2016) "why should i trust you?": Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. KDD '16, Association for Computing Machinery, ???, pp 1135–1144. https://doi.org/10.1145/2939672.2939778. Accessed 13 Nov 2024

Rivas P, Zhao L, Orduz J (2021) Hybrid quantum variational autoencoders for representation learning. In: 2021 international conference on computational science and computational intelligence (CSCI), pp 52–57. https://doi.org/10.1109/CSCI54926.2021.00085, https://ieeexplore.ieee.org/document/9799154. Accessed 17 Oct 2024

Rohe T, Schuman D, Nusslein J, Sunkel L, Stein J, Linnhoff-Popien C (2024) The questionable influence of entanglement in quantum optimisation algorithms. https://doi.org/10.48550/arXiv.2407.17204

Romero J, Olson JP, Aspuru-Guzik A (2017) Quantum autoencoders for efficient compression of quantum data. 2(4):045001. IOP Publishing. https://doi.org/10.1088/2058-9565/aa8072. Accessed 14 Nov 2024

Sammani F, Joukovsky B, Deligiannis N (2024) Visualizing and understanding contrastive learning. IEEE Trans Image Process 33:541–555. https://doi.org/10.1109/TIP.2023.3346295. Accessed 12 Oct 2024

Saranya A, Subhashini R (2023) A Syst Rev Explainable Artif Intell Models Appl Recent Develop Future Trends 7. https://doi.org/10.1016/j.dajour.2023.100230. Accessed 08 Oct 2024

Sarkar A, Vijaykeerthy D, Sarkar A, Balasubramanian VN (2022) A framework for learning ante-hoc explainable models via concepts, IEEE Computer Society, ???, pp 10276–10285. https://doi.org/10.1109/CVPR52688.2022.01004. https://www.computer.org/csdl/proceedings-article/cvpr/2022/694600k0276/1H1l4U6Bu3m. Accessed 13 Nov 2024

Schuld M, Bocharov A, Svore KM, Wiebe N (2020) Circ-centric Quant Classifiers 101(3). American Physical Society. https://doi.org/10.1103/PhysRevA.101.032308. Accessed 06 Oct 2024

Schuld M, Killoran N (2019) Quant Mach Learn Feature Hilbert Spaces 122(4). American Physical Society. https://doi.org/10.1103/PhysRevLett.122.040504. Accessed 06 Oct 2024

Schuld M, Killoran N (2022) Is quantum advantage the right goal for quantum machine learning? 3(3):030101. American Physical Society. https://doi.org/10.1103/PRXQuantum.3.030101. Accessed 06 Oct 2024

Schuld M, Petruccione F (2018) Supervised learning with quantum computers. Quant Sci Technol. Springer, ???. https://doi.org/10.1007/978-3-319-96424-9. Accessed 10 Jun 2024

Shahariar GM, Hasan T, Iqbal A, Uddin G (2023) Contrastive learning for API aspect analysis. In: 2023 38th IEEE/ACM international conference on automated software engineering (ASE), pp 637–648. https://doi.org/10.1109/ASE56229.2023.00064. ISSN: 2643-1572. https://ieeexplore.ieee.org/abstract/document/10298556 Accessed 17 Oct 2024

Shrikumar A, Greenside P, Kundaje A (2019) Learning important features through propagating activation differences. https://doi.org/10.48550/arXiv.1704.02685. Accessed 10 Oct 2024

Strobl M, Kuehn E, Fischer M, Streit A (2024) Improving noisy hybrid quantum graph neural networks for particle decay tree reconstruction. EPJ Web Conf. https://doi.org/10.1051/epjconf/202429512004

Štrumbelj E, Kononenko I (2014) Explaining Prediction Models Individ Predictions Feature contributions. 41(3):647–665. https://doi.org/10.1007/s10115-013-0679-x. Accessed 10 Oct 2024

Thakare PM (2023) Bridging the gap between quantum computing and artificial intelligence. Int J Sci Res Eng Manag. https://doi.org/10.55041/ijsrem27848

Thumwanit N, Lortaraprasert C, Raymond R (2021) Invited: Trainable discrete feature embeddings for quantum machine learning. In: 2021 58th ACM/IEEE design automation conference (DAC), pp 1352–1355. https://doi.org/10.1109/DAC18074.2021.9586190. Accessed 17 Oct 2024

Treinish M (2023) Qiskit 0.44.0. Zenodo. https://doi.org/10.5281/ZENODO.2573505. Accessed 09 Aug 2023

Vedaldi A, Soatto S (2008) Quick shift and kernel methods for mode seeking. In: Forsyth D, Torr P, Zisserman A (eds) Computer Vision – ECCV 2008, Springer, ???, pp 705–718. https://doi.org/10.1007/978-3-540-88693-8sps52

Vimbi V, Shaffi N, Mahmud M (2024) Interpreting Artif Intell Models syst Rev ApplLIME and SHAP Alzheimer's Dis Detect 11(1):10. https://doi.org/10.1186/s40708-024-00222-1. Accessed 13 Oct 2024

Wang X, Qi GJ (2023) Contrastive learning with stronger augmentations. IEEE Trans Patt Anal Mach Intell 45(5):5549–5560. https://doi.org/10.1109/TPAMI.2022.3203630. Accessed 17 Oct 2024

Wootton JR, Loss D (2012) High Threshold Error Correct Surf Code 109(16). American Physical Society. https://doi.org/10.1103/PhysRevLett.109.160503. Accessed 15 Nov 2024

Xiao H, Rasul K, Vollgraf R (2017) Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. https://doi.org/10.48550/arXiv.1708.07747. Accessed 24 Sept 2023

Xu Y, Raja K, Pedersen M (2022) Supervised contrastive learning for generalizable and explainable DeepFakes detection. In: 2022 IEEE/CVF winter conference on applications of computer vision workshops (WACVW), pp 379–389. https://doi.org/10.1109/WACVW54805.2022.00044. ISSN: 2690-621X. https://ieeexplore.ieee.org/document/9707568. Accessed 12 Oct 2024

Yano H, Suzuki Y, Itoh KM, Raymond R, Yamamoto N (2021) Efficient discrete feature encoding for variational quantum classifier. IEEE Trans Quant Eng 2:1–14. https://doi.org/10.1109/TQE.2021.3103050. Accessed 17 Oct 2024

Yuan X, Tian Y, Zhang C, Ye Y, Chawla NV, Zhang C (2024) Graph cross supervised learning via generalized knowledge. In: Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining. KDD '24, Association for Computing Machinery, ???, pp 4083–4094. https://doi.org/10.1145/3637528.3671830. Accessed 17 Oct 2024

Zhang Z, Jang J, Trabelsi C, Li R, Sanner S, Jeong Y, Shim D (2022) ExCon: Explanation-driven supervised contrastive learning for image classification. https://doi.org/10.48550/arXiv.2111.14271. Accessed 13 Nov 2024

Zhang S, Zhou Y, Qin Z, Li R, Du C, Xiao Z, Zhang Y (2024) Machine-learning insights on entanglement-trainability correlation of parameterized quantum circuits

Zheng M, Wang F, You S, Qian C, Zhang C, Wang X, Xu C (2021) Weakly supervised contrastive learning. In: 2021 IEEE/CVF international conference on computer vision (ICCV), pp 10022–10031. https://doi.org/10.1109/ICCV48922.2021.00989. ISSN: 2380-7504. https://ieeexplore.ieee.org/document/9710997 Accessed 17 Oct 2024

Quantum Gradient Class Activation Map for Model Interpretability

Hsin-Yi Lin [*†], Huan-Hsin Tseng [‡§], Samuel Yen-Chi Chen[¶‖], Shinjae Yoo [‡**]
[*] Department of Mathematics and Computer Science, Seton Hall University, South Orange NJ, USA
[‡] Artificial Intelligence & Machine Learning Department, Brookhaven National Laboratory, Upton NY, USA
[¶] Wells Fargo, New York, NY, USA
Email: [†] hsinyi.lin@shu.edu, [§] htseng@bnl.gov, [‖] yen-chi.chen@wellsfargo.com, [**] syjoo@bnl.gov

***Abstract*—Quantum machine learning (QML) has recently made significant advancements in various topics. Despite the successes, the safety and interpretability of QML applications have not been thoroughly investigated. This work proposes using Variational Quantum Circuits (VQCs) for activation mapping to enhance model transparency, introducing the Quantum Gradient Class Activation Map (QGrad-CAM). This hybrid quantum-classical computing framework leverages both quantum and classical strengths and gives access to the derivation of an explicit formula of feature map importance. Experimental results demonstrate significant, fine-grained, class-discriminative visual explanations generated across both image and speech datasets.**

***Index Terms*—Variational quantum circuits, quantum neural networks, gradient-based localization.**

I. Introduction

Recent advances in quantum computing and machine learning (ML) have drawn significant attention, leading to efforts to combine these two fascinating technologies. Although current quantum devices still face challenges due to noise and other imperfections, a hybrid quantum-classical computing framework has been proposed to leverage the strengths of both quantum and classical computing [1]. Variational quantum circuits (VQCs) serve as the foundational elements of this hybrid framework. Within this framework, computational tasks that can benefit from quantum advantages are executed on quantum computers, while others are handled by classical computers. VQC-based algorithms have been shown to have certain advantages over classical models [2]–[4] and have demonstrated success in various ML tasks, including classification [5]–[7], sequential modeling [8], audio and language processing [9]–[11], and reinforcement learning [12]–[14].

Despite these successes, certain aspects of quantum machine learning (QML) models have not been thoroughly investigated, particularly concerning the safety of QML applications. Model interpretability and transparency are crucial for understanding and ensuring proper use, especially due to extensive machine learning applications and the continuing development of policy and regulation. With the rapid development of quantum computing, it is important to explore whether any quantum advantage could be offered in this direction.

Various approaches have been developed to address model interpretability from different aspects, such as from feature importance techniques [15] and rule-based methods [16]. Model-agnostic methods provide local interpretability by approximating complex models with simpler ones [17]–[19].

Among diverse approaches, visual techniques stand out for their intuitive appeal. Methods such as Partial Dependence Plot (PDP) [20] and Individual Conditional Expectation (ICE) [21] visualize the relationship between a feature and the predicted outcome. Class Activation Mapping (CAM) [22] and its variants [23], [24] have emerged as powerful tools for generating class-discriminative localization as visual explanations for Convolutional Neural Networks (CNNs).

In this work, we propose using VQC for activation mapping as a pioneering exploration of quantum circuit applications for model transparency. Our proposed method, Quantum Gradient Class Activation Map (QGrad-CAM), employs a VQC to weigh the importance of activation maps generated from a CNN-based network.

Based on the structure of VQC, we will derive the explicit formula for the importance of each activation map, which can serve as an example of this advantage. Furthermore, experiments are performed on image and speech datasets for validation. Meaningful highlighted regions are returned using the proposed method for all cases.

To summarize, our main contributions include

- Deriving an explicit formula for the VQC importance of activation maps and visualize model decisions.
- Conducting experiments to show the VQC can effectively weigh the importance of activation maps and provide textual explanations for model decisions.

The rest of the paper is organized as follows. In Sec. II, we point out relevant work in previous literature. Our proposed method, QGrad-CAM, is introduced in Sec. III. The details of QGrad-CAM are written in Sec. III-C after a VQC review and discussions of the critical ideas in Sec. III-A and Sec. III-B that motivate our proposal. With the VQC structure, we derive the explicit formula for the importance of feature maps in Sec. IV. The experimental results can be found in Sec. V, and the conclusions in Sec. VI.

II. Related work

Explaining QML Models. VQC-based QML methods have demonstrated significant success in the domain of classification [5]. Noteworthy advancements include the development of Quantum Convolutional Neural Networks (QCNN) [6], [7], quantum transfer learning techniques [25], and hybrid models incorporating tensor networks [26]. Despite these achievements, the explainability of these models has not been thoroughly addressed. Several preliminary attempts have been made to explain the predictions generated by QML models. For instance, the study in [27] investigates feature importance within quantum Support Vector Machine (SVM) models using the Iris dataset, which consists of only four features. The generalizability of this method to larger-scale datasets, such as image data or hybrid models combining classical NNs and VQCs, remains uncertain. Another approach, as outlined in [28], involves calculating the Shapley values for gates in VQCs to determine their respective contributions. The objective of the methods presented in [28] is to rigorously evaluate the quality of various circuit architectures. In contrast, our proposed framework aims to uncover feature importance given a fixed VQC architecture using a scalable gradient-based method.

Visualizing & CNN localization. CNNs have long demonstrated exceptional performance across a wide range of applications. This sparked extensive research aimed at understanding the underlying properties of CNNs. Several works developed techniques for visualizing the CNN-learned latent representation by, for example, analyzing convolution layers [29], [30] and inverting deep features [31]–[33]. The research prompted the discovery of CNNs' ability to localize objects. CAM was proposed in [22], where CNN layers localize objects unsupervised and produce visual explanations for each class. Different pooling methods were explored in [34], [35] with a similar structure as CAM. Grad-CAM [23] generalized CAM by combining the class discriminative property with gradient techniques. The method allowed fine-grained discriminative localization and was improved, especially for multiple instances scenarios in [24].

III. Quantum gradient class-activation map (QGrad-CAM)

A. Variational Quantum Circuits

VQCs also referred to as Parameterized Quantum Circuits (PQCs), are quantum circuits characterized by tunable parameters that can be optimized based on specific metrics or signals [5]. VQCs or PQCs serve as fundamental components of Quantum Neural Networks (QNNs), which underlie current QML techniques. The VQC as a QML method operates n qubits in space $\mathcal{H} = \bigotimes^n \mathbb{C}^2 \cong \mathbb{C}^{2^n}$, where $\mathcal{H}$ is a Hilbert space containing a standard basis written as $\beta = \{|00\cdots0\rangle, |00\cdots1\rangle, \ldots, |11\cdots1\rangle\}$ such that any quantum state $|\psi\rangle \in \mathcal{H}$ can be expanded by β, *i.e.,*

$$|\psi\rangle = c_0 |00\cdots0\rangle + \cdots + c_N |11\cdots1\rangle$$

for some coefficients $c_i \in \mathbb{C}$ with $N = 2^n$. Let $\mathcal{L}(\mathcal{H})$ denotes all linear operators on $\mathcal{H}$ and $\mathcal{U}(\mathcal{H})$ be the collection of unitary operators in $\mathcal{L}(\mathcal{H})$. A VQC typically functions through the following three steps (see Fig. 1),

1) A quantum encoding $V : \mathbb{R}^n \to \mathcal{U}(\mathcal{H})$,
2) A *variational* quantum gate $U(\theta) \in \mathcal{U}(\mathcal{H})$ parameterized by θ,
3) A quantum measurement of Q as an output $\langle\cdot|Q|\cdot\rangle : \mathcal{H} \to \mathbb{R}$.

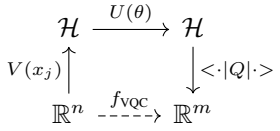


Fig. 1. The diagram of a VQC with a classical input $x_j \in \mathbb{R}^n$ received.

Assume data is of classical form $\mathcal{D} = \{(x_j, y_j) \,|\, x_j \in \mathbb{R}^n, y_j \in \mathbb{R}^m, j = 1, \ldots, N\}$ where x_j is an input of sample index j and y_j be the corresponding label. A quantum encoding scheme chooses a fixed gate sequence $V : \mathbb{R}^n \to \mathcal{U}(\mathcal{H})$ to convert classical data into quantum states such that each input corresponds to a unitary, $x_j \mapsto V(x_j)$. One then associates the quantum state $|\psi_{x_j}\rangle := V(x_j)|\psi_0\rangle$ to data x_j up to a random initial $|\psi_0\rangle \in \mathcal{H}$. For instance, $V(x) = e^{i \tan^{-1}(x)\sigma_k}$ injects data $x \in \mathbb{R}$ into a 1-qubit space in a non-linearly fashion [5]. Typical choices of V include combinations of Hadamard gates, CNOT gates, and gates generated by Pauli matrices $\mathcal{P} = \{I, \sigma_1, \sigma_2, \sigma_3\}$. Subsequently, the encoded state $|\psi_{x_j}\rangle$ is deformed by the parameterized gate $U(\theta)$ such that $|\psi_{x_j}\rangle \mapsto U(\theta)|\psi_{x_j}\rangle$, where θ represents the learnable parameters subject to certain specific optimization routines. It is noted that the deformation ability of VQC majorly comes from $U(\theta)$ where a convention is taking tensor products of the 1-parameter subgroup generated by $\mathcal{P}$,

$$U(\theta) = \prod_{\ell=1}^{k} \left(\bigotimes_{q=1}^{n} e^{-\frac{i}{2}\theta_q^{(\ell)}\sigma_q^{(\ell)}} \right) \circ \mathcal{C}_\ell \in \mathcal{U}(\mathcal{H}) \tag{1}$$

where $q = 1, \ldots, n$ is the qubit index, $\ell = 1, \ldots, L$ is the index of variational circuit layers up to L with each $\sigma_q^{(\ell)} \in \mathcal{P}$, $\mathcal{C}_\ell$ is the unitary of all other non-parameterized gates such as CNOT gates etc, and $\theta = \{(\theta_1^{(\ell)}, \ldots, \theta_n^{(\ell)})\}_{\ell=1}^{L} \in \mathbb{R}^{nL}$ denotes the collection of all variational (learnable) parameters. Often the 1-parameter subgroup $\theta \mapsto e^{-\frac{i}{2}\theta\sigma_k}$ generated by $\sigma_k \in \mathcal{P}$ is denoted as $\{I, R_x(\theta), R_y(\theta), R_z(\theta)\}$ correspondingly.

A final measurement step selects m Hermitian operators $Q_1, \ldots, Q_m$ (each $Q_i \in \mathcal{L}(\mathcal{H})$) to collapse state $U(\theta)|\psi_{x_j}\rangle$ and yields an m-dimensional output vector $y = (\langle Q_1\rangle, \ldots, \langle Q_m\rangle)$ by

$$\langle Q_i\rangle := \langle\psi_0| V^\dagger(x_j) U^\dagger(\theta)\, Q_i\, U(\theta) V(x_j) |\psi_0\rangle \in \mathbb{R} \tag{2}$$

with $i = 1, \ldots, m$. Collectively, the three steps in VQC can be written as $f_{\text{VQC}} : \mathbb{R}^n \to \mathbb{R}^m$ transforming $x_j \mapsto f_{\text{VQC}}(x_j)$, see Fig. 1. By writing $f_{\text{VQC},\theta}$ we emphasize the dependency on θ. We can drop the subscript θ when the context is clear.

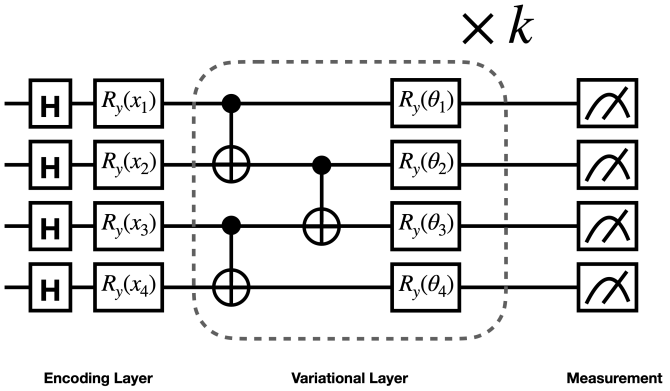


Fig. 2. A circuit representation of the VQC used in this work, Eq. (1). The dashed-line box is repeated k times to increase the depth of the circuit.

Additional components, such as a classical neural network, is possible to be implemented to process raw data into a latent vector of smaller dimensions suitable for a VQC [25], [26], [36]. This technique is particularly useful when the data dimension exceeds the current capabilities of quantum devices or simulators. Furthermore, the output values from the VQC $y = (\langle Q_1 \rangle, \ldots, \langle Q_m \rangle)$ can be refined through further quantum or classical processing. For example, the output from a VQC can be processed by another classical neural network or a VQC. This is useful when the desired values are in a range not provided by the quantum observables and require certain rescaling. A loss function can be chosen as

$$L(F_\phi, G_\eta, f_{\text{VQC},\theta}; \mathcal{D}) = \sum_j d\left(y_j, (G_\eta \circ f_{\text{VQC},\theta} \circ F_\phi)(x_j)\right) \tag{3}$$

where F_ϕ represents the classical pre-processing network, G_η denotes the classical post-processing network, and d is a function measuring the *distance* between the predicted results and the ground truth y_j. The whole hybrid quantum-classical model, including both classical and quantum parameters, can be trained in an end-to-end manner through gradient-based [26] or gradient-free [36] optimization algorithms. The goal is to find the optimal Θ^*, the collection of all quantum and classical trainable parameters $\{\theta, \phi, \eta\}$, by $\Theta^* = \arg\min_\Theta L(F_\phi, G_\eta, f_{\text{VQC},\theta}; \mathcal{D})$.

VQC-based models have been proved to be able to outperform classical NN when certain conditions are met [2], [3]. For example, it is possible to train QML models trained on smaller training dataset [4], [37] while maintaining the generalizability. Empirically, VQC has been shown to be successfully in various ML tasks [5], [6], [8], [9], [12]–[14], [26].

B. Regularities of VQC compared to neural networks

Fully-connected networks are the building blocks of classical networks which are of the form,

$$f = f_n \circ f_{n-1} \circ \cdots \circ f_1 \tag{4}$$

where each layer $f_k(z) = \sigma_k(W_k\, z + b_k) : \mathbb{R}^{\ell_{k-1}} \to \mathbb{R}^{\ell_k}$ is composed of an activation function σ_k, a bias vector $b_k \in \mathbb{R}^{\ell_k}$ and a *weight* matrix $W_k \in \mathcal{L}\left(\mathbb{R}^{\ell_{k-1}}, \mathbb{R}^{\ell_k}\right)$. Here $\mathcal{L}(A, B)$ denotes the collection of linear maps between linear spaces A and B. Typically, there is no restriction on (classical network) weights W_k to be trained such that very often W_k is not *invertible* even if $\dim A = \dim B$. This results in a problem where a network prediction $y \in \mathbb{R}^m$ is hard to be traced back as $W_k^{-1}(y)$ does not exist nor $f_k^{-1}(y)$ is well-defined due to the structure of activation functions. Owing to this reason, classical networks are called black boxes and thus lack certain regularity even though the prediction ability is powerful.

On the contrary, since VQCs are comprised of unitary matrices, all gates U are invertible, and a quantum state $|\psi\rangle := U|\psi_0\rangle$ is easily revertible by $|\psi_0\rangle = U^*|\psi\rangle$. From this point of view, VQC possesses certain transparency and better regularity.

Additionally, in view of matrix groups, variational gates $U(\theta)$ in Eq. (1) form a Lie subgroup of $GL(\mathcal{H})$, all invertible matrices in $\mathcal{L}(\mathcal{H})$, so that $\{U(\theta)\}$ naturally inherited smooth structures from the differentiable submanifold [38]. This provides us some hints that the VQC training may be more stable as the training iterations $\theta^{(\text{iter})} \mapsto U(\theta^{(\text{iter})})$ being contained on the smooth submanifold $\mathcal{U}(\mathcal{H})$. Motivated by these properties, it leads us to consider viewing the explainability of VQC via a classical technique Grad-CAM.

C. Quantum Grad-CAM by VQC

Grad-CAM is a technique to interpret and visualize the decisions of CNNs via the gradient calculation of a target classifier. Let a set of CNN filters be $\{\mathcal{W}_{s_1,s_2,c,k}\}_{s_1=1,s_2=1,c=1,k=1}^{S,S,C,K}$ where C is the number of input channels and K is the number of output channels, and S is the kernel size. A *feature map* $\{A_{ij}^k\}_{i,j,k=1}^{W,H,K}$ is the convolution (output) of the CNN kernels with an input image $x = \{x_{i,j,c}\}_{c=1}^{C}$ of C channels given by,

$$A_{ij}^k := \sum_{s_1,s_2,c}^{S,S,C} \mathcal{W}_{s_1,s_2,c,k} \cdot x_{i+s_1-1,j+s_2-1,c} + b_k \tag{5}$$

where $W, H \in \mathbb{N}$ are the output image size, $i = 1, \ldots, W$ and $j = 1, \ldots, H$ are the output pixel location on the k^{th} channel and b_k is the associated bias.

It has been recognized that each filter serves a specific purpose to detect certain features, such as edges, textures, or patterns. The convolutional output Eq. (5) is also called the *activation map* to emphasize that certain input regions are highlighted and *activated* by the filters.

Our proposed method, QGrad-CAM is designed to probe the importance of the activation maps $\{A_{ij}^k\}$ via a VQC classifer with respect to the K filtered channels. Specifically, if a classifier $f : \mathbb{R}^{WHK} \to \mathbb{R}^m$ of m classes is welded after the CNN output $\{A_{ij}^k\}$ (see Fig. 3) such that

$$f(A_{ij}^k) = \left(f^1(A_{ij}^k), \ldots, f^m(A_{ij}^k)\right) \in \mathbb{R}^m \tag{6}$$

where each $f^\ell(A_{ij}^k)$ is the prediction for class $\ell = 1, \ldots, m$, then a *weighting function* $w_k^\ell : \mathbb{R}^{WHK} \to \mathbb{R}$ associated to classifier f can be defined,

$$w_k^\ell(A_{ij}^k) := \frac{1}{WH} \sum_{i,j}^{W,H} \frac{\partial f^\ell(A_{ij}^k)}{\partial A_{ij}^k}, \tag{7}$$

with the gradients of the class predictions computed and averaged out. Consequently, the Grad-CAM heatmap is obtained by the composition of the ReLU function with a weighted sum of feature maps Eq. (5), (7),

$$(\text{Grad-CAM heatmap})_{ij} = \text{ReLU}\left(\sum_k^K A_{ij}^k \cdot w_k^\ell(A_{ij}^k)\right), \tag{8}$$

The construction of the weighting function Eq. (7) naturally inherits important information for the final classification. Thus, it serves as a fundamental indicator for the resulting output.

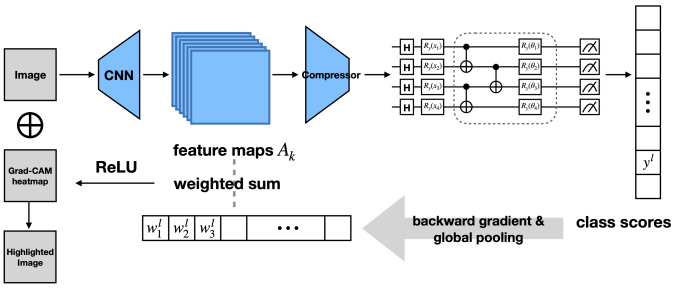


Fig. 3. The workflow of Quantum Grad-CAM.

IV. Explainability by Quantum Grad-CAM

It is our finding that the importance weighting w_k^ℓ Eq. (7) can be explicitly computed in certain cases of VQC, and the role of each image channel can be understood from the perspective of VQC. In contrast to the classical network, QGrad-CAM gives rise to a certain degree of explainability and learning transparency, which is the key investigation of this study.

Consider the quantum encoding depicted by Fig. 2,

$$V(x) = \bigotimes_{q=1}^{n} e^{-\frac{i}{2} x_q \sigma_{k_q}} \circ H_q \tag{9}$$

where $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ is an input of VQC, H_q is *any gate* on the q^{th} qubit not related to x, such as the Hadamard gate, and σ_{k_q} is a Pauli matrix of index $k_q \in \{0, 1, 2, 3\}$ in $\mathcal{P}$ depending on the q^{th} qubit. Then the measurement of an observable $Q \in \mathcal{L}(\mathcal{H})$ can be computed by a generalized form of Eq. (2) in terms of a density matrix $\rho \in \mathcal{L}(\mathcal{H})$,

$$\langle Q \rangle(x) = tr\left(Q\, U(\theta) V(x)\, \rho_0\, V^\dagger(x) U^\dagger(\theta)\right) \tag{10}$$

where $\rho_0 = |\psi_0\rangle \otimes \langle\psi_0|$, $U(\vartheta)$ is as Eq. (1) and tr is the trace operation on $\mathcal{L}(\mathcal{H})$. We calculate,

$$\begin{aligned} \frac{\partial \langle Q \rangle}{\partial x_q}(x) = tr\left(Q\, U(\theta) \frac{\partial V(x)}{\partial x_q}\, \rho_0\, V^\dagger(x) U^\dagger(\theta)\right) \\ + tr\left(Q\, U(\theta) V(x)\, \rho_0\, \frac{\partial V^\dagger(x)}{\partial x_q} U^\dagger(\theta)\right) \end{aligned} \tag{11}$$

Since the differential of a tensor product $x \mapsto A(x) \otimes B(x)$ is defined as,

$$\frac{\partial}{\partial x_q}\left(A(x) \otimes B(x)\right) = \left(\frac{\partial A(x)}{\partial x_q}\right) \otimes B(x) + A(x) \otimes \left(\frac{\partial B(x)}{\partial x_q}\right) \tag{12}$$

we have,

$$\frac{\partial}{\partial x_q} V(x) = -\frac{i}{2} V_{k_1}(x_1) \otimes \cdots \left(\sigma_{k_q} V_{k_q}(x_q)\right) \cdots \otimes V_{k_n}(x_n) \tag{13}$$

where we denote $V_{k_q}(x_q) := e^{-\frac{i}{2} x_q \sigma_{k_q}} \circ H_q$ for simplicity. We expand a density matrix by the tensorial basis $\{\sigma_{i_1} \otimes \cdots \otimes \sigma_{i_n}\}$,

$$\rho_0 = \sum_{i_1, \ldots, i_n} C_{i_1 \cdots i_n}\, \sigma_{i_1} \otimes \cdots \otimes \sigma_{i_n} \tag{14}$$

for some coefficients $C_{i_1 \cdots i_n} \in \mathbb{C}$. Then Eq. (11) yields,

$$\begin{aligned} \frac{\partial \langle Q \rangle}{\partial x_q}(x) = -\frac{i}{2} \sum_{i_1, \ldots, i_n} C_{i_1 \cdots i_n}\, tr\Big\{ U^\dagger(\theta) Q U(\theta) \\ \left(V_1(x_1) \sigma_{i_1} V_1^\dagger(x_1)\right) \otimes \cdots \otimes \left[\sigma_{k_q}, V_{k_q}(x_q) \sigma_{i_q} V_{k_q}^\dagger(x_q)\right] \otimes \cdots \\ \otimes \left(V_n(x_n) \sigma_{i_n} V_n^\dagger(x_n)\right) \Big\} \end{aligned} \tag{15}$$

where the middle term contains a Lie bracket $[A, B] := AB - BA$ at the q^{th} qubit. In fact, it can be explicitly computed,

$$\begin{aligned} \left[\sigma_{k_q}, V_{k_q}(x_q) \sigma_{i_q} V_{k_q}^\dagger(x_q)\right] = i \left(V_{k_q}\left(x_q + \frac{\pi}{2}\right) \sigma_{i_q} V_{k_q}^\dagger\left(x_q + \frac{\pi}{2}\right)\right. \\ \left. - V_{k_q}\left(x_q - \frac{\pi}{2}\right) \sigma_{i_q} V_{k_q}^\dagger\left(x_q - \frac{\pi}{2}\right)\right) \end{aligned} \tag{16}$$

Together, Eq. (15) and (16) give us the explicit formula of the importance weighting Eq. (7) in the VQC case. This then reveals how a VQC views the image channels and makes important selections in the sense of Grad-CAM.

V. Experiment

QGrad-CAM is applied to three datasets: MNIST, Dogs vs. Cats, and the TIMIT corpus [39], each with its respective classification task. Training is conducted end-to-end on all parameters of a 3-layer CNN and a 4-block VQC, both initialized from scratch. The gradient is computed at the end of the VQC towards the last layer of the CNN to generate the results.

Image classifications. The MNIST dataset consists of grayscale images sized 28×28, labeled across 10 classes. The Dogs vs. Cats dataset contains color images, resized to 128×128 for binary classification of dogs and cats.

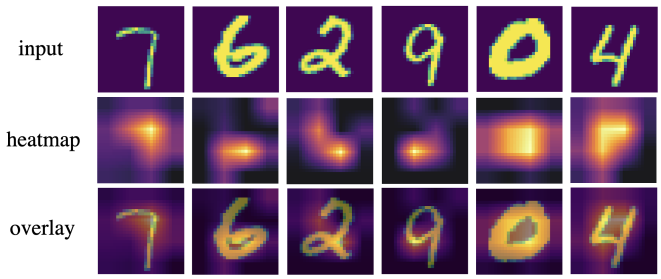


Fig. 4. **MNIST.** The generated CAM heatmaps highlight the regions with unique and distinguishable shapes and contours for each digit. For example, the heatmap for the digit '7' emphasizes the sharp turn, while the heatmap for '6' focuses on the closed loop.

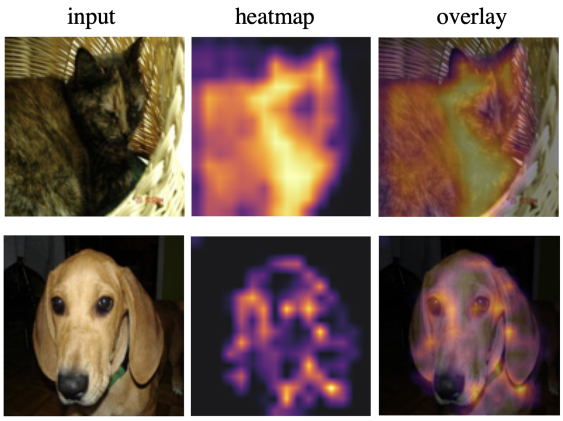


Fig. 5. **Dogs vs. Cats.** The higher resolution and colored images allow the detection of fine-grained textures and contours, which are crucial for identifying unique features to each animal.

Example results of QGrad-CAM with MNIST and Dogs vs. Cats are demonstrated in Fig 4 and Fig 5. It is observed that the generated QGrad-CAMs effectively highlight the regions where key features of each class are located. For MNIST, the heatmaps generally focus on areas where lines turn and curves form, emphasizing angles and the smoothness or sharpness of lines as discriminative features. For the Dogs vs. Cats dataset, the VQC classifier seems to learn critical textures and contours unique to cats and dogs. The heatmaps successfully identify the discriminative regions used for categorization.

Speech classifications. TIMIT contains recordings of American English speakers. The samples are in the 16 kHz WAV format and converted by Short-Time Fourier Transform (STFT) into spectrograms as inputs. Random samples are selected and corrupted by helicopter noise into background to create two classes: with or without a noisy background.

A spectrogram provides a visual representation of the frequency content of a speech signal over time. The horizontal axis represents time, the vertical axis represents frequency, and the color intensity indicates the amplitude at each frequency-time point. Human speech typically occupies specific frequency ranges in the spectrogram corresponding to the characteristics of the human vocal tract. In contrast, background noise often appears as diffuse, spread-out patterns across various frequencies and times without distinct features.

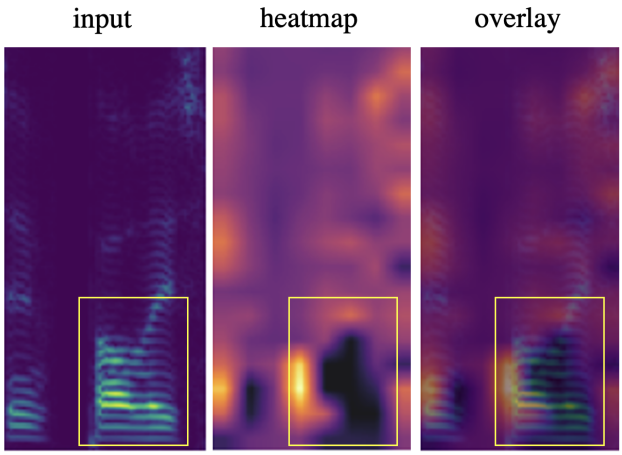


Fig. 6. **TIMIT.** Example of the spectrogram of a clean speech utterance. The heatmap has lower intensity inside the area enclosed by the yellow rectangle, where human speech signals are located.

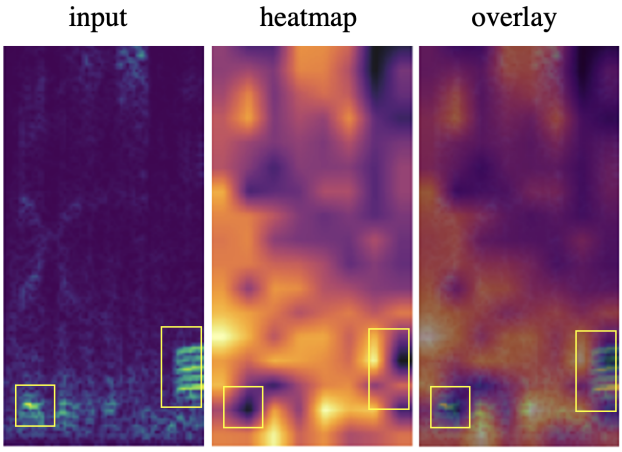


Fig. 7. **TIMIT.** Example of a spectrogram for a speech utterance corrupted with helicopter noise. The helicopter noise appears as diffuse areas in the spectrogram. The accompanying heatmap visually indicates that the network focuses on the background, outside the region of the human speech signal.

Figures 6 and 7 show spectrograms of clean and noisy speech samples, respectively. Yellow rectangles highlight the regions containing human speech signals for better clarity.

Our results indicate that the network focuses on areas outside the speech utterance to determine whether an utterance is corrupted by noise. This is observed in Figures 6 and 7, where the heatmaps show lower intensity within the yellow rectangles. Rather than concentrating on the portions of the spectrogram containing the speech signal, the network examines the background areas to detect signs of noise, allowing for more accurate identification of corruption.

VI. Conclusion

Motivated by the structured nature of the quantum framework, this work introduces QGrad-CAM, a novel method for providing visual explanations for model decisions. Our approach integrates the VQC with CNN gradient techniques to generate detailed, class-specific image localization. Experimental results on both image and speech datasets demonstrate

the method's effectiveness in highlighting discriminative features. Furthermore, an explicit importance weighting function associated to a VQC classifier can be analytically derived. Our results suggest potential advantages of quantum techniques in enhancing interpretability, highlighting the need for further exploration into the quantum advantage in this area.

REFERENCES

[1] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke *et al.*, "Noisy intermediate-scale quantum algorithms," *Reviews of Modern Physics*, vol. 94, no. 1, p. 015004, 2022.

[2] A. Abbas, D. Sutter, C. Zoufal, A. Lucchi, A. Figalli, and S. Woerner, "The power of quantum neural networks," *Nature Computational Science*, vol. 1, no. 6, pp. 403–409, 2021.

[3] Y. Du, M.-H. Hsieh, T. Liu, and D. Tao, "Expressive power of parametrized quantum circuits," *Physical Review Research*, vol. 2, no. 3, p. 033125, 2020.

[4] M. C. Caro, H.-Y. Huang, M. Cerezo, K. Sharma, A. Sornborger, L. Cincio, and P. J. Coles, "Generalization in quantum machine learning from few training data," *Nature communications*, vol. 13, no. 1, pp. 1–11, 2022.

[5] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, "Quantum circuit learning," *Physical Review A*, vol. 98, no. 3, p. 032309, 2018.

[6] S. Y.-C. Chen, T.-C. Wei, C. Zhang, H. Yu, and S. Yoo, "Quantum convolutional neural networks for high energy physics data analysis," *Physical Review Research*, vol. 4, no. 1, p. 013231, 2022.

[7] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," *Nature Physics*, vol. 15, no. 12, pp. 1273–1278, 2019.

[8] S. Y.-C. Chen, S. Yoo, and Y.-L. L. Fang, "Quantum long short-term memory," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 8622–8626.

[9] R. Di Sipio, J.-H. Huang, S. Y.-C. Chen, S. Mangini, and M. Worring, "The dawn of quantum natural language processing," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 8612–8616.

[10] J. Stein, I. Christ, N. Kraus, M. B. Mansky, R. Müller, and C. Linnhoff-Popien, "Applying qnlp to sentiment analysis in finance," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 2. IEEE, 2023, pp. 20–25.

[11] S. S. Li, X. Zhang, S. Zhou, H. Shu, R. Liang, H. Liu, and L. P. Garcia, "Pqlm-multilingual decentralized portable quantum language model," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[12] S. Y.-C. Chen, C.-H. H. Yang, J. Qi, P.-Y. Chen, X. Ma, and H.-S. Goan, "Variational quantum circuits for deep reinforcement learning," *IEEE access*, vol. 8, pp. 141 007–141 024, 2020.

[13] O. Lockwood and M. Si, "Reinforcement learning with quantum variational circuit," in *Proceedings of the AAAI conference on artificial intelligence and interactive digital entertainment*, vol. 16, no. 1, 2020, pp. 245–251.

[14] W. J. Yun, J. Park, and J. Kim, "Quantum multi-agent meta reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 9, 2023, pp. 11 087–11 095.

[15] A. Fisher, C. Rudin, and F. Dominici, "All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously," *Journal of Machine Learning Research*, vol. 20, no. 177, pp. 1–81, 2019.

[16] H. Nori, R. Caruana, Z. Bu, J. H. Shen, and J. Kulkarni, "Accuracy, interpretability, and differential privacy via explainable boosting," in *International conference on machine learning*. PMLR, 2021, pp. 8227–8237.

[17] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[18] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[19] M. T. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-precision model-agnostic explanations," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[20] J. H. Friedman, "1999 reitz lecture," *Ann. Stat.*, vol. 29, no. 5, pp. 1189–1232, 2001.

[21] A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin, "Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation," *journal of Computational and Graphical Statistics*, vol. 24, no. 1, pp. 44–65, 2015.

[22] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[23] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[24] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks," in *2018 IEEE winter conference on applications of computer vision (WACV)*. IEEE, 2018, pp. 839–847.

[25] A. Mari, T. R. Bromley, J. Izaac, M. Schuld, and N. Killoran, "Transfer learning in hybrid classical-quantum neural networks," *Quantum*, vol. 4, p. 340, 2020.

[26] S. Y.-C. Chen, C.-M. Huang, C.-W. Hsing, and Y.-J. Kao, "An end-to-end trainable hybrid classical-quantum classifier," *Machine Learning: Science and Technology*, vol. 2, no. 4, p. 045021, 2021.

[27] L. Power and K. Guha, "Feature importance and explainability in quantum machine learning," *arXiv preprint arXiv:2405.08917*, 2024.

[28] R. Heese, T. Gerlach, S. Mücke, S. Müller, M. Jakobs, and N. Piatkowski, "Explaining quantum circuits with shapley values: Towards explainable quantum machine learning," *arXiv preprint arXiv:2301.09138*, 2023.

[29] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*. Springer, 2014, pp. 818–833.

[30] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Object detectors emerge in deep scene cnns," *arXiv preprint arXiv:1412.6856*, 2014.

[31] A. Dosovitskiy, T. Brox *et al.*, "Inverting convolutional networks with convolutional networks," *arXiv preprint arXiv:1506.02753*, vol. 4, no. 2, p. 3, 2015.

[32] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5188–5196.

[33] ——, "Visualizing deep convolutional neural networks using natural pre-images," *International Journal of Computer Vision*, vol. 120, pp. 233–255, 2016.

[34] P. O. Pinheiro and R. Collobert, "From image-level to pixel-level labeling with convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1713–1721.

[35] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Is object localization for free?-weakly-supervised learning with convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 685–694.

[36] S. Y.-C. Chen, C.-M. Huang, C.-W. Hsing, H.-S. Goan, and Y.-J. Kao, "Variational quantum reinforcement learning via evolutionary optimization," *Machine Learning: Science and Technology*, vol. 3, no. 1, p. 015025, 2022.

[37] M. C. Caro, H.-Y. Huang, N. Ezzell, J. Gibbs, A. T. Sornborger, L. Cincio, P. J. Coles, and Z. Holmes, "Out-of-distribution generalization for learning quantum dynamics," *Nature Communications*, vol. 14, no. 1, p. 3751, 2023.

[38] J. M. Lee, *Smooth Manifolds*. New York, NY: Springer New York, 2012, pp. 1–31. [Online]. Available: https://doi.org/10.1007/978-1-4419-9982-5_1

[39] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Getting started with the darpa timit cd-rom: An acoustic phonetic continuous speech database," *National Institute of Standards and Technology (NIST), Gaithersburgh, MD*, vol. 107, p. 16, 1988.

QuXAI: Explainers for Hybrid Quantum Machine Learning Models

Saikat Barua[1], Mostafizur Rahman[2], Shehenaz Khaled[3], Md Jafor Sadek[4], Rafiul Islam[5], and Dr. Shahnewaz Siddique[6]

[1]North South University, Dhaka, saikat.barua@northsouth.edu
[2]North South University, Dhaka, mostafizur.rahman10@northsouth.edu
[3]North South University, Dhaka, shehenaz.khaled@northsouth.edu
[4]North South University, Dhaka, jafor.sadek@northsouth.edu
[5]North South University, Dhaka, rafiul.islam19@northsouth.edu
[6]Associate Professor, North South University, Dhaka, shahnewaz.siddique@northsouth.edu

Abstract

The emergence of hybrid quantum-classical machine learning (HQML) models opens new horizons of computational intelligence but their fundamental complexity frequently leads to black box behavior that undermines transparency and reliability in their application. Although XAI for quantum systems still in its infancy, a major research gap is evident in robust global and local explainability approaches that are designed for HQML architectures that employ quantized feature encoding followed by classical learning. The gap is the focus of this work, which introduces QuXAI, an framework based upon Q-MEDLEY, an explainer for explaining feature importance in these hybrid systems. Our model entails the creation of HQML models incorporating quantum feature maps, the use of Q-MEDLEY, which combines feature based inferences, preserving the quantum transformation stage and visualizing the resulting attributions. Our result shows that Q-MEDLEY delineates influential classical aspects in HQML models, as well as separates their noise, and competes well against established XAI techniques in classical validation settings. Ablation studies more significantly expose the virtues of the composite structure used in Q-MEDLEY. The implications of this work are critically important, as it provides a route to improve the interpretability and reliability of HQML models, thus promoting greater confidence and being able to engage in safer and more responsible use of quantum-enhanced AI technology.

Our code and experiments are open-sourced at: https://github.com/GitsSaikat/QuXAI

Keywords Quantum Machine Learning · Explainable AI · Hybrid Quantum-Classical Algorithms · Interpretable Quantum Systems · Quantum Feature Engineering · Trustworthy Quantum AI

1 Introduction

The emergence of Quantum Machine Learning (QML), as a field promises novel computational paradigms and enhanced learning capabilities [6, 11]. Near-term applications often utilize hybrid quantum-classical (HQML) algorithms, in which quantum processors manage to solve such tasks as data representation or kernel evaluation, with classical computers focusing on optimization and control [37, 32]. While this pragmatic approach accelerates the exploration of quantum-enhanced learning, it often inherits the "black box" nature common to complex classical models, obscuring the internal decision-making processes. This aspect of QML systems lack of transparency presents a great barrier to wider acceptance and trust in the QML systems. Thus, building strong eXplainable AI (XAI) techniques customized for these hybrid architectures must be considered. Such explainability is not only important to debug and validate models but also to establish user trust, regulatory compliance and finally open up the potential of QML for scientific discovery and real-world problem-solving by ensuring that their operational logic is scrutable.

The pressing need for interpretability in QML systems has led to research on transferring classical XAI principles and generation of new quantum-specific explanation mechanisms. Approaches like Shapley values have been extended to measure the influence of components in quantum circuits [1, 14], and there are quantum iterations of LIME (Q-LIME) that intend to give local instance-based explanations for quantum neural networks [9]. Other explorations include learning capability of parameterized quantum circuits [29] and diagnostics such as quantum neural tangent kernels [5]. However, while a significant fair amount of great, overarching explainability approaches exist, there is a conspicuous lack of these approaches developed specifically for the common HQML paradigm in which classical input features are transformed via a quantum feature map into a new form (e.g. amplitudes of the state vector or the kernel matrices) on which a classical learner then operates Existing tools tend to either treat the whole HQML system as a black box or aim at explaining quantum circuit in isolation of the hybrid data flow without describing the impact of initial classical attributes through the hybrid data flow. Quantum Computing is error prone, quantum error correction causes additional overhead [24]; therefore, as QML models, including those which could be used for tasks such as enhancing the error correction, become more complicated, the necessity for the tools to understand such behavior becomes of even greater importance to make such models trusted and operational. In order to handle this, we present QuXAI, a unified scheme with a Q-MEDLEY explainer, which gives global feature importance scores by carefully examining perturbations at the classical input level and tracing their effects through quantum encoding and subsequent classical learning process.

Our methodology within the QuXAI framework encompasses three core components: (i) the construction of HQML models, wherein classical input data is encoded into quantum states using defined quantum feature maps (e.g., RX rotations for amplitude encoding, or quantum kernel functions), and the resulting quantum-derived representations are then processed by standard classical machine learning algorithms; (ii) the Q-MEDLEY explainer, which synthesizes insights from Drop-Column Importance (DCI) and Permutation Importance (PI) by perturbing original classical features and, critically, re-evaluating the quantum feature mapping stage for each perturbation before assessing the impact on the classical learner's performance, thereby respecting the hybrid data flow; and (iii) a visualization module that renders these feature importance scores as accessible bar charts. This approach is novel in its specific adaptation to the operational pipeline of feature-encoding HQML models. Unlike generic model-agnostic explainers that do not differentiate the quantum processing step, or pure quantum XAI methods that might focus on circuit parameters, Q-MEDLEY directly attributes importance to the initial classical features based on their influence throughout the entire hybrid model. This tailored design, drawing inspiration from robust classical explainability techniques [26], allows for a more faithful and nuanced interpretation of feature contributions in such systems.

Our empirical studies show the practicality of the QuXAI framework. We first show that the wide variation of amplitude-encoded HQML models devised are competitive in their predictive performance to their classical counterparts, even under the condition of additional noisy and redundant attributes appending the datasets, thus validating them as legitimate objects for explanation. Whilst applied to these HQML models, the Q-MEDLEY explainer systematically produces a coherent feature importance hierarchy that is capable of discriminating between original, meaningful features from synthetic noise or redundancies in different datasets, as demonstrated for the Noisy Iris (Figure 3) and Noisy Wine (Figure 4) datasets. Moreover, rigorous validation of Q-MEDLEY in the classical ML settings, with interpretable models as ground truth, demonstrates its good performance – we achieve high Recall@3 scores (Figure 5), and Spearman rank correlations (Figure 6) which are often comparable, Ablation studies (Table 1) also reinforce that the composite architecture employed by Q-MEDLEY, which is a combination of an adaptive weighting approach and interaction-aware mechanisms, leads to an improvement in its robustness and accuracy of feature attribution.

The main contributions of this paper are as follows:

- The development of Q-MEDLEY, a novel global and local feature importance explainer specifically designed for hybrid quantum-classical machine learning models that utilize quantum feature encoding.
- The introduction of the QuXAI framework, an integrated environment for training, evaluating, and explaining HQML models, complete with data processing and visualization capabilities along with a detailed mathematical foundation.
- An in-depth analysis of Q-MEDLEY's core components, including adaptive weighting and interaction-aware permutation importance. This study highlights how each component contributes to the overall effectiveness of the explainer and provides general insights into the design principles of interpretable machine learning systems.

The rest of the article is organized as follows: Section 2 reviews related works. Section 3 describes the methodology used in this study. Section 4 presents the results obtained. Section 5 discusses how the results achieved our research objectives. Finally, Section 6 concludes with remarks on future work.

2 Related Works

Numerous near-term QML applications use hybrid quantum-classical algorithms, wherein quantum processors are tasked with specific, frequently data representation or kernel evaluation oriented, tasks, which are executed by classical computers responsible for optimization and control [37, 32]. This pragmatic approach, however, receives the plight of the "black box" common for classical deep learning. And as quantum systems, built upon variational quantum circuits (VQCs) for supervised learning [6], etc., that are increasingly faced with more complex tasks, the necessity for translucence becomes more acute. Nonetheless, a straightforward transcription of eXplainable AI (XAI) methods from classical machine learning to a quantum context poses several non-trivial challenges, mainly as a result of the intrinsic probabilistic mouse on quantum measurements and the exponential scaling of quantum state-space dimensions, presenting insurmountable stumbling blocks to the contemporary implementations of XAI [7]. In order to add our study to the broader context of quantum machine learning and explainability, a map of central literature is given in Figure 1 specifying key connections and themes.

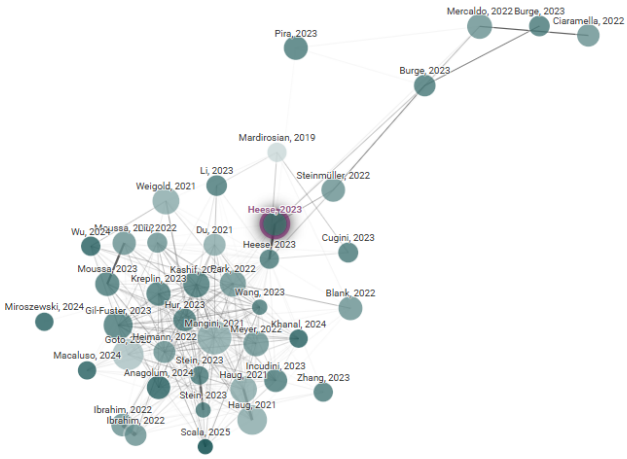


Figure 1: Citation network map illustrating the relationships and thematic clustering of key research papers relevant to the QuXAI framework, hybrid quantum-classical models, and explainable AI in quantum machine learning. Node size represent citation counts or other metrics, and edges indicate citation links or strong thematic overlap.

2.1 Shedding Light on the Quantum Black Box

One of the important directions on explainability of Quantum Models is the use of Shapley values, a notion borrowed from cooperative game theory – to associate importance scores with various parts of a quantum computation. Scientists have shown that Shapley values are instrumental in decoupling parameterized quantum circuits, measuring the effect of single gates or groups of gates on the circuit's behavior for particular tasks, such as classification/generative modeling [1, 17]. This helps to explain why a specific circuit configuration works and, also, provides insight into what informs robust PQC design. Despite the problem being intractable, however, the computational load of computing Shapley values is still a concern. To solve this problem, development of specialized quantum algorithms is under way that can calculate these values with polynomial efficiency, which makes such explanations more practicable for complex QML models [25, 14]. Beyond these global, model-centric explanations, methods of local, instance-specific interpretability are also comes in handy. For instance, Q-LIME, a quantum version of the classical LIME scheme, has been suggested for explaining the predictions of quantum neural networks for single data points, having provided a means of delimiting areas where predictions may be governed by the natural stochasticity of quantum measurements rather than learned aspects [9]. The applicability of such XQML techniques is investigated in various practical applications, such as improving the reliability of QML systems for mobile malware detection through revealing the characteristics determining which classification decision-taking is made [30].

2.2 Innovations on Architectures and Feature Representation

The efficiency and explainability of QML systems are inherently linked with the quantum model architectures that underlies the techniques of data representation. Variational Quantum Circuits (VQCs) provide a flexible backbone

for many QML algorithms including applications from supervised learning [6] to quantum reinforcement learning, where improved optimized action decoding within policy gradient frameworks achieve performance advantages on both simulators and early quantum hardware [3]. The design of the PQC ansatz is also an active area of research, where AI inspired approaches such as reducing width QNNs are developed to alleviate issues such as barren plateaus and increased trainability, [16, 27]. Architecture aspects also reach distributed quantum computing where new entanglement connection methods between multi-layer hardware-efficient ansätze can be searched for tasks such as distributed Variational Quantum Eigensolvers [31]. The systematic comparison of various architectures of PQC regarding metric of learning capability, is vital to develop design guidelines, and understand their expressivity [29].

Classical-quantum interface is a fundamental assumption of the hybrid QML that is implemented through the quantum feature engineering. Neural quantum encoding (NQE) uses deep learning to optimize the representation of classical data in quantum states for high classification accuracy [12], which is also theoretically proven for universal approximation in quantum feature maps [39]. Quantum kernel methods develop with compact circuits for binary classifiers [19] and scalable variational algorithms such as VQASVM [28] showed satisfactory results. Quantum Hartley Transform for complex distributions, among other primitives, is beneficial in generative modeling [34]. Analytic theories of wide QNN dynamics [38] are used to achieve richer understanding that is applied to QML in high-energy physics [41] and to cybersecurity [43].

2.3 Addressing Trainability, Noise, and Generalization

Major challenges are inherent in the path to utilize the complete capacity of QML, especially with regard to model trainability, ability to resist noise, and ability to generalize from a small sample set. Finite-sampling noise, a given concern in estimating expectation values from quantum circuits, can derail training and degradate performance. Approaches like variance regularization are being designed to suppress this noise through direct inclusion in the training system partly without the need for extra quantum circuit verifying ideas, hence increasing the convergence rate and producing better output quality [2]. The performance of QML models is also highly sensitive to hyperparameters choices. Systematic studies that use functional ANOVA for example are absolutely necessary for determining the most important hyperparameters (e.g., the learning rate, ansatz structure), and for designing data-driven prior beliefs to direct more effective hyperparameter optimization searches over a variety of data sets [4, 40].

It is imperative to make Quantum Neural Network (QNN) predictions a priori so that tools such as Quantum Neural Tangent Kernel (QNTK) provide diagnostic abilities of model design and resource allocation [5]. Generalization in Noisy Intermediate-Scale Quantum (NISQ) QML is a major concern of interest with current surveys describing error bounds [10]. The classical generalization theories are challenged because of the possibility of the QNNs to fit random data, and it is necessary to have new perspectives [36]. To improve trainability and generalization, approaches like reduced-domain initialization are explored [35]. The choice between cost function design (global vs. local) has a huge effect on training dynamics[23], while efficient shot estimation is critical for quantum kernel methods [22].

2.4 Scaling, Optimization, and Resource Efficiency

Turning postulated QML promises into advantage, the quantum community requires collective efforts in algorithm scaling, optimization and resource efficiency as well. Frameworks such as Élivágar of the Quantum Circuit Search (QCS) type give a promising way for automating the design of high quality and noise-resistant PQCs by intelligently traversing complicated architectural spaces, taking hardware constraints into account [8]. To address bigger problems, quantum processor can solve, distributed quantum computing paradigms can be further explored. Examples of QUDIO are schemes that attempt to speed up variational quantum algorithms by distributing learning problems across several local quantum processors coordinated by a classical server[13]. Another limit is managing big datasets which is a standard need in real-world machine learning. Examples of innovations include using randomized measurements to estimate quantum kernels, so that the quantum computational time, scales linearly, in contrast to quadratically, and thus allow for QML with larger inputs, [18, 21].

Limitations on available resources of current and near-term quantum devices are another driver of exploration of model compression and efficiency methods. Adaptation of ensemble learning methods – such as bagging and AdaBoost, which are believed to been employed within classical ML – are being explored for QNNs. Not only do these provide a mechanism to build more powerful/robust models from many, possibly weaker/smaller, quantum models, but they can also ameliorate the effects of hardware noise and also happen to save a certain amount of quantum resources [20]. Finally, bridging the gap between abstract algorithms of quantum computing and concrete hardware implementation, we need optimization at the lowest possible levels. Quantum devices can be customized through the pulse-level control of quantum gates, and entanglers in PQCs. This pulse-level optimization has been demonstrated to drastically cut down

state preparation times, preserve or improve circuit expressibility and facilitate VQAs trainability on real quantum hardware in a much more efficient and effective manner, opening doors to more efficient QML use [15, 33].

The joint breakthroughs on these interrelated fields, from the underlying XAI ideas for the quantum systems to revolutionary model architectures, strong training schemes and hardware aware optimizations, are moving the frontier of quantum machine learning towards effective practice of the real world applications.

3 Methodology

Our work introduces QuXAI, a comprehensive framework intended to assist with both the process of training and explanation of hybrid quantum-classical machine learning (HQML) models. There are three major components of the framework: the model building of HQML models using quantum feature maps, a custom Q-MEDLEY explainer tailored for hybrid architectures and a visualization module to make sense of the outputs of the explanations. In our approach, the HQML models use quantum circuits to carry data that can either be the encoding of classical features to quantum amplitudes or construction of quantum kernels which are further managed by classical algorithms. Q-MEDLEY, an explainer that we created, combines insights from Drop-Column and Permutation Importance paradigms in order to provide strong global feature importance metrics, accounting for quantum encoding effects very carefully. Finally, the framework incorporates a way of visualizing the importance scores into a simple set of bar charts that facilitates easy evaluation of feature significance. The algorithm 2 guides the system through data transformation, HQML training, performance evaluation, Q-MEDLEY integration for feature interpretation and visualization of results, each step of the way ensuring that this hybrid quantum machine learning workflow is robust and interpretable. The workflow of the QuXAI framework is illustrated in 2.

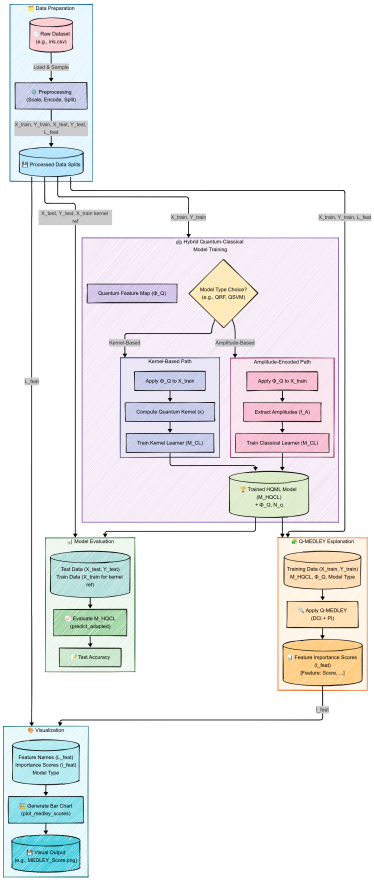


Figure 2: Overall workflow of the QuXAI framework, illustrating the sequential stages from data preparation, through hybrid quantum-classical model (HQML) training and evaluation, to feature importance explanation using Q-MEDLEY and subsequent visualization of the results. The training stage depicts distinct paths for amplitude-encoded and kernel-based HQML models. The complete pipeline is detailed in Algorithm 2.

3.1 Q-MEDLEY

The Q-MEDLEY explainer introduces a robust methodology for quantifying global feature importance in hybrid quantum-classical machine learning (HQML) architectures. These HQML models, denoted as $M_{HQCL}(x) = M_{CL}(f_Q(\Phi_Q(x)))$, typically involve a two-stage process: an initial quantum feature map $\Phi_Q : \mathbb{R}^D \to \mathcal{H}$ that transforms classical D-dimensional input vectors $x \in X$ into quantum states $|\psi(x)\rangle \in \mathcal{H}$, followed by a classical learning algorithm M_{CL} that operates on a classical representation $f_Q(|\psi(x)\rangle)$ derived from these quantum states (e.g., state vector amplitudes or kernel evaluations). Q-MEDLEY's design, inspired by the MEDLEY explainer concept proposed for classical models in KAXAI [26], is predicated on the principle that a synthesis of multiple, distinct feature attribution techniques can yield more stable and comprehensive insights than any single method alone. It achieves this by integrating two established perturbation-based approaches: Drop-Column Importance (DCI) and Permutation Importance (PI). This combined approach shares the goal of providing feature-level insights, similar to efforts adapting Shapley values [1] or LIME [9] for quantum circuits, but distinguishes itself by its specific aggregation strategy and direct applicability to HQML feature vectors. The core rationale is that DCI assesses a feature's necessity by observing performance degradation upon its complete removal (approximated by neutralization to $x_j = 0$), while PI evaluates its predictive value by disrupting its relationship with the target variable Y through random shuffling. By systematically applying these perturbations to a reference dataset (X_{ref}, Y_{ref}) and meticulously evaluating the HQML model's response, Q-MEDLEY aims to provide a holistic measure of each feature's contribution to the model's overall predictive accuracy, $\mathcal{A}$.

The internal procedure of Q-MEDLEY is critically adapted to the hybrid nature of M_{HQCL}. Given a trained model M_{HQCL} and the reference data (X_{ref}, Y_{ref}), a baseline performance $\mathcal{A}_{base} = \mathcal{A}(M_{HQCL}(X_{ref}), Y_{ref})$ is established. To compute DCI for the j-th feature, X_{ref} is modified to $X_{ref}^{(j,0)}$ by replacing its j-th column with a neutral vector (e.g., $x_j \leftarrow 0$), and the importance is given by:

$$I_j^{DCI} = \mathcal{A}_{base} - \mathcal{A}(M_{HQCL}(X_{ref}^{(j,0)}), Y_{ref}). \tag{1}$$

For PI, the j-th column of X_{ref} is randomly permuted K times, creating datasets $X_{ref}^{(j,\pi_k)}$ where $k = 1, \ldots, K$, and the importance is:

$$I_j^{PI} = \mathcal{A}_{base} - \frac{1}{K}\sum_{k=1}^{K} \mathcal{A}(M_{HQCL}(X_{ref}^{(j,\pi_k)}), Y_{ref}). \tag{2}$$

The crucial step is the evaluation of M_{HQCL} on these perturbed datasets. If M_{HQCL} is amplitude-based, where $f_Q(|\psi(x)\rangle)$ are derived from the amplitudes (e.g., $f_Q(|\psi(x)\rangle) = [|\langle 0|\psi(x)\rangle|^2, \ldots, |\langle 2^N - 1|\psi(x)\rangle|^2]$ for an N-qubit system), then for any perturbed input x', the quantum feature map $\Phi_Q(x')$ must be re-evaluated to obtain the new amplitudes for M_{CL}. If M_{HQCL} is kernel-based, where M_{CL} utilizes a quantum kernel $K(x_i, x_l) = \kappa(\Phi_Q(x_i), \Phi_Q(x_l)) = |\langle \psi(x_i)|\psi(x_l)\rangle|^2$, then evaluating on a perturbed dataset X'_{ref} requires computing a new kernel matrix $K(X'_{ref}, X_{ref})$ by re-evaluating the kernel function κ between perturbed instances and the original reference training instances. The final importance for feature j is an aggregation:

$$I_j = \frac{1}{2}(I_j^{DCI} + I_j^{PI}). \tag{3}$$

The significance of Q-MEDLEY lies in its tailored approach to providing global feature explanations for this distinct class of HQML models. Unlike generic model-agnostic explainers, Q-MEDLEY's internal prediction mechanism explicitly accounts for the quantum feature encoding stage, $x \xrightarrow{\Phi_Q} |\psi(x)\rangle \xrightarrow{f_Q}$ classical features/kernel. This ensures that the impact of perturbations to the original classical features is correctly propagated through the quantum transformation Φ_Q before being assessed by the classical learner M_{CL}. This nuanced handling is essential for accurately attributing importance to features whose influence is mediated by quantum mechanical representations, a challenge central to making quantum AI systems interpretable [9, 7]. For instance, pre-calculating the reference kernel matrix $K(X_{ref}, X_{ref})$ during an initial setup phase optimizes subsequent kernel evaluations involving perturbed data against this fixed reference, a specific adaptation for quantum kernel machines. By providing a principled framework that combines the strengths of DCI and PI while respecting the unique data flow of HQML systems, Q-MEDLEY offers a significant step towards enhancing the transparency and trustworthiness of emergent quantum-enhanced predictive models, thereby facilitating deeper scientific understanding and more targeted model refinement, contributing to the broader effort of building explainable QML systems [17].

Algorithm 1 Q-MEDLEY Explainer

Input: HQML model M_{HQCL}, type T_M, map Φ_Q, data (X_{ref}, Y_{ref}), repeats K
Output: Feature importance scores I

function PREDICTADAPTED(X_{eval}, X_{ref_train}) ▷ Handles Φ_Q and M_{CL} based on T_M
  if T_M is amplitude-based **then return** M_{CL}(Amplitudes($\Phi_Q(X_{eval})$))
  else if T_M is kernel-based **then return** M_{CL}(QuantumKernel($\Phi_Q(X_{eval}), \Phi_Q(X_{ref\_train})$))
  end if
end function
$\mathcal{A}_{base} \leftarrow$ Accuracy(PredictAdapted(X_{ref}, X_{ref}), Y_{ref})
$D \leftarrow$ num_features(X_{ref})
for $j \leftarrow 1$ to D **do**
  $X_{drop} \leftarrow X_{ref}$ with j-th feature neutralized
  $I_j^{DCI} \leftarrow \mathcal{A}_{base} -$ Accuracy(PredictAdapted(X_{drop}, X_{ref}), Y_{ref})
  scores_perm $\leftarrow []$
  for $iter \leftarrow 1$ to K **do**
    $X_{perm} \leftarrow X_{ref}$ with j-th feature permuted
    Add Accuracy(PredictAdapted(X_{perm}, X_{ref}), Y_{ref}) to scores_perm
  end for
  $I_j^{PI} \leftarrow \mathcal{A}_{base} -$ Mean(scores_perm)
  $I_j \leftarrow (I_j^{DCI} + I_j^{PI})/2$
end for
return I

3.2 Hybrid Quantum Classical Models

The hybrid quantum-classical models (HQML) investigated herein offer a strategic framework for integrating quantum information processing into established machine learning paradigms, particularly targeting applications amenable to near-term quantum simulations. The design philosophy is to harness quantum mechanics for sophisticated data representation, transforming classical input data $x \in \mathbb{R}^D$ into quantum states $|\psi(x)\rangle \in \mathcal{H}$ residing in a 2^N-dimensional Hilbert space (where N is typically proportional to D). This quantum state preparation, defined by a quantum feature map $\Phi_Q : x \mapsto |\psi(x)\rangle$, constitutes the quantum subroutine. Subsequently, a classical representation $f_Q(|\psi(x)\rangle)$ is extracted from this quantum state, which then serves as input to a conventional classical machine learning algorithm M_{CL} for the final learning or inference task. Thus, the overall HQML model can be expressed as $M_{HQCL}(x) = M_{CL}(f_Q(\Phi_Q(x)))$. This hybrid architecture is motivated by the desire to explore potential quantum enhancements in feature space construction, while leveraging the robustness and scalability of classical learning algorithms for optimization and decision-making, thereby navigating the current limitations of quantum hardware.

The operationalization of these HQML models proceeds via two principal methodologies, differentiated by the nature of the classical representation f_Q derived from the quantum state $|\psi(x)\rangle$. The first, termed "amplitude-encoded HQML," utilizes the probability amplitudes of the quantum state. After applying the feature map $\Phi_Q(x)$, which might involve a sequence of parameterized quantum gates $U_j(x_j)$ such that $|\psi(x)\rangle = (\prod_j U_j(x_j))\,|0\rangle^{\otimes N}$, the classical representation is formed by the squared magnitudes of the amplitudes in the computational basis:

$$f_Q(|\psi(x)\rangle) = \left[|\langle 0|\psi(x)\rangle|^2, |\langle 1|\psi(x)\rangle|^2, \ldots, |\langle 2^N-1|\psi(x)\rangle|^2\right]^T \in \mathbb{R}^{2^N}. \tag{4}$$

This vector $x'_q = f_Q(|\psi(x)\rangle)$ then becomes the input for a classical algorithm M_{CL}, such as a Random Forest or Logistic Regression, which is trained on a dataset $(X'_{q,ref}, Y_{ref})$.

The second methodology encompasses "quantum kernel-based HQML." In this paradigm, the quantum feature map $\Phi_Q(x)$ is employed to define a quantum kernel function $\kappa(x_i, x_l)$, which measures the similarity between the quantum states corresponding to two classical data points x_i and x_l. A common choice for this is the fidelity kernel:

$$\kappa(x_i, x_l) = |\langle \psi(x_i)|\psi(x_l)\rangle|^2. \tag{5}$$

For a training set X_{ref}, an $N_{ref} \times N_{ref}$ Gram matrix K_{ref} is constructed, where each element $(K_{ref})_{il} = \kappa(X_{ref}[i], X_{ref}[l])$. This kernel matrix K_{ref} is then directly utilized by a classical kernel-based algorithm M_{CL}, such as a Support Vector Machine operating with a precomputed kernel, $M_{CL}(K_{ref})$. For distance-based classifiers like k-Nearest Neighbors, the kernel values can be transformed into a distance metric, for example:

$$d(x_i, x_l) = \sqrt{1 - \kappa(x_i, x_l)}. \tag{6}$$

A distinctive characteristic of this framework is the explicit association of the quantum feature map Φ_Q and the number of qubits N with the trained classical model M_{CL}. This is achieved by augmenting the classical model object with these quantum-specific attributes post-training. This procedural choice is critical for enabling subsequent explainability analyses, such as those performed by Q-MEDLEY. When an explainer perturbs an input feature x_j to x'_j, it must have access to Φ_Q to correctly propagate this change through the quantum encoding step, i.e., to compute the new quantum state $|\psi(x')\rangle$ or its contribution to a kernel evaluation. For instance, the evaluation of the model on a perturbed input x' for an amplitude-based model requires recomputing Equation 4 with $|\psi(x')\rangle$, while for a kernel model, it involves re-evaluating Equation 5 using $|\psi(x')\rangle$ against the reference states $|\psi(X_{ref}[l])\rangle$.

HQML models, M_{HQCL}, are important in their own right because they present a clear and easily available infrastructure to compare the representation of quantum data aided with actionable classical machine learning. By restricting quantum operations to feature mapping, these models facilitate the study of the impact that different quantum encodings may have on classical learners' results, without the need for complete quantum training. This modularity makes it easy to benchmark and compare, utilizing the classical machine learning infrastructure. Including specific information about the quantum feature map, in the model object is essential for the creation of concentrated interpretability devices such Q-MEDLEY. These tools are essential for constructive interpretation of the influence of quantum features or kernel similarities in model prediction, thus increasing understanding and contributing to the responsible quantum AI development. The classical learner can produce the following result that can be presented in its generalized form as:

$$\hat{Y} = M_{CL}(f_Q(\Phi_Q(X))), \tag{7}$$

where $\hat{Y}$ are the predictions for a set of inputs X.

Algorithm 2 QuXAI Framework Pipeline

Input: Raw data D_{raw}, target T_{col}, model type M_{type}, Q-MEDLEY repeats K_P
Output: Trained M_{HQCL}, Importances I_{feat}, Visualization V_{scores}
Define: Φ_Q: Quantum Feature Map; f_A: Amplitude Extraction; κ: Quantum Kernel Func.

function PREPAREDATA(D_{raw}, T_{col})
  $X, Y \leftarrow$ Features/Target(D_{raw}); Scale X; Encode Y
  return TrainTestSplit(X, Y) giving $X_{tr}, X_{te}, Y_{tr}, Y_{te}, L_{feat}$
end function
function TRAINHQML($X_{tr}, Y_{tr}, M_{type}, \Phi_Q$)
  if M_{type} is amplitude-based **then** $X_{tr\_repr} \leftarrow f_A(\Phi_Q(X_{tr}))$
  else if M_{type} is kernel-based **then** $X_{tr\_repr} \leftarrow \kappa(\Phi_Q(X_{tr}), \Phi_Q(X_{tr}))$ ▷ Forms kernel matrix
  end if
  $M_{CL} \leftarrow$ InitClassicalLearner(M_{type}); M_{CL}.fit(X_{tr_repr}, Y_{tr})
  Attach Φ_Q to M_{CL} (now M_{HQCL}); **return** M_{HQCL}
end function

▷ Main Workflow
$X_{tr}, X_{te}, Y_{tr}, Y_{te}, L_{feat} \leftarrow$ PrepareData(D_{raw}, T_{col})
$M_{HQCL} \leftarrow$ TrainHQML($X_{tr}, Y_{tr}, M_{type}, \Phi_Q$)
$\text{Acc}_{te} \leftarrow$ Accuracy(M_{HQCL}.predict_adapted(X_{te}, X_{tr}), Y_{te}) ▷ predict_adapted uses Φ_Q
Print "Accuracy for " M_{type} ": " Acc_{te}
$I_{feat} \leftarrow$ Q-MEDLEY($M_{HQCL}, (X_{tr}, Y_{tr}), K_P$) ▷ Refers to Alg. 1
$V_{scores} \leftarrow$ PlotBarChart(L_{feat}, I_{feat}, title=M_{type} + Scores)
return $M_{HQCL}, I_{feat}, V_{scores}$

3.3 Visualization

The visualization module plays the role of an interpretative component in the QuXAI framework; it modifies the feature importance scores into a human-perceptible graphical display while retaining the values of the form vector $I = [I_1, I_2, \ldots, I_D]$ describing D features. The primary objective is to allow the users to readily understand the contributions of different features with reference to the output of the Q-MEDLEY explainer. If a given set of labels for features $L = [l_1, l_2, ..., l_D]$ and their importance scores I are provided, the visualization tool $V(L, I)$ generates a horizontal bar chart. The reason for adopting such a way is that it is capable of clearly illustrating and comparing the relative magnitudes of importance scores I_j for each feature l_j. By having a horizontal orientation, the labels of features are able to be shown without impairing the clarity of the chart, no matter how long the labels are. With the provision of a simple visual interface for achieving interpretation of importance scores, this method minimises the complexity

of feature attribution, leading to faster knowledge about which features matter the most in hybrid quantum-classical models.

The internal procedure for generating this visualization, $V(L, I)$, involves rendering each feature importance score I_j as the length of a bar associated with the label l_j. The construction can be conceptualized as creating a set of graphical elements where the j-th bar has a horizontal extent proportional to I_j. For clarity and standard presentation, the features are typically ordered such that the feature l_k with the maximum score $I_k = \max_j(I_j)$ is displayed at the top. This is achieved by sorting the pairs (l_j, I_j) based on I_j in descending order before rendering. Let the sorted importance scores be $I_{(1)} \geq I_{(2)} \geq \cdots \geq I_{(D)}$ with corresponding labels $L_{(j)}$. The visualization then maps these ordered pairs to a chart where the y-axis represents the feature labels $L_{(j)}$ and the x-axis represents the magnitude of the MEDLEY Score.

4 Results

4.1 Explanations generated by Q-MEDLEY

The Q-MEDLEY explainer, a key element of the QuXAI architecture, provides the abundance of medley scores, of which shed light on decision-making mechanisms of hybrid quantum-classical models (HQML). By systematically perturbing input features on a reference dataset and quantifying the damage to model performance–once the input has been quantum feature encoded and undergone classic processing–Q-MEDLEY assigns a numerical score to each of the classical features. This methodology offers a valuable window into the behaviour of these complex hybrid systems, taking us beyond the black box performance metrics to give feature level granularity.

The effectiveness of Q-MEDLEY in detecting informative features is evident in different data sets and HQML models settings, as depicted in Figure 3 for the Noisy Iris data set in the multipanel. In most of the ten HQML models encoded with amplitudes tested, Q-MEDLEY correctly picks up the same original semantically meaningful Iris characteristics, much more influential than the noisy and redundant features introduced synthetically. For example, the Q.RandomForest and Q.DecisionTree subplots of Figure 3 clearly indicate that the queried petal characteristics score much more highly in Q-MEDLEY than the synthetic ones. Such results reveal the ability of the explainer in following predictive influence to relevant classical inputs despite being transformed via a quantum feature map, thus justifying the model's emphasis on biologically relevant features.

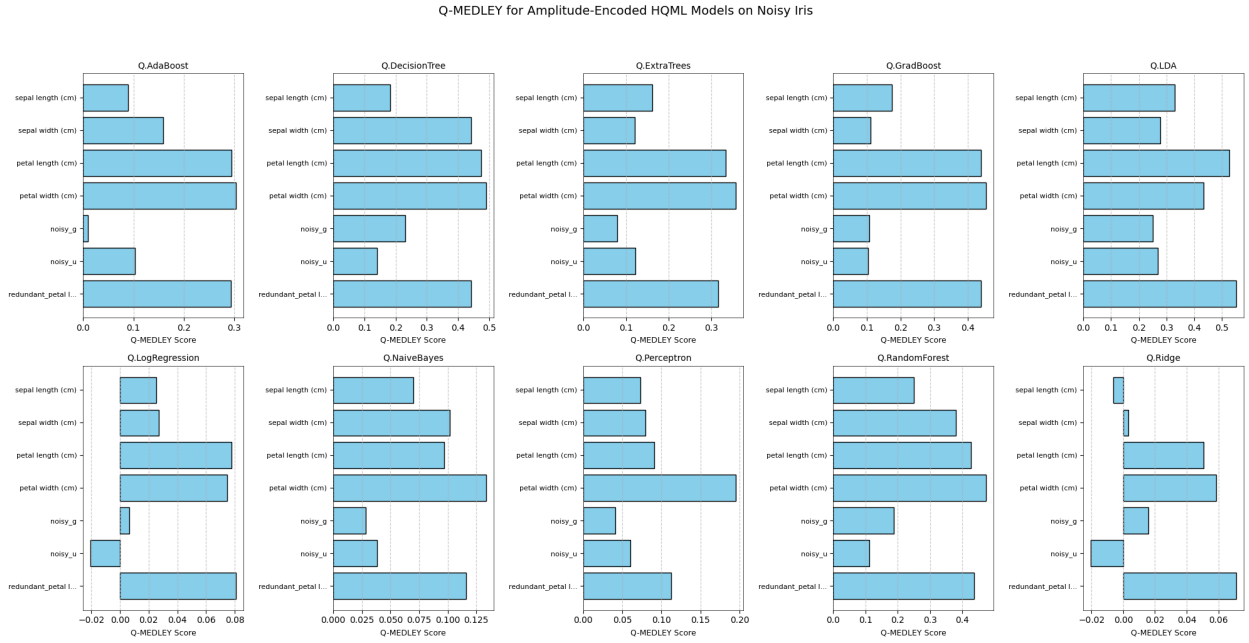


Figure 3: Q-MEDLEY scores for ten different amplitude-encoded HQML models trained on the Noisy Iris dataset. Each panel shows the importance attributed by Q-MEDLEY to each input feature for a specific HQML model type.

Equivalent understandings are obtained from the more featured Noisy Wine data set, which issues for ten HQML models displayed in the subplots of Figure 4. In this, Q-MEDLEY further emphasizes the relevance of the established indicators of wine quality as opposed to the artificially added noisy and redundant features for many of the HQML

models. Although the exact order of the most important features shows some fluctuation dependent on which particular classical learner is used within the HQML architecture (one can see this, for example, comparing Q.LDA subplot to Q.ExtraTrees subplot in Figure 4). This subtle pattern, described by Q-MEDLEY, is an important diagnostic signal, which shows how the different classical algorithms exploit the quantum representations inferred from the input features.

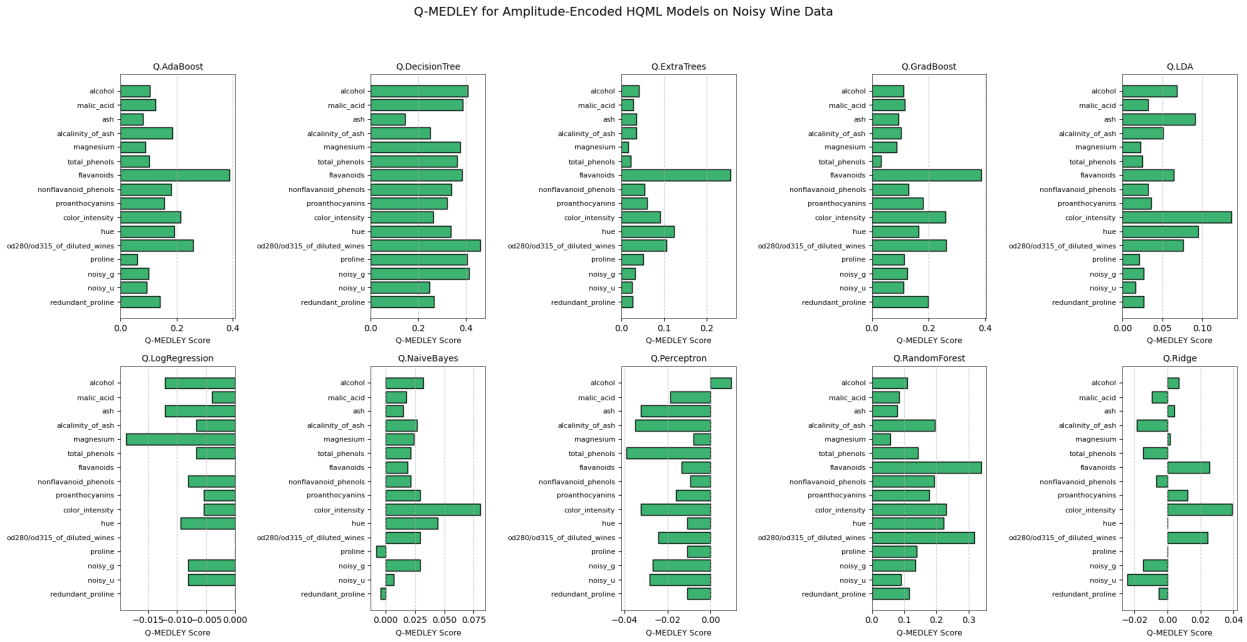


Figure 4: Q-MEDLEY scores for ten different amplitude-encoded HQML models trained on the Noisy Wine dataset. Each panel displays Q-MEDLEY scores, highlighting the relative importance of input features for various HQML model architectures.

The usefulness of these explanations enable us to increase the transparency and trustworthiness of HQML systems. Q-MEDLEY enables the investigation of how quantum feature encoding has effects on classical learning algorithms to produce model predictions by offering clear and quantitative feature attributions in each model-specific subplot (Figures 3 and 4). This interpretability is critical to model validation, whereby domain experts can tell whether given HQML models are learning scientifically plausible relationships, and debugging, where faulty architectures are flagged if irrelevant or noisy features are having disproportionate effects on outcomes in particular hybrid architectures.

4.2 Evaluating Q-MEDLEY

The Q-MEDLEY explainer was rigorously evaluated by benchmarking its performance compared to the known eXplainable AI (XAI) methodologies under a controlled classical machine learning environment. This involved the use of interpretable models and specifically trained Decision Tree and Random Forest classifiers on the Noisy Iris dataset whose features importances could be directly obtained from the models as ground truth. For purposes of evaluation, there were two critical metrics; Remember@3, a metric that evaluates how the percentage of correct true top-3 most important features is captured by an explainer, Spearman Rank Correlation, a metric that measures the extent to which the full feature importance ranking provided by an explainer complies with ground-truth ranking. Such measures reflect quantitative information on the accuracy of determining the most influential features, as well as global estimates of the faithfulness of the importance attributions.

As seen in Figure 5, the Recall@3 scores indicate that Q-MEDLEY possesses a competitive capability of distinguishing the most salient features. For the Decision Tree (Noisy) and Random Forest (Noisy) models, Q-MEDLEY performed similarly or better than the stand-alone Drop-Column Importance (DCI) and Permutation Importance (PI, k=5) in terms of Recall@3 scores. Indeed, Q-MEDLEY's performance was comparable to that of TreeSHAP, which is a well-known model-specific explainer for tree-based ensembles. For example, with the Random Forest model, Q-MEDLEY correctly identified a large proportion of the top-3 true features, proving its value in identifying key variables even when there are noisy and redundant features, a context posed to challenge explainability methods.

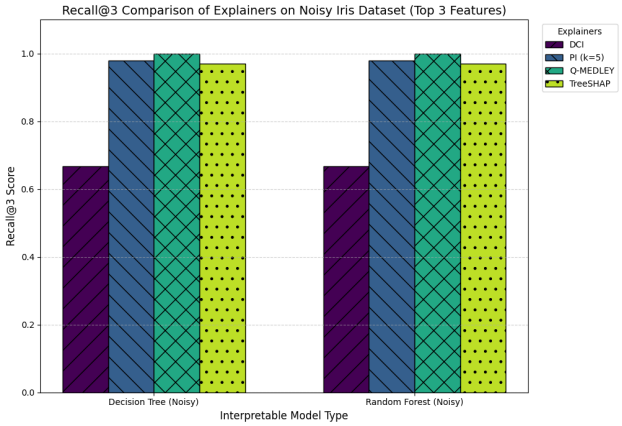


Figure 5: Recall@3 comparison of explainers (DCI, PI (k=5), Q-MEDLEY, TreeSHAP) on the Noisy Iris dataset for Decision Tree and Random Forest models, considering the top 3 identified features.

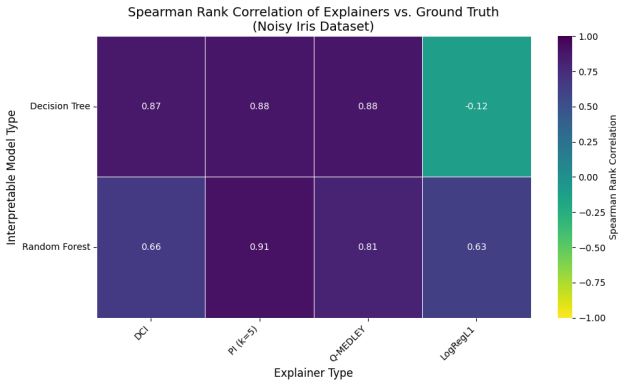


Figure 6: Spearman Rank Correlation heatmap comparing feature importance rankings from different explainers (DCI, PI (k=5), Q-MEDLEY, LogRegL1) against ground truth importances from Decision Tree and Random Forest models on the Noisy Iris Dataset.

Additional support of Q-MEDLEY's feature attribution fidelity is given by the Spearman Rank Correlation, depicted as a heatmap in Figure 6. This metric measures the monotonicity of the ground truth ranking and the full ranking of feature importances obtained for each explainer. In the case with the Q-MEDLEY, strong positive correlations with ground truth importances were always observed for the Decision Tree and the Random Forest models. For instance, providing support for Random Forest, Q-MEDLEY demonstrated reliable level of agreement and high Spearman correlation coefficient, remonstrating with basic DCI substantially and even seizing parity in the PI method. Its correlation was also markedly higher than what was observed with regard to the LogRegL1 coefficients (L1-regularization-based feature importance proxy used here to explain the Decision Tree) in explaining the referenced Decision Tree, again highlighting the superior performance of Q-MEDLEY in capturing the holistic importation landscape as shaped by these interpretable classical models.

4.3 Ablation Studies

To deconstruct the additions of the individual components of Q-MEDLEY and design selections, we performed ablation studies. This investigation systematically compared various setups of the Q-MEDLEY explainer from its baseline combination to more advanced versions that include an adaptive scoring of such constituent scores and a mechanism that makes the PI interaction aware. The research applied five different data sets with synthetically added noisy and redundant features to introduce a demanding task of explainability. For every dataset, interpretable classical models were trained and the intrinsic feature importances (e.g., Gini importance) were used as a ground truth. The performance of every Q-MEDLEY setting was qualified by the Recall @ 3 performance metric, indicating its performance in the top three most significant features according to the ground truth model. The results discussed below in Table 1 are informative with regard to the performance of each of the components.

Table 1: Q-MEDLEY Functional Component Ablation Study Results

Q-MEDLEY Configuration	BreastCancer		Covtype		Diabetes		Iris		Wine	
	DT	RF	DT	RF	DT	RF	DT	RF	DT	RF
Q-MEDLEY (DCI+PI Avg)	0.87	0.80	0.86	0.85	0.88	1.00	0.94	0.82	0.86	0.82
Q-MEDLEY + AdaptiveWeighting	0.89	0.83	0.87	0.86	0.89	1.00	0.93	1.00	0.88	0.81
Q-MEDLEY with InteractionPI	0.90	0.82	0.88	0.84	0.91	0.89	1.00	0.78	0.87	0.80
Q-MEDLEY + AdaptiveWeighting + InteractionPI	0.92	0.85	0.90	0.83	0.93	0.83	1.00	0.97	0.91	1.00

4.4 Implication of QuXAI framework

Table 2: Comparative Performance of Classical and Amplitude-Encoded HQML Models

Dataset	Model	Accuracy		F1-Macro		Precision-Macro		Recall-Macro	
		Classical	QuXAI	Classical	QuXAI	Classical	QuXAI	Classical	QuXAI
BreastCancer-Noisy	QAda	0.95	0.88	0.94	0.87	0.95	0.87	0.94	0.87
	QDT	0.91	0.88	0.91	0.88	0.91	0.88	0.90	0.88
	QExtra	0.95	0.88	0.95	0.87	0.96	0.88	0.94	0.87
	QGB	0.93	0.88	0.93	0.87	0.93	0.88	0.92	0.87
	QLDA	0.94	0.87	0.93	0.87	0.95	0.87	0.92	0.87
	QLogistic	0.98	0.88	0.98	0.87	0.98	0.88	0.98	0.87
	QNB	0.94	0.88	0.94	0.87	0.94	0.88	0.93	0.87
	QPerceptron	0.97	0.88	0.96	0.87	0.96	0.86	0.97	0.87
	QRF	0.95	0.87	0.94	0.87	0.95	0.87	0.94	0.87
	QRidge	0.94	0.88	0.94	0.87	0.95	0.88	0.93	0.87
Iris-Noisy	QAda	0.93	0.88	0.93	0.88	0.94	0.88	0.93	0.88
	QDT	0.91	0.87	0.91	0.87	0.92	0.87	0.91	0.87
	QExtra	0.91	0.90	0.91	0.90	0.92	0.91	0.91	0.90
	QGB	0.93	0.88	0.93	0.88	0.94	0.87	0.93	0.88
	QLDA	0.96	0.87	0.96	0.87	0.96	0.87	0.96	0.87
	QLogistic	0.87	0.88	0.87	0.87	0.87	0.87	0.87	0.88
	QNB	0.93	0.88	0.93	0.88	0.94	0.88	0.93	0.88
	QPerceptron	0.87	0.88	0.87	0.88	0.87	0.88	0.87	0.88
	QRF	0.93	0.88	0.93	0.88	0.94	0.88	0.93	0.88
	QRidge	0.87	0.88	0.87	0.88	0.87	0.88	0.87	0.88
Wine-Noisy	QAda	1.00	0.87	1.00	0.86	1.00	0.86	1.00	0.87
	QDT	0.85	0.86	0.85	0.86	0.85	0.86	0.85	0.86
	QExtra	0.91	0.90	0.83	0.86	0.86	0.86	0.86	0.87
	QGB	0.86	0.87	0.86	0.86	0.86	0.86	0.86	0.87
	QLDA	1.00	0.87	1.00	0.86	1.00	0.86	1.00	0.87
	QLogistic	1.00	0.87	1.00	0.86	1.00	0.86	1.00	0.87
	QNB	0.86	0.87	0.85	0.86	0.85	0.86	0.86	0.87
	QPerceptron	1.00	0.87	1.00	0.86	1.00	0.86	1.00	0.87
	QRF	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86
	QRidge	1.00	0.87	1.00	0.86	1.00	0.86	1.00	0.87

The ablation results summarized in Table 1 indicate a definite pattern of increasing performance by adding more advanced components into the Q-MEDLEY structure. Although, the baseline "Q-MEDLEY (DCI+PI Avg)" configuration already demonstrates good robust recall at 3 scores for different dataset-model combinations (e.g., yielding 1.00 for RF on Diabetes, 0.94 for DT on Iris), incremental addition of (learning) weighting and interaction-aware Remarkably, "Q-MEDLEY + AdaptiveWeighting + InteractionPI" set up, which corresponds to a fully-fledged version of the explainer, could obtain the maximum or among the maximum Recall@3 scores as well. For example, the full configuration alone was the only one to achieve a perfect Recall@3 of 1.00 on the Wine dataset using Random Forest model, and it is always strong in performance like 0.92 for DT on BreastCancer and 0.93 for DT on Diabetes. This implies that when optimally balancing the DCI and PI contributions, depending on relative signal, and using a PI variant that models feature interactions, Q-MEDLEY is better able to discriminate and rank the true drivers of model prediction more accurately in complex datasets with inter-feature dependencies and noise.

5 Discussion

In this work, we proposed QuXAI, a complete framework that purposedly aims towards helping training, evaluation and, importantly, explaining HQML model. At the heart of this framework is our new Q-MEDLEY explainer designed to output medley scores for HQML architectures that utilise quantum feature encoding. Our empirical inquires show the applicability of the HQML models considered and the effectiveness of Q-MEDLEY in revealing their traits.

5.1 Interpretation and Comparison with Existing Work

Interpretation of quantum machine learning models, and in particular, hybrid systems remain an active subject of investigation[17, 7, 9]. Although the methods such as the quantum Shapley values [1, 14, 25] or quantum LIME (Q-LIME) [9] can be used to explain certain features of quantum circuits or to explain Leveraging ensemble explanation ideas in the classical XAI [26], our approach diverges by explicitly incorporating the quantum transformation step into its internal prediction design while examining the feature perturbations. In contrast to generic model-agnostic explainers that may consider the whole HQML pipeline as one single black box, Q-MEDLEY's design preserves the hybrid nature of the data flow, and thus the effect of perturbing classical input features properly flows through the quantum encoding before being analyzed by the classical learners. The examination of meaningful features from noise in HQML models, as presented in Figures 3 and 4, hints at the possible usefulness of the Q-MEDLEY in exploring the learned representations in these hybrids.

5.2 Strengths of the QuXAI Framework and Q-MEDLEY

Our framework, QuXAI, as shown has various strengths. It delivers an end-to-end workflow (Algorithm 2) from data preparation and HQML model training to performance and, importantly, feature importance explanation through Q-MEDLEY. This synergetic technique is useful for a systematic study of interpretability of HQML. One of the major strengths of Q-MEDLEY is its particular adaptation to HQML models. its internal prediction method has been developed to reexamine quantum feature maps or kernels after perturbation so that explanations would be sensitive to the quantum part of the hybrid model. From the ablation studies (Table 1), the robustness and performance of Q-MEDLEY can be identified as even the base version performs on par with the advanced component like adaptive weighting and interaction-aware PI. In addition, the empirical validation against ground truth importances from classical interpretable models (Figures 5 and 6) vouches for its relevance in feature attribution functionality.

5.3 Limitations and Considerations

Despite these strengths, we acknowledge several limitations. First of all, our empirical validation of Q-MEDLEY has been primarily concentrated on amplitude-encoded HQML models. Although the core explainer class definition contains logic for kernel-based HQMLs, the results for these were not vast as the main feature of the described evaluation suite. this aspect should be further developed in future work. Secondly, Q-MEDLEY, as any perturbation-based explainers is time-consuming, specifically, when a substantial number of features are used for the dataset, or in the case of the usage of a high number of repeats for Permutation Importance. The scalability of quantum feature map simulations themselves, particularly, for increasing qubits (features), also proves to be a practical constraint. Finally, deriving conclusive "ground truth" feature importance for HQML models is a per se difficult task. Our practical reliance on classical interpretable models as proxies for validation may not a perfect reflection of all the intricacies of feature influence in the quantum domain.

5.4 Implications for Trustworthy and Interpretable Quantum AI

As quantum machine learning models transition from theoretical constructs to practical tools, the ability to understand why a model makes a certain prediction, or which input features drive its behavior, becomes paramount for user trust, model debugging, and regulatory compliance [17, 7]. Our work tries to fulfill this requirement for a unified class of near-term HQML architectures directly. By giving information on how classical features, which are transformed to quantum states, can benefit the final decision of a hybrid system, the Q-MEDLEY can contribute to research and practice by letting them: (i) validate whether HQML models are extracting meaningful patterns or are simply capitalizing on spurious correlations; (ii) debug the models to see if the noisy or irrelevant features are exerting excessive influence in determining the outcomes; and (iii) refine designs for quantum feature maps based on their consequences on the salience of the features. This enhanced transparency is crucial for the responsible development of QML, ensuring that these powerful new models are not only performant but also interpretable.

6 Conclusion

In this study, we have presented and illustrated the QuXAI framework, an integrated Quantum-XAI environment for hybrid quantum-classical machine learning models' design and interpretation. One of the key elements of our contribution is the Q-MEDLEY explainer designed to deliver feature importance attributions for the HQML systems that rely on quantum feature encoding. We have demonstrated that our suite of amplitude-encoded HQML models have strong predictive potential, hence they are relevant topics for explainability studies. Employing systematic examination via direct application to these HQML models and benchmarking in classical settings versus proven XAI approaches and true importances, we have validated Q-MEDLEY's capabilities to isolate informative features from noise and its sturdiness, which is supplemented systematically through ablation studies of its components. The representations generated from the use of Q-MEDLEY provide an easy way to understand the intricate relationship between classical data and quantum transformations involved in these new learning paradigms.

In future, we plan to extend our evaluations to a wider array of HQML architectures, including those based on quantum kernels and more sophisticated variational circuits, and to explore the impact of diverse quantum feature mapping strategies. Addressing the computational demands of perturbation-based explanations for larger-scale quantum systems remains a key challenge, alongside upgrading the existing local, instance-specific XAI methods for HQML. As we keep developing these tools and use them to tackle ever more complex, real-world problems, we expect to find such explainability frameworks quite useful for demystifying the quantum-enhanced machine learning, rendering it acceptable for everyday use, and guiding the co-design of both performant and interpretable quantum algorithms.

Declaration of Competing Interest

The authors declare no competing interests.

References

[1] R. Heese, T. Gerlach, S. Mucke, S. Muller, M. Jakobs, and N. Piatkowski. Explaining quantum circuits with shapley values: towards explainable quantum machine learning. *Quantum Machine Intelligence*, 7:27, 2023.

[2] D. Kreplin and M. Roth. Reduction of finite sampling noise in quantum neural networks. *Quantum*, 8:1385, 2024.

[3] N. Meyer, D. D. Scherer, A. Plinge, C. Mutschler, and M. Hartmann. Quantum policy gradient algorithm with optimized action decoding. *arXiv preprint arXiv:2212.06663*, 2022.

[4] C. Moussa, Y. J. Patel, V. Dunjko, T. Bäck, and J. N. van Rijn. Hyperparameter importance and optimization of quantum neural networks across small datasets. *Machine Learning*, 113:1941–1966, 2023.

[5] F. Scala, C. Zoufal, D. Gerace, and F. Tacchino. Towards practical quantum neural network diagnostics with neural tangent kernels. *arXiv preprint arXiv:2503.01966*, 2025.

[6] S. Mardirosian. Quantum-enhanced supervised learning with variational quantum circuits. 2019. Note: Abstract: S2 TL;DR: It is shown how near-term quantum devices open up a new avenue to combine quantum computing with classical machine learning methods, to achieve new quantum-enhanced classifiers, specifically using so-called variational quantum circuits for a supervised learning binary classification task.

[7] P. Steinmüller, T. Schulz, F. Graf, and D. Herr. eXplainable AI for quantum machine learning. *arXiv preprint arXiv:2211.01441*, 2022.

[8] S. Anagolum, N. Alavisamani, P. Das, M. Qureshi, and Y. Shi. Elivagar: Efficient quantum circuit search for classification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*, 2024.

[9] L. Pira and C. Ferrie. On the interpretability of quantum neural networks. *Quantum Machine Intelligence*, 6:52, 2023.

[10] B. Khanal, P. Rivas, A. Sanjel, K. Sooksatra, E. Quevedo, and A. R. Pérez. Generalization error bound for quantum machine learning in NISQ era - a survey. *Quantum Machine Intelligence*, 2024.

[11] A. Macaluso. Quantum supervised learning. *Künstliche Intelligenz*, 38:277–291, 2024.

[12] T. Hur, I. F. Araujo, and D. K. Park. Neural quantum embedding: Pushing the limits of quantum supervised learning. *Physical Review A*, 110:022411, 2024.

[13] Y. Du, Y. Qian, and D. Tao. Accelerating variational quantum algorithms with multiple quantum processors. *arXiv preprint arXiv:2106.12819*, 2021.

[14] I. Burge, M. Barbeau, and J. García. Quantum algorithms for shapley value calculation. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 01, pages 1–9, 2023.

[15] M. Ibrahim, H. Mohammadbagherpoor, C. Rios, N. Bronn, and G. T. Byrd. Evaluation of parameterized quantum circuits with cross-resonance pulse-driven entanglers. *IEEE Transactions on Quantum Engineering*, 3:1–13, 2022.

[16] J. Stein, T. Rohe, F. Nappi, J. Hager, D. Bucher, M. Zorn, M. Kölle, and C. Linnhoff-Popien. Introducing reducing-width-QNNs, an AI-inspired ansatz design pattern. *arXiv preprint arXiv:2306.05047*, 2023.

[17] R. Heese, T. Gerlach, S. Mucke, S. Müller, M. Jakobs, and N. Piatkowski. Explainable quantum machine learning. *arXiv preprint arXiv:2301.09138*, 2023.

[18] T. Haug, C. Self, and M. Kim. Large-scale quantum machine learning. *arXiv preprint arXiv:2108.01039*, 2021.

[19] C. Blank, A. J. da Silva, L. P. de Albuquerque, F. Petruccione, and D. Park. Compact quantum kernel-based binary classifier. *Quantum Science & Technology*, 7:045013, 2022.

[20] M. Incudini, M. Grossi, A. Ceschini, A. Mandarino, M. Panella, S. Vallecorsa, and D. Windridge. Resource saving via ensemble techniques for quantum neural networks. *Quantum Machine Intelligence*, 5:30, 2023.

[21] T. Haug, C. Self, and M. S. Kim. Quantum machine learning of large datasets using randomized measurements. *Machine Learning: Science and Technology*, 4:015019, 2023.

[22] A. Miroszewski, M. F. Asiani, J. Mielczarek, B. L. Saux, and J. Nalepa. In search of quantum advantage: Estimating the number of shots in quantum kernel methods. *arXiv preprint arXiv:2407.15776*, 2024.

[23] M. Kashif and S. Al-Kuwari. The impact of cost function globality and locality in hybrid quantum neural networks on NISQ devices. *Machine Learning: Science and Technology*, 4:015027, 2023.

[24] S. Barua, S. E. U. Shubha, M. Rahman, A. J. Uchash, and M. Mahdy. RESCUED: Robust quantum error correction with surface code in noisy channels using ensemble decoder. In *2023 IEEE International Conference on Telecommunications and Photonics (ICTP)*, pages 01–05. IEEE, 2023.

[25] I. Burge, M. Barbeau, and J. García. A quantum algorithm for shapley value estimation. *arXiv preprint arXiv:2301.04727*, 2023.

[26] S. Barua and S. Momen. Kaxai: An integrated environment for knowledge analysis and explainable AI. *arXiv preprint arXiv:2401.00193*, 2023.

[27] J. Stein, T. Rohe, F. Nappi, J. Hager, D. Bucher, M. Zorn, M. Kölle, and C. Linnhoff-Popien. Introducing reduced-width QNNs, an AI-inspired ansatz design pattern. In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence - Volume 3: ICAART*, pages 709–717, 2023.

[28] S. Park, D. Park, and J. Rhee. Variational quantum approximate support vector machine with inference transfer. *Scientific Reports*, 13:3159, 2023.

[29] D. Heimann, G. Schonhoff, E. Mounzer, H. Hohenfeld, and F. Kirchner. Learning capability of parametrized quantum circuits. *arXiv preprint arXiv:2209.10345*, 2022.

[30] F. Mercaldo, G. Ciaramella, G. Iadarola, M. Storto, F. Martinelli, and A. Santone. Towards explainable quantum machine learning for mobile malware detection and classification. *Applied Sciences*, 12(23):12025, 2022.

[31] S. Zhang, Z. Qin, Y. Zhou, R. Li, C. Du, and Z. Xiao. Single entanglement connection architecture between multi-layer HEA for distributed VQE. 2023. Note: No DOI or arXiv ID provided in Semantic Scholar entry for this one.

[32] S. Mangini, F. Tacchino, D. Gerace, D. Bajoni, and C. Macchiavello. Quantum computing models for artificial neural networks. *Europhysics Letters*, 134:10002, 2021.

[33] M. Ibrahim, H. Mohammadbagherpoor, C. Rios, N. Bronn, and G. T. Byrd. Pulse-level optimization of parameterized quantum circuits for variational quantum algorithms. 2022. Note: Abstract: S2 TL;DR: Pulse-level access to quantum machines and understanding of their two-qubit interactions are utilized to optimize the design of two- qubit entanglers in a manner suitable for VQAs, and results show that pulse-optimized ansatze reduce state preparation times by more than half, maintain expressibility relative to standard PQCs, and are more trainable through local cost function analysis. (Likely preprint or internal report if no other details).

[34] H.-Y. Wu, V. Elfving, and O. Kyriienko. Multidimensional quantum generative modeling by quantum Hartley transform. *Advanced Quantum Technologies*, page 2400337, 2024.

[35] Y. Wang and B. Qi. Enhanced generalization of variational quantum learning under reduced-domain initialization. In *2023 42nd Chinese Control Conference (CCC)*, pages 6771–6776, 2023.

[36] E. Gil-Fuster, J. Eisert, and C. Bravo-Prieto. Understanding quantum machine learning also requires rethinking generalization. *Nature Communications*, 15:1968, 2024.

[37] M. Weigold, J. Barzen, F. Leymann, and D. Vietz. Patterns for hybrid quantum algorithms. In W. Hasselbring, S. Eicker, and R. H. Reussner, editors, *Software Architecture - ECSA 2021 Tracks and Workshops*, volume 12884 of *Lecture Notes in Computer Science*, pages 18–31. Springer International Publishing, 2021.

[38] J. Liu, K. Najafi, K. Sharma, F. Tacchino, L. Jiang, and A. Mezzacapo. An analytic theory for the dynamics of wide quantum neural networks. *arXiv preprint arXiv:2203.16711*, 2022.

[39] T. Goto, Q. Tran, and K. Nakajima. Universal approximation property of quantum machine learning models in quantum-enhanced feature spaces. *Physical Review Letters*, 127(9):090506, 2021.

[40] C. Moussa, J. N. Rijn, T. Back, and V. Dunjko. Hyperparameter importance of quantum neural networks across small datasets. In T. Bäck, M. Preuss, A. Deutz, H. Wang, C. Doerr, M. Emmerich, and H. Trautmann, editors, *Parallel Problem Solving from Nature – PPSN XVII*, volume 13398 of *Lecture Notes in Computer Science*, pages 37–50. Springer International Publishing, 2022.

[41] D. Cugini, D. Gerace, P. Govoni, A. Perego, and D. Valsecchi. Comparing quantum and classical machine learning for vector boson scattering background reduction at the large hadron collider. *Quantum Machine Intelligence*, 5:21, 2023.

[42] Q. Li, Y. Huang, X. Hou, Y. Li, X. Wang, and A. Bayat. Ensemble-learning variational shallow-circuit quantum classifiers. 2023. Note: Abstract: S2 TL;DR: While both of the protocols substantially outperform error-mitigated primitive classifiers, the adaptive boosting shows better performance than the bootstrap aggregating, leading to a favorable complexity for practical realization. (Likely preprint if no other details).

[43] G. Ciaramella, G. Iadarola, F. Mercaldo, M. Storto, A. Santone, and F. Martinelli. Introducing quantum computing in mobile malware detection. In *Proceedings of the 17th International Conference on Availability, Reliability and Security (ARES '22)*, Article No. 61, pages 10, 2022.

Explaining Quantum Circuits with Shapley Values: Towards Explainable Quantum Machine Learning

Raoul Heese[1,*], Thore Gerlach[2], Sascha Mücke[3], Sabine Müller[1], Matthias Jakobs[3], and Nico Piatkowski[2]

[1]Fraunhofer ITWM, [2]Fraunhofer IAIS, [3]TU Dortmund
[*]raoul.heese@gmail.com

Abstract

Methods of artificial intelligence (AI) and especially machine learning (ML) have been growing ever more complex, and at the same time have more and more impact on people's lives. This leads to explainable AI (XAI) manifesting itself as an important research field that helps humans to better comprehend ML systems. In parallel, quantum machine learning (QML) is emerging with the ongoing improvement of quantum computing hardware combined with its increasing availability via cloud services. QML enables quantum-enhanced ML in which quantum mechanics is exploited to facilitate ML tasks, typically in the form of quantum-classical hybrid algorithms that combine quantum and classical resources. Quantum gates constitute the building blocks of gate-based quantum hardware and form circuits that can be used for quantum computations. For QML applications, quantum circuits are typically parameterized and their parameters are optimized classically such that a suitably defined objective function is minimized. Inspired by XAI, we raise the question of the explainability of such circuits by quantifying the importance of (groups of) gates for specific goals. To this end, we apply the well-established concept of Shapley values. The resulting attributions can be interpreted as explanations for why a specific circuit works well for a given task, improving the understanding of how to construct parameterized (or variational) quantum circuits, and fostering their human interpretability in general. An experimental evaluation on simulators and two superconducting quantum hardware devices demonstrates the benefits of the proposed framework for classification, generative modeling, transpilation, and optimization. Furthermore, our results shed some light on the role of specific gates in popular QML approaches.

1 Introduction

Machine learning (ML) has a significant impact on many applications in different domains. With the ongoing improvement of ML models and their growing complexity, another aspect is becoming increasingly important in addition to pure performance: the explainability of ML systems [1]. Explainability[1] refers to methods that make the behavior of ML systems—or, more generally, artificial intelligence (AI) systems [3]—comprehensible for humans. Realizing explainable AI (XAI) is a highly non-trivial task with a potentially great impact on many applications and can therefore be considered as an important research field. For example, XAI can be used to analyze the fairness of models to avoid discrimination, or their security against adversarial attacks, to name just two important aspects. Currently, reasoning about decisions is typically only possible to a very limited extent or even intractable for state-of-the-art ML models. A review of XAI exceeds the scope of this paper and we instead refer to, e.g., [1, 2, 4–6] and references therein.

Apart from explainability, another promising research direction of ML that has recently arisen with the emergence of new technology is quantum machine learning (QML) [7, 8]. This field addresses how quantum computers (or quantum information processing in general [9]) can be used to provide a benefit for ML. To identify possible advantages, the structure of the data seems to be key [10–

[1]We do not distinguish between *explainability* and *interpretability* because these terms are not strictly defined and are often used interchangeably in the ML community [2].

12]. However, since currently only noisy intermediate-scale quantum (NISQ) devices [13] with limited capabilities are available, a practical application of QML is restricted to toy examples. On the other hand, the identification of a quantum advantage is not necessarily of central importance for current research [14]. Instead, the study of fundamental aspects of QML and their prospective exploitation with better hardware in the future might be more promising.

With this in mind, it seems natural to study explainability for QML in a similar way as for *classical* (i.e., non-quantum) ML, giving rise to *explainable QML* (XQML) with the goal to provide humanly understandable interpretations of QML systems. This aspect of QML has been virtually unexplored at the time of the initial submission of this manuscript, except for one other study in the context of malware detection [15]. In the meantime, however, there have been a number of new research efforts that indicate a growing awareness of the topic. For example, in the form of review articles [16, 17] that also introduce novel methods [18] as well as articles focusing on quantum neural networks [19–21], the visualization of model behavior [22, 23], or reinforcement learning [24, 25], just to name a few.

XAI methods can be divided into model-specific approaches and model-agnostic approaches. While the former are limited to specific model classes, the latter can be used for arbitrary models. Hence, model-agnostic XAI methods can be applied to QML models in the same way as to classical ML models. This leaves us with a rich set of classical tools that can be directly explored and potentially refined for XQML [16, 18]. Since QML is still in a very early stage in comparison to ML, explainability can be adopted from the beginning for comparatively simple tasks that are much easier to analyze than the classical state-of-the-art problems. Their exploration could lead to fundamental insights, similar to the classical analog.

In the study of the decision making process of classical ML systems, two central, interrelated questions are typically addressed by XAI:

1. What is the influence of the model inputs (i.e., features) on its predictions?

2. How does the model architecture (its components and/or trainable parameters) affect its predictions?

The first question takes an external perspective by investigating the (potentially black-box) relationship between inputs and outputs, and can be considered the more commonly asked. The second shifts attention inward, exploring how the architecture and parameters of the model itself affect its behavior.

In the present paper, we limit ourselves to the second perspective and propose a framework for approaching model architecture investigations in an XQML context. Specifically, our framework allows to explore the influence of gates (or groups of gates) as the building blocks of gate-based quantum computations using Shapley values (SVs) [26, 27], a well-known model-agnostic tool in XAI [28–30]. While our considerations are generally applicable to any kind of gate-based quantum computation, we focus mostly on QML applications driven by our original motivation for such an analysis. We refer to our approach as *Shapley values for quantum circuit explanations* (SVQXs).

The study of internal structures of ML models is an important topic in deep learning, for example to visualize the behavior of complex systems [31]. Often, it is closely related to the search for an optimal model design, especially in the context of neural networks, where it is known as *neural architecture search* (NAS) [32, 33] and can also be based on SVs [34–38]. NAS is a special branch of automatic machine learning (AutoML), a research field concerned with the automation of the entire ML pipeline [39]. This concept has also been transferred to QML, where it is known as *automatic quantum machine learning* (AutoQML) [40, 41]. In this connection, several approaches have been proposed to optimize quantum circuit designs for QML applications in the sense of a *quantum circuit architecture search* (QAS) [42–46]. While our proposed SVQXs can in principle also be used in the context of QAS, we focus on the explainability aspect in this paper and demonstrate its potential for circuit design optimization only by way of example.

Our main contributions can be listed as follows:

- We present SVQXs as a method to measure the impact of every gate in a (variational) quantum circuit on the overall expressibility, entanglement capability, or any arbitrary quantity of interest.

- The proposed method is grounded in the theory of uncertain SVs and therefore extends this domain by a quantum approach while facilitating inherent robustness to the noise from NISQ devices.

- SVQXs can be leveraged for XQML, which is the primary focus in this work.
- We perform experiments on simulators and actual quantum hardware to showcase different use cases.
- A software toolbox and the data from our experiments are publicly available online.

The remainder of this manuscript is divided into four sections. In Sec. 2, we introduce the prerequisites for our proposed method, which we present in Sec. 3. We subsequently study experimentally in Sec. 4 how our method can be used for a variety of applications. Finally, we conclude with a summary and outlook in Sec. 5.

2 Background

In the present section, we outline the prerequisites for our proposed XQML method of SVQXs. For this purpose, we first present a short summary of SVs and how they can be applied to classical ML. Next, we give a brief introduction to QML.

2.1 Shapley values

SVs [26, 27] are a concept originating in game theory that can be used to evaluate the performance of individual players in a coalition game. By definition, they fulfill many desirable properties that distinguish them as a particularly suitable evaluation measure for this purpose. Presume a group of $N \in \mathbb{N}$ players denoted by the set $\mathfrak{S} := \{1, \ldots, N\}$, a coalition game is defined by the value function (or characteristic function)

$$v : \mathcal{P}(\mathfrak{S}) \to \mathbb{R}, \tag{1}$$

that assigns a (real-valued) payoff $v(S)$ to each subset (or *coalition*) of players $S \subseteq \mathfrak{S}$. The SV of player $i \in \mathfrak{S}$ is then given by the expectation value

$$\phi_i := \mathbb{E}[\boldsymbol{V}_i] \tag{2}$$

of the random variable[2] $\boldsymbol{V}_i \sim \mathbb{V}_i$ with probability mass function

$$\mathbb{V}_i(v) := \sum_{S \subseteq \mathfrak{S} \setminus \{i\}} w(S) \cdot \mathbb{I}(i, S, v). \tag{3}$$

Here, we have introduced the weight

$$w(S) := \frac{1}{N} \binom{N-1}{|S|}^{-1} = \frac{|S|!\,(N - |S| - 1)!}{N!} \tag{4}$$

of coalition S based on its set cardinality $|S|$. Furthermore,

$$\mathbb{I}(i, S, v) := \begin{cases} 1, & \text{if } \Delta_i v(S) = v \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

denotes the indicator function with the marginal contribution

$$\Delta_i v(S) := v(S \cup \{i\}) - v(S) \tag{6}$$

of player i to the coalition S. In total, there are 2^N different coalitions including the grand coalition $\mathfrak{S}$ and the empty coalition $\varnothing$.

The SV framework is very general with typical applications in economy and finance [47, 48] as well as XAI [28]. Due to the computational complexity of exact calculations, random sampling strategies can be employed to obtain approximate results [49]. A quantum algorithm for the calculation of (classical) SVs in an XAI context was recently proposed in [50].

[2] All random variables are in bold.

In our subsequent analyses, we apply SVs to quantum circuits. For this purpose, gates represent players and the value function is obtained from measurement results, for example to quantify the contribution of each gate to produce a desired distribution of bits. Due to the universality of the theory, the value function can be chosen in many different ways depending on the specific use case. Our approach is explained in more detail in Sec. 3, where we also give detailed examples for possible value functions.

2.2 Shapley values under uncertainty

So far, we have considered a deterministic value function $v(S)$, Eq. (7). However, for value functions which are based on quantum processing unit (QPU) calculations, such a deterministic behavior is not given since results from QPUs are subject to

- the statistical uncertainty of measurements that results from the inherent uncertainty of quantum physics and the limited number of shots,
- the typically non-deterministic mapping of logical qubits onto physical qubits together with a decomposition of the circuit unitary that can be realized on the hardware in form of elementary gates (i.e., transpilation), and
- the hardware-related noise of NISQ devices as a result of, e.g., decoherence and dissipation effects [51] as well as state preparation and measurement (SPAM) errors [52].

Consequently, in addition to randomness that occurs in classical ML algorithms (e.g., dropout [53]), additional randomness may arise in the context of QML. Handling value functions under randomness, noise or uncertainty is therefore essential for deriving meaningful SVs for QML. Recently, a theoretical framework for such value functions has been proposed [54]. Following this reference, we consider an uncertain value function of the form

$$\boldsymbol{v}(S) := v(S) + \boldsymbol{\nu}(S) \tag{7}$$

based on $v(S)$ from Eq. (1) with an additive noise given by the random variable $\boldsymbol{\nu}(S) \sim \mathbb{Q}(\cdot \mid S)$ that follows an arbitrary but fixed probability density $q(\cdot \mid S)$ for all $S \subseteq \mathfrak{S}$. Here, $\boldsymbol{\nu}(S)$ is assumed to cover all sorts of noise which arise from QPU calculations. This form of additive noise implies a specific structure of the uncertainty under consideration. However, since the additive noise term in Eq. (7) explicitly depends on the coalition S, we can in fact use it to formally describe any kind of noisy behavior. As also already pointed out in [54], it remains an open research question to evaluate other types of noise representations. In the following, we limit ourselves to the additive form. SVs for such uncertain value functions can then be defined in analogy to Eq. (8) as an expectation value

$$\Phi_i := \mathbb{E}[\boldsymbol{U}_i] \tag{8}$$

of the random variable $\boldsymbol{U}_i \sim \mathbb{P}_i$ with probability density

$$p_i(u) = \sum_{S \subseteq \mathfrak{S} \setminus \{i\}} w(S) \int_{\mathbb{R}} q(\nu \mid S \cup \{i\})\; q(\nu + \Delta_i v(S) - u \mid S)\, \mathrm{d}\nu, \tag{9}$$

where we recall Eqs. (4) and (6). It can be shown that SVs of uncertain value functions as defined in Eq. (8) represent *regular* SVs in the sense of Eq. (2) with a deterministic but shifted value function. Therefore, all properties of regular SVs also apply for SVs of uncertain value functions. We refer to [54] and references therein for a more detailed discussion.

In practice, the probability density $q(\cdot \mid S)$ is typically unknown (e.g., because the statistics of the noise are unknown) and Φ_i can consequently only be estimated. In the following, we briefly present two possible estimators, which are also outlined in [54]. For the first approach, the value function is sampled $K \in \mathbb{N}$ times for every coalition $S \in \mathfrak{S}$. An unbiased estimator for Φ_i is then given by the sample mean

$$\hat{\Phi}_i^K := \frac{1}{K} \sum_{S \subseteq \mathfrak{S} \setminus \{i\}} w(S) \sum_{k=1}^{K} \left[\tilde{v}^k(S \cup \{i\}) - \tilde{v}^k(S) \right], \tag{10}$$

where $\tilde{v}^k(S)$ represents the kth realization of the value function $\boldsymbol{v}(S) \sim \mathbb{Q}(\,\cdot - v(S) \mid S)$ for $k \in \{1, \ldots, K\}$. In total, this approach requires $2^N K$ value function samples. Due to its linearity, Eq. (10) can also be understood as a mean of K SVs, each based on one value function sample.

An alternative estimator can be employed without iterating K times over all 2^N possible coalitions. For this purpose, the coalitions themselves are treated as a random variable $\boldsymbol{S} \sim w$, Eq. (4), and $n \in \mathbb{N}$ random samples are drawn from the set $\mathfrak{S} \setminus \{i\}$ for each player i, which leads to the multiset of realizations $\mathfrak{S}_i^{\mathrm{r}}$. For each realization $S \in \mathfrak{S}_i^{\mathrm{r}}$, the value functions $\boldsymbol{v}(S) \sim \mathbb{Q}(\,\cdot - v(S) \mid S)$ and $\boldsymbol{v}(S \cup \{i\}) \sim \mathbb{Q}(\,\cdot - v(S) \mid S \cup \{i\})$ are both sampled K times. In total, $2nK$ value functions are sampled for each player i. Once the sampling for all N players is completed, all value function samples are collected in the multigroup $\mathcal{V}^{\mathrm{r}}$. Thus, a (possibly empty) multiset of realizations $\mathcal{V}^{\mathrm{r}}(S) := \{\tilde{v}^1(S), \ldots, \tilde{v}^{|\mathcal{V}^{\mathrm{r}}|}(S)\}$ can be assigned to every coalition $S \in \mathfrak{S}$ from the total collection $\mathcal{V}^{\mathrm{r}}$. The latter corresponds to the additive union $\mathcal{V}^{\mathrm{r}} = \biguplus_{S \in \mathfrak{S}} \mathcal{V}^{\mathrm{r}}(S)$. Similar to Eq. (10), $\tilde{v}^k(S)$ represents the kth realization of the value function for $k \in \{1, \ldots, |\mathcal{V}^{\mathrm{r}}(S)|\}$, where $|\mathcal{V}^{\mathrm{r}}(S)| \geq 0$ denotes the number of value function samples for the coalition $S \in \mathcal{V}^{\mathrm{r}}$. In conclusion, the expression

$$\hat{\Phi}_i^{n,K} := \frac{1}{n} \sum_{S \in \mathfrak{S}_i^{\mathrm{r}}} \left[\frac{1}{|\mathcal{V}^{\mathrm{r}}(S \cup \{i\})|} \sum_{v \in \mathcal{V}^{\mathrm{r}}(S \cup \{i\})} v - \frac{1}{|\mathcal{V}^{\mathrm{r}}(S)|} \sum_{v' \in \mathcal{V}^{\mathrm{r}}(S)} v' \right] \tag{11}$$

also represents an unbiased estimator for Φ_i. In total, it requires up to $2nNK$ value function samples. This estimator can therefore be evaluated with less value function samples than Eq. (10), if n is chosen such that $n < 2^{N-1}/N$ (presuming that K is chosen equally).

SVs are defined as expectation values of random variables. The corresponding higher moments consequently allow further insight into the probability distribution of these random variables. For example, the standard deviation

$$\sigma[\Phi_i] := \sqrt{\mathbb{E}[\boldsymbol{U}_i^2] - \Phi_i^2} \tag{12}$$

can be considered as a measure of the effective uncertainty of Φ_i.

2.3 Shapley values for classical machine learning

In order to lay out how SVs can be applied to quantum circuits, we recall their usage in the context of ML [55]. We specifically limit ourselves to two types of applications that are related to our work: (i) SVs as a model-agnostic XAI tool to explain the influence of model inputs on predictions and (ii) SVs as part of a NAS framework with the purpose to analyze the importance of the building blocks of neural networks. We only provide a brief summary of these two applications here, a detailed review of SVs for ML is beyond the scope of this paper.

For the first application, the use of SVs as a model-agnostic XAI tool [28–30, 35, 36, 38], features (i.e., distinct dimensions of a data point used as the model input) take the role of players and the value function is based on the corresponding model output. SVs of such a coalition game can then be used as an importance measure of each feature. This can be considered as the "traditional" usage of SVs in XAI.

Explanations are typically generated for a single data point, consisting of an N-dimensional feature vector $\vec{x} \in \mathbb{R}^N$. In this case, each feature $i \in \{1, \ldots, N\}$ corresponds to a player and the value function is of the form

$$v(S) = v(S; \vec{x}, f), \tag{13}$$

where f denotes the model of interest. An alternative approach is to aggregate a score over multiple or all data points, which leads to feature importances with respect to an entire data set $\{\vec{x}_1, \ldots, \vec{x}_d\}$ consisting of data points $\vec{x}_i \in \mathbb{R}^N$ for $i \in \{1, \ldots, d\}$. Consequently, the value function is of the form

$$v(S) = v(S; \{\vec{x}_1, \ldots, \vec{x}_d\}, f) \tag{14}$$

in analogy to Eq. (13).

In general, value functions of the form presented in Eqs. (13) and (14) can be realized in many different variants leading to SVs with different interpretations [56]. To evaluate coalitions of features, a common approach is to eliminate all features not present in a coalition S, introduced as SHAP by [29]. To this end, the expectation of the model output is taken w.r.t. all features not present in the coalition,

$\mathbb{E}[f(\boldsymbol{X}) \mid x_S]$. Since computational complexity is a major issue for practical applications [57], this expectation is usually approximated by sampling from a given data set. Additional assumptions like feature independence and linearity can reduce the computational cost further. Specialized methods, e.g., KernelSHAP [29], employ locally linear surrogate models to approximate SVs. There are also critical aspects of SVs in this context as for example studied in [58, 59].

For the second application, the use of SVs as part of a NAS framework [34–38], building blocks of a neural network constitute the players, and the value function is based on the performance of the pruned network, which contains only the building blocks from the respective coalition.

Since (game theoretic) NAS can be implemented in many different ways [35], we present here a high-level description that focuses on the key ideas. A typical approach is to choose a specific ML task (e.g., image classification on a particular data set) and an appropriate neural network to solve this task. Then, the network is decomposed into c mutually excluding structural components (e.g., layers or parts of layers). Such a decomposition is not unique and each choice leads to a different coalition game. Formally, each structural component $i \in \{1, \ldots, c\}$ corresponds to a player and the value function is of the form

$$v(S) = v(S; f, t), \tag{15}$$

where f denotes the neural network of interest and t the ML task (including the data). To evaluate Eq. (15), f is first pruned to contain only the structural elements that represent the players in the coalition S. Then, the pruned model is trained on the task t and a predefined performance metric (e.g., classification accuracy) is determined as the outcome of $v(S)$.

Within the NAS framework, the resulting SVs can then for example be used to iteratively prune the network by removing the structural components that are associated with the smallest SVs [34]. As for the first XAI application of explaining features, computational complexity is again a major issue for the practical realization of a SV-based NAS. A common solution is to use a random sampling approximation [49].

2.4 Quantum machine learning

The interdisciplinary field of QML lies at the interface of ML and quantum computing [7, 8]. It encompasses several research directions that consider data and algorithms that may both be classical, quantum, or a hybrid combination thereof. In this manuscript, we focus on quantum-enhanced ML [60], which considers the use of quantum algorithms to improve upon ML tasks for classical data. This is typically achieved by hybrid algorithms that include a classical and a quantum part. Such kind of algorithms can already be run on current NISQ devices.

In Fig. 1, we show a simplified sketch of such a *hybrid quantum-classical ML pipeline*, which we also shortly refer to as *hybrid ML pipeline* in the following. Classical data is presented to a classical *host*—typically consisting of (one or more) central processing units (CPUs), graphics processing units (GPUs), and random-access memory (RAM)—, which communicates with a QPU. A quantum circuit is executed on the QPU. This circuit is controlled by features and variational parameters from the classical host. Features in this context refer to appropriately preprocessed classical data points that can be encoded in the circuit by a suitable combination of gates such that the resulting quantum state contains the respective information, for example in form of amplitudes or angles [61, 62]. We denote a classical k-dimensional feature vector by $\vec{x} \in \mathcal{X} \subseteq \mathbb{R}^k$.

Furthermore, the variational parameters represent the degrees of freedom of the circuit depending on the chosen ansatz. An ansatz in this sense describes a particular sequence of gates, where each gate can depend on one or more variational parameters. We denote a p-dimensional vector of variational parameters by $\vec{\theta} \in \mathcal{P} \subseteq \mathbb{R}^p$. Quantum circuits depending on such parameters are also known as parameterized or *variational circuits* and are fully defined by a unitary operator $U(\vec{x}, \vec{\theta})$ acting on the joint Hilbert space of all q qubits that have initially been prepared in the ground state $|0\rangle$ as shown in Fig. 1. Hence, the unitary operator yields a final quantum state

$$|\Psi(\vec{x}, \vec{\theta})\rangle := U(\vec{x}, \vec{\theta})\,|0\rangle \tag{16}$$

of the joint system of qubits.

After execution, the measurement results are returned to the classical host. For s shots (or repeated circuit evaluations per execution), the measurement results consist of a multiset of s bit strings

$$\vec{B}(\vec{x}, \vec{\theta}) \equiv \vec{B} := \{\vec{b}_1, \ldots, \vec{b}_s\}, \tag{17}$$

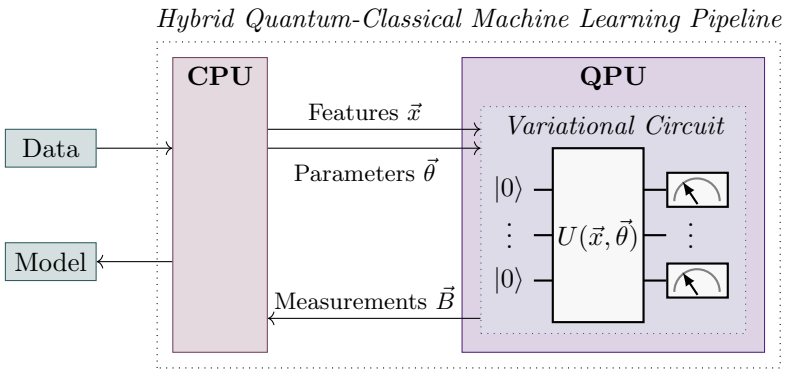


Figure 1: Simplified sketch of a *hybrid quantum-classical ML pipeline* (or *hybrid ML pipeline* for short) in form of a variational quantum algorithm. Preprocessed features $\vec{x}$ and parameters $\vec{\theta}$ from a classical host (represented here by a CPU) control a variational circuit that is executed on a QPU. The measurement results $\vec{B}$ are returned to the classical host (in form of bit strings) and enable a quantum classical optimization loop for the parameters. In the end, a suitable model for the proposed ML problem can be realized. Prospectively, such a pipeline involves multiple CPUs and QPUs as well as GPUs [63].

each of which represents the measured states of all q qubits in form of a vector $\vec{b}_i \in \{0,1\}^q \, \forall \, i \in \{1,\ldots,s\}$. In this sense, each bit string $\vec{b}_i$ can be considered as a realization of a random variable $\vec{\boldsymbol{b}}$ with probability mass

$$\mathbb{P}_{\vec{x},\vec{\theta}}(\vec{b}) := \langle \Psi(\vec{x},\vec{\theta}) | M(\vec{b}) | \Psi(\vec{x},\vec{\theta}) \rangle \, , \tag{18}$$

where $M(\vec{b})$ represents the positive operator-valued measure (POVM) element of the measurement corresponding to the outcome $\vec{b}$ [9]. Summarized, the variational quantum circuit can be understood as a mapping

$$\vec{x},\vec{\theta} \mapsto \vec{B} \tag{19}$$

from features $\vec{x}$ and variational parameters $\vec{\theta}$ to a realization $\vec{B}$ of a random variable $\vec{\boldsymbol{B}}$.

Typically, the variational parameters are optimized with a classical optimizer during training such that a suitable model for the proposed ML problem can eventually be realized after a certain number of quantum-classical iterations. Since the variational circuit is of central importance, this kind of hybrid ML pipeline is also referred to as variational quantum algorithm (VQA) [64]. The resulting model involves a classical host and (optionally) a QPU for inference.

Classical ML and QML problems can be divided into three different categories:

- *Supervised learning.* Given a labeled dataset, the goal is to infer the label for unknown samples (see, e.g., [65]).
- *Unsupervised learning.* The task is to learn patterns of unlabeled data, for example to perform clustering (see, e.g., [66]).
- *Reinforcement learning.* An agent is trained to learn actions that maximize a given reward function (see, e.g., [67]).

All three problem categories can be approached with the aforementioned hybrid ML pipeline based on variational circuits.

For this purpose, measurement results are mapped onto a result value

$$h : \vec{B} \mapsto \vec{y} \in \mathcal{Y} \tag{20}$$

in some solution space $\mathcal{Y}$. For classification or clustering, $\mathcal{Y}$ may represent a set of class labels, whereas for regression, it can correspond to (a subset of) $\mathbb{R}^t$ for t-dimensional targets. For reinforcement learning, $\mathcal{Y}$ may be a set of feasible policies for an agent.

In each of those ML scenarios, there are common loss functions (or objectives) which are used to adjust the variational parameters $\vec{\theta}$ during training, e.g., through gradient descent or evolutionary optimization. Typically, a loss function

$$\ell : \{(\vec{x}_1,\vec{y}_1),\ldots,(\vec{x}_d,\vec{y}_d)\}, \vartheta \mapsto \mathbb{R} \tag{21}$$

is aggregated over a training dataset $\{(\vec{x}_1, \vec{y}_1), \ldots, (\vec{x}_d, \vec{y}_d)\}$ consisting of d tuples of feature vectors $\vec{x}_i \in \mathcal{X}$ and corresponding solutions $\vec{y}_i \in \mathcal{Y}$ for all $i \in \{1, \ldots, d\}$, where each of the latter is determined via Eq. (20) using a respective measurement to obtain Eq. (17). Additionally, ℓ may depend on meta data ϑ such as ground-truth labels in supervised learning, depending on context. Examples for loss functions include mean squared error, cross entropy loss or log-likelihood.

3 Shapley values for quantum circuits

In Sec. 2.4, we have outlined a typical hybrid quantum-classical ML pipeline in form of a VQA as sketched in Fig. 1. It relies on the execution of variational quantum circuits, which are defined by a unitary operator $U(\vec{x}, \vec{\theta})$ acting on the joint Hilbert space of all q qubits that have initially been prepared in the ground state $|0\rangle$. Moreover, it depends on the k-dimensional features $\vec{x} \in \mathcal{X} \subseteq \mathbb{R}^k$ as well as the p-dimensional variational parameters $\vec{\theta} \in \mathcal{P} \subseteq \mathbb{R}^p$.

Since SVs enable model-agnostic explainability, they can be applied to QML models (i.e., models, which result from a quantum-enhanced ML pipeline) in complete analogy to the "traditional" XAI approach for classical ML, where features represent players and the model output determines the value function [16, 18].

In this manuscript, we go beyond this feature-based application and propose a more subtle use of SVs that is centered around the architecture of QML models. Specifically, presume that a hybrid ML pipeline or its resulting model contains a (possibly parameterized) circuit for quantum evaluations. Our goal is to find an interpretation or explanation for this crucial component within the pipeline or model. For this purpose, we consider the gates (or groups of gates) as players (or their respective indices, to be more precise) and a value function that is determined by the measurement results. This approach allows us to assign a SV to every gate of the circuit with respect to the chosen value function and hence assign a corresponding contribution. Consequently, we are able to evaluate the quality of different ansätze and their building blocks in great detail. We refer to our approach as *Shapley values for quantum circuit explanations* (SVQXs)[3].

The premise of SVQXs is conceptionally related to SV-based NAS in the sense that we consider building blocks of ML models as players. However, we present our approach in a way that is customly tailored for quantum computations and yet highly generic. Furthermore, we focus on its use for explainability rather than using it within a QAS framework.

In summary, we identify two approaches for applying SVs in QML as sketched in Fig. 2: either using features as players (the "traditional" XAI approach) or using gates as players (what we consider in this work with SVQXs). Both concepts found on the same premise of a coalition game.

Game theory in the context of quantum computing has been considered before; see, e.g., [68] for a recent review of (non-cooperative) quantum games. SVQXs allow to assign a contribution to individual gates as a whole, where the gates can either be parameterized with a fixed parameter value or non-parameterized. This is in contrast to other approaches that measure the importance of the gate parameters themselves, for example with gradient-based methods [69].

In the remaining part of this section, we first introduce a formal definition of SVQXs. Next, we motivate and present a selection of possible value functions that can be used for various QML use cases. We conclude with a brief summary of the key ingredients of our proposed method.

3.1 Definition of SVQXs

To implement SVQXs, we decompose the unitary operator

$$U(\vec{x}, \vec{\theta}) = \prod_{g \in \{1, \ldots, G\}} U_g(\vec{x}^g, \vec{\theta}^g) := U_G(\vec{x}^G, \vec{\theta}^G) \cdots U_1(\vec{x}^1, \vec{\theta}^1) \tag{22}$$

from the quantum circuit of interest, Eq. (16), into G unitary operators $U_g(\vec{x}^g, \vec{\theta}^g)$ that represent gates (or groups of gates) within the circuit and may depend on (a possibly empty subset of) features

[3] In an earlier version of the manuscript, we used the term *quantum Shapley values* (QSVs). However, we have changed this term to better reflect the fact that we are applying classical (uncertain) Shapley values to quantum gates and quantum-based value functions without actually extending the concept of Shapley values themselves to a quantum version.

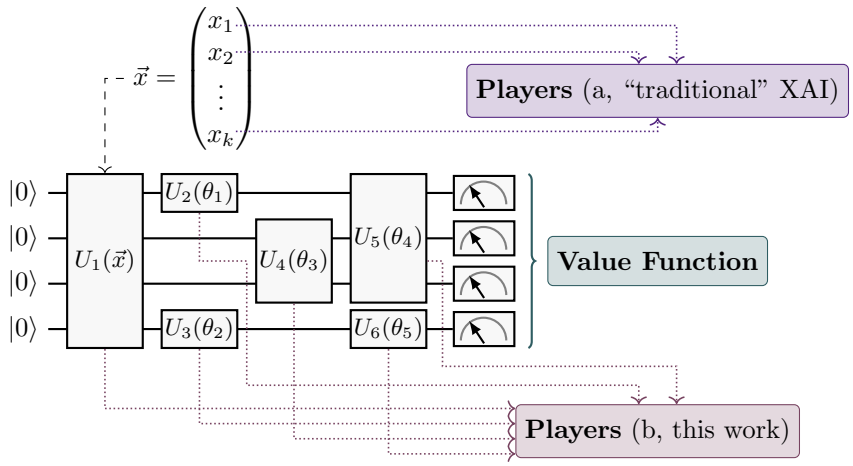


Figure 2: Two possible approaches for SVs in QML involving variational quantum circuits: (a) SVs from classical ML for which features represent players. (b) Newly proposed SVQXs for which quantum gates represent players. In both cases, the value function, Eq. (30), is determined by the measurement results. Shown is an exemplary four-qubit variational circuit consisting of a data encoding unitary for a k-dimensional feature vector $\vec{x} := \{x_1, \ldots, x_k\}$ and a set of five parameterized gates as outlined in Fig. 1, where each parameterized gate depends on a single variational parameter $\vec{\theta}^{i+1} := \{\theta_i\} \subset \vec{\theta} \forall\, i \in \{1, \ldots, 5\}$ with $\vec{\theta} := \{\theta_1, \ldots, \theta_5\}$. In terms of Eq. (22), $G = 6$, $U_1(\vec{x}^1, \vec{\theta}^1) := U_1(\vec{x})$, and $U_{1+i}(\vec{x}^{1+i}, \vec{\theta}^{1+i}) := U_{1+i}(\theta_i) \,\forall\, i \in \{1, \ldots, 5\}$. For both of the SV approaches presented here, the gate parameters $\vec{\theta}$ are assumed to be arbitrary but fixed.

$\vec{x}^g \subseteq \vec{x}$ and variational parameters $\vec{\theta}^g \subseteq \vec{\theta}$, respectively, for all $g \in \{1, \ldots, G\}$. The product in Eq. (22) is to be understood in such a way that the terms are ordered as indicated.

Furthermore, we define a set

$$A := \{A_1, \ldots, A_N\} \subseteq \{1, \ldots, G\} \tag{23}$$

of active gates, which participate as players, whereas the remaining gates

$$R := \{1, \ldots, G\} \setminus A \tag{24}$$

are treated as passive (or remaining) part of the circuit. The decomposition of the unitary operator into gates and the respective subset of active gates can be chosen at will for the use case of interest.

The corresponding coalition game involves $N = |A|$ players. Coalitions in this game represent sets of active gates that can be associated with circuits consisting of the gates of the coalition and the passive gates in R. For a given coalition S, such a circuit is defined by the unitary operator

$$U(S, R, \vec{x}, \vec{\theta}) := \prod_{g \in \{A_a \,|\, a \in S\} \cup R} U_g(\vec{x}^g, \vec{\theta}^g) \tag{25}$$

that yields a quantum state

$$|\Psi(S, R, \vec{x}, \vec{\theta})\rangle := U(S, R, \vec{x}, \vec{\theta})\,|0\rangle \tag{26}$$

in analogy to Eq. (16). The product in Eq. (25) is to be understood as ordered in the same way as in Eq. (22). Measurement results for q qubits are consequently given by

$$\vec{B}(S, R, \vec{x}, \vec{\theta}) \equiv \vec{B}(S) := \{\vec{b}_1(S), \ldots, \vec{b}_s(S)\}, \tag{27}$$

in analogy to Eq. (17) with $\vec{b}_i(S) \in \{0, 1\}^q$ and

$$\vec{b}_i(S) \sim \mathbb{P}_{S,R,\vec{x},\vec{\theta}}(\vec{b}) := \langle \Psi(S, R, \vec{x}, \vec{\theta}) | M(\vec{b}) | \Psi(S, R, \vec{x}, \vec{\theta}) \rangle \tag{28}$$

for $i \in \{1, \ldots, s\}$ in analogy to Eq. (18). Hence, Eq. (19) can be written as

$$S, R, \vec{x}, \vec{\theta} \mapsto \vec{B}(S), \tag{29}$$

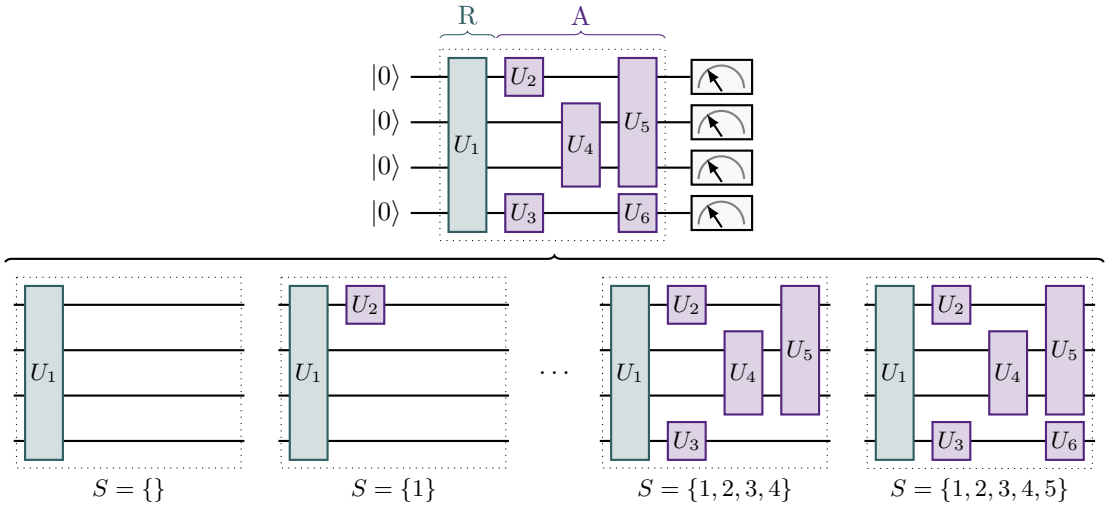


Figure 3: Schematic representation of the set of circuits that are involved in a coalition game for SVQXs. Exemplarily, we consider the circuit from Fig. 2 as the circuit of interest with an active gate set $A = \{A_1 = 2, A_2 = 3, A_3 = 4, A_4 = 5, A_5 = 6\}$, Eq. (23), and a set of remaining gates $R = \{1\}$, Eq. (24). The original circuit of interest is shown on top, it contains the six gates $U_1, \ldots, U_6$, where we omit all arguments. The index of each gate indicates the corresponding gate index $g \in \{1, \ldots, G\}$ with $G = 6$ in Eq. (22). Each coalition S can be associated with a circuit that contains the gates whose indices are included in the set $\{A_a \,|\, a \in S\} \cup R$ as defined in Eq. (25). Four exemplary coalitions and their corresponding circuits are shown at the bottom. The leftmost and rightmost circuits are associated with the empty and grand coalition, respectively.

which also depends on the active gates (players) in S and the passive gates in R. In Fig. 3, we exemplarily outline the set of circuits that are involved in such a coalition game.

Technically, SVQXs are computed in the same way as conventional SVs via Eq. (8). Their premise is a coalitional game based on gates within a quantum circuit as players. For such a coalition game, the value function, Eq. (7), takes the form

$$v(S) = v(S; R, \vec{x}, \vec{\theta}, \vec{B}(S)). \tag{30}$$

More generally, the value function can in analogy to Eq. (14) depend on multiple feature vectors $\vec{x}_i \in \mathcal{X}$ and variational parameters $\vec{\theta}_i \in \mathcal{P}$. Measurement results are then obtained for each $\vec{B}(S, R, \vec{x}_i, \vec{\theta}_i)$, Eq. (27), for all $i \in \{1, \ldots, d\}$, hence

$$v(S) = v(S; R, \Xi_d). \tag{31}$$

Here, we made use of the abbreviation

$$\Xi_d := \{(\vec{x}_1, \vec{\theta}_1, \vec{B}(S, R, \vec{x}_1, \vec{\theta}_1)), \cdots, (\vec{x}_d, \vec{\theta}_d, \vec{B}(S, R, \vec{x}_d, \vec{\theta}_d))\}. \tag{32}$$

For $d = 1$, Eq. (31) reduces to Eq. (30). A value function may also depend on additional meta data.

In the most general case of value functions, the measurement results, Eq. (27), may refer not just to direct measurements of the circuit of interest, but instead to an alternation or modification of this circuit in the sense that the unitary from Eq. (25) is mapped onto another unitary, i.e., $U(S, R, \vec{x}, \vec{\theta}) \to U'(S, R, \vec{x}, \vec{\theta})$. This allows to realize experiments like a SWAP test [70, 71]. The results may also depend on a particular order, for example to realize an iterative quantum state tomography algorithm [72]. In the next section, we discuss examples for value functions, which also include such special cases.

In addition to executing circuits on a QPU, we also consider the case where a circuit is simulated on classical hardware. Such an approach allows to perform quantum computations that are not yet feasible on NISQ devices due to hardware-related imperfections like dissipation and decoherence [73]. There are two major differences between simulating a circuit classically and running it on an actual QPU. Firstly, the simulation is noise-free (but noise can optionally be modeled [74]). And secondly, the state $|\Psi(S, R, \vec{x}, \vec{\theta})\rangle$ from Eq. (26) can be determined directly without the need for a measurement.

In terms of the presented formalism, a simulation corresponds to the case of infinite shots, i.e., $s \to \infty$ in Eq. (17) such that the probability distribution $\mathbb{P}_{S,R,\vec{x},\vec{\theta}}$, Eq. (28), can be obtained with arbitrary precision for all bit strings $\vec{b} \in \{0,1\}^q$. Alternatively, a value function for simulated circuits may also directly depend on the quantum state instead of the measurement results. An exemplary simulation-based value function is provided in the following section as well.

3.2 Value functions for SVQXs

Value functions for SVQXs, Eq. (31), can be chosen in many different ways. First of all, value functions from classical XAI can be used in the same way for QML tasks. In analogy to Eqs. (13) and (14), we can define

$$v(S) = v(S; \vec{x}, h(\vec{B}(S, R, \vec{x}, \vec{\theta})), \tag{33}$$

for single data points $\vec{x} \in \mathbb{R}^N$ and

$$v(S) = v(S; \vec{x}_1, \ldots, \vec{x}_d, h(\vec{B}(S, R, \vec{x}_1, \vec{\theta}_1), \ldots, h(\vec{B}(S, R, \vec{x}_d, \vec{\theta}_d)) \tag{34}$$

for a data set $\vec{x}_1, \ldots, \vec{x}_d$ consisting of d data points $\vec{x}_i \in \mathbb{R}^N$ for $i \in \{1, \ldots, d\}$. Here, we recalled h from Eq. (20). For example, a loss function of the form of Eq. (21) can be used as the value function in Eq. (34), that is,

$$v(S) = \ell(\{(\vec{x}_1, h(\vec{B}(S, R, \vec{x}_1, \vec{\theta}_1))), \ldots, (\vec{x}_d, h(\vec{B}(S, R, \vec{x}_d, \vec{\theta}_d)))\}, \vartheta). \tag{35}$$

SVQXs based on such a value function allow to assign a contribution of each gate to the loss.

Moreover, more specialized value functions can be used independent of a particular ML task to evaluate certain properties of a quantum circuit. In this context, we specifically propose four typical use cases in form of questions:

1. How uniformly is the unitary space explored by an ansatz circuit?
2. What ability has an ansatz circuit to create entangled states?
3. How does a circuit evaluation perform on NISQ hardware in comparison with an idealized classical simulator?
4. How efficiently can a circuit be executed on a specific NISQ hardware device?

In the following, be briefly outline potential value functions to answer these questions.

For the first use case, a suitable metric to quantify the uniform exploration capability of the unitary space is given by the *expressibility* [75]. It corresponds to the distance between the distribution of unitaries generated by the ansatz with randomly chosen variational parameters and the maximally expressive uniform distribution of unitaries. We consider an ansatz circuit with a corresponding unitary, Eq. (16), that depends on a parameter vector $\vec{\theta}$, whereas the feature vector $\vec{x}$ is presumed to be constant. An estimator for the expressibility of this circuit $\eta(S, R, \vec{x}, c_p, c_b, \mathcal{F})$ is presented in App. A. Here, c_p and c_b denote parameters that determine the precision of the estimator, whereas $\mathcal{F}$ contains measurement results from SWAP tests. The corresponding value function

$$v(S) = -\eta(S, R, \vec{x}, c_p, c_b, \mathcal{F}) \leq 0 \tag{36}$$

allows to quantify the contribution of each gate to the expressibility, i.e., its exploration capability of the unitary space. This quantity is in particular independent of a specific parameter vector $\vec{\theta}$. The negative sign in Eq. (36) ensures that larger values indicate a higher expressibility, whereas smaller values indicate a lower expressibility.

To answer the question of the second use case, the *entangling capability* represents a suitable metric [75]. As we present in App. B, it is based on the mean Meyer-Wallach entanglement measure [76] for randomly drawn variational parameters. The metric can be directly used as a value function

$$v(S) = \lambda(S, R, \vec{x}, c_s, \mathcal{T}) \in [0, 1] \tag{37}$$

that allows to quantify the contribution of each gate to the overall entangling capability of the circuit, where larger values indicate more entangling capability. This quantity is in particular independent of

a specific parameter vector $\vec{\theta}$. Here, c_s determines the number of drawn parameter samples, whereas $\mathcal{T}$ represents a list of measurement results for quantum state tomography.

For the third use case, we evaluate the circuit of interest for a given coalition, Eq. (25), both on an actual QPU as well as with a (noise-free) simulator running on a classical host. As briefly outlined in App. C, this approach allows us to use the *Hellinger fidelity* [77] to quantify the degree of agreement between the measurement results from the quantum device with the results from the classical simulator based on the respective distributions of bit strings. Two effects are responsible for a potential deviation between these two results. First, the limited number of shots on the QPU and second, hardware-related uncertainty from the NISQ device that is not present on the idealized simulator. Since a lower degree of agreement consequently results from deficiencies (or inadequacies) of the QPU, the Hellinger fidelity is a measure of the severity of these deficiencies. The Hellinger fidelity can be directly used as a value function

$$v(S) = H(S, R, \vec{x}, \vec{\theta}, \vec{B}(S)) \in [0, 1] \tag{38}$$

that allows to quantify the contribution of each gate to the quantum hardware deficiencies. By definition, smaller values indicate larger deficiencies. Here, $\vec{B}(S)$ denotes the acquired measurement results, Eq. (27). In particular, the evaluation of this value function requires both a simulation of the circuit as well as its execution on a QPU.

The fourth use case considers the practical challenge of efficiently executing an arbitrary quantum circuit on a NISQ device. As explained in App. D, any quantum circuit must be subjected to a suitable equivalence transformation to allow execution on a specific hardware device, a process also known as *transpilation* [78]. The transpilation procedure transforms a circuit into a form, where it only contains gates from the set $\mathcal{U}$ of available gates as specified by the hardware device of interest. In this sense, the transpilation can be understood as a necessary *classical* preprocessing procedure for all QPU evaluations. However, such a procedure is ambiguous by definition and can therefore lead to different gate decompositions with a varying number of gates. In general, a lower number of gates is preferable since each gate contributes to the noise of the corresponding QPU computation. As briefly outlined in App. D, the estimated execution efficiency $\tau(S, R, \mathcal{U}, s_1, s_2)$ represents a metric that measures how efficiently a circuit can be executed on a specific hardware device by assigning a negative penalty value to each gate based on the most favorable transpilation of the circuit. The penalty value is determined by the penalty parameters $s_1 < 0$ and $s_2 < s_1$ for one-qubit and two-qubit gates, respectively. The corresponding value function

$$v(S) = \tau(S, R, \mathcal{U}, s_1, s_2) \leq 0 \tag{39}$$

can therefore be used to quantify the contribution of each gate to the estimated efficiency of executing the circuit on a specific hardware device, where larger values indicate a higher estimated efficiency.

In contrast to use case three, only the efficiency of the transpilation process and not the actual hardware errors that arise during execution are considered. Therefore, the premise of the final use case is different in comparison with the other three: no QPU is used to evaluate the value function, instead the transpilation is run on a CPU to determine the estimated efficiency based on the properties of a specific hardware device based on the chosen penalty function.

3.3 Summary

In conclusion, SVQXs can be defined in analogy to conventional SVs via Eq. (8) based on a coalition game in which certain (parameterized or non-parameterized) gates of a quantum circuit constitute the players, Eq. (23). The respective value functions, Eq. (30), involve the unitary operators each coalition, Eq. (25), with possible use of results from a QPU or a classical host or both. For the sake of convenience, we use the notation

$$\Phi_{(g)} := \Phi_i \text{ with } A_i = g \tag{40}$$

to refer to the SVQX of an active gate with gate index g, Eq. (22), where $A_i \in A$, Eq. (23). Examples for value functions based on typical use cases are presented in Eqs. (35) to (39). Value functions based on QPU calculations may suffer from intrinsic noise and are consequently described by SVs of uncertain value functions, Eq. (8), which can be evaluated, e.g., via Eq. (10) or Eq. (11).

4 Experiments

In this section, we demonstrate the use of SVQXs to evaluate (variational) circuits. For this purpose, we perform multiple experiments on classical simulators and actual IBM quantum hardware. In total, we consider five use cases for which we recall a selection of the proposed value functions from Sec. 3.2. Three use cases are from the QML domain as a first step towards XQML: two classifiers in Secs. 4.1 and 4.2, respectively, and a generative model in Sec. 4.3. Finally, the last two use cases consider circuit transpilation as a classical preprocessing task in Sec. 4.4 and solving a combinatorial optimization problem with a variational quantum eigensolver (VQE) in Sec. 4.5, which allows us to demonstrate the usability of SVQXs beyond XQML. The goal of this section is to present different application areas for SVQXs without going too deep into possible interpretations or even conclusive explanations of the results.

The experiments have been realized with Python using Qiskit [78] and a SVQX toolbox [79]. All data used in the experiments is publicly available online [80, 81].

We use a selection of (simulated and NISQ) QPUs to perform quantum computations. Specifically, as simulators we use (i) an idealized statevector simulator (`state_CPU`), which allows a deterministic evaluation of all amplitudes of the final state and (ii) an idealized shot simulator (`shot_CPU`), which samples a finite number of measurements (shots) from the amplitudes of the final state. Both simulators are implemented in Qiskit and run on a classical host. In contrast to the `state_CPU` simulator, the `shot_CPU` simulator has a non-deterministic behavior due to the shot noise. As NISQ devices, we use the `ibmq_ehningen` and `ibm_oslo` QPUs. Both of these QPUs are superconducting quantum hardware devices provided by IBM Quantum through cloud services [82, 83]. Detailed specifications can be found in App. E.

For the estimation of SVQXs, we use either Eq. (10) or Eq. (11). The first approach depends on the chosen number of repetitions K and the second on both K and the number of samples n. For practical purposes, we specify K and a sampling fraction $\alpha \in (0, 1]$ to fully define our estimation method. A choice of $\alpha < 1$ indicates that we use Eq. (11) with a number of samples

$$n \equiv n(\alpha) := \lceil \alpha 2^{N-1} \rceil, \tag{41}$$

whereas $\alpha = 1$ indicates that we use Eq. (10).

4.1 Quantum support vector machine

As our first use case, we consider a quantum support vector machine (QSVM) [84] with the task of solving a supervised classification problem on a toy data set with two-dimensional features. This QML model is based on a classical support vector machine (SVM) [85] that operates on a kernel matrix from a QPU. That is, the elements of the kernel matrix are estimated from the transition amplitude by measuring the number of all-zero strings of a composition of two consecutive feature map circuits, each parameterized by one of the two feature vectors of interest [84]. Consequently, both training the QVSM and making predictions with it requires the evaluation of quantum circuits. As feature map circuit, we use a two-qubit second-order Pauli-Z evolution circuit with $7r$ gates, where r denotes the number of repetitions as shown in Fig. 4. We particularly consider $r \in \{1, 2, 3\}$. For the parameterization of the circuit (with two-dimensional feature vectors $\vec{x} \in \mathbb{R}^2$), we use the abbreviations

$$\varphi_i(x_i) := 2x_i \tag{42}$$

for $i \in \{1, 2\}$ and

$$\varphi(\vec{x}) := 2(\pi - x_1)(\pi - x_2), \tag{43}$$

respectively. For a recent review of QSVMs, we refer to [86] and references therein.

For training and testing purposes, we use the artificial data set from [84] with two-dimensional features $\vec{x} \in (0, 2\pi]^2 \subset \mathbb{R}^2$ and binary class labels $y \in \{0, 1\}$. By construction, this data can in principle be fully separated by the chosen feature map with $r = 2$. We choose a separation gap of 0.3, which is a hyperparameter that determines the geometrical shape of the data. In total, we uniformly draw 40 training points and 1000 test points from the feature space, both with an equal distribution of zero and one class labels, as shown in Fig. 5.

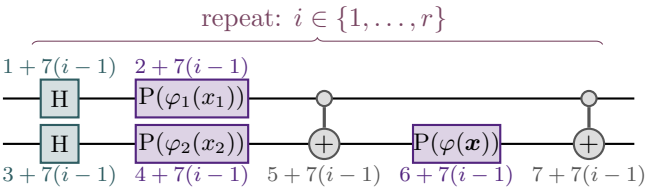


Figure 4: Feature map circuit for the QSVM: two-qubit second-order Pauli-Z evolution circuit with r repetitions parameterized by feature vectors $\vec{x} \in \mathbb{R}^2$. We use the circuit symbols from App. F and the abbreviations from Eqs. (42) and (43). The number near each gate represents its gate index g, Eq. (22). Here and in the following, we omit initialization and measurements in circuit sketches to simplify our presentation. These operations are by default always realized as shown in Fig. 3.

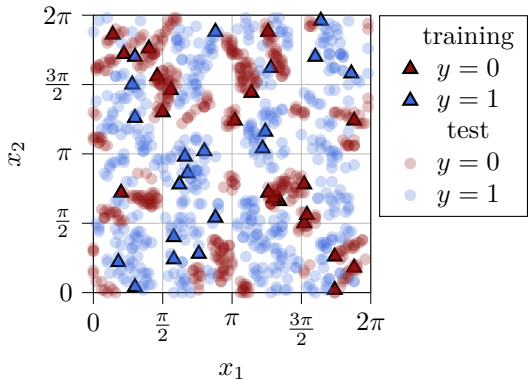


Figure 5: Artificial test and training data sets for the supervised classification problem with two-dimensional features $\vec{x} \in (0, 2\pi]^2$ and binary labels $y \in \{0, 1\}$. We use a QSVM to solve this problem.

For this use case, all gates are active gates such that $A = \mathfrak{S}$ and $R = \varnothing$ as defined in Eqs. (23) and (24). Furthermore, $\Phi_{(g)} = \Phi_i$ according to Eq. (40). The considered value function $v(S)$ is defined by the following procedure:

1. Evaluate the kernel matrix for the training data set based on the feature map circuit, Fig. 4, that contains only gates from S, Eq. (25).
2. Train the QSVM using this kernel matrix. The training algorithm is deterministic.
3. Evaluate the accuracy of the trained model on the test data set to determine $v(S) = \text{acc}$ using the same feature map circuit as before.

The accuracy is defined in the usual sense as

$$\text{acc} := \frac{\text{number of correctly predicted test labels}}{\text{total number of test points}} \in [0, 1] \tag{44}$$

and is therefore of the form of Eq. (35). This value function allows to quantify the contribution of each gate to the kernel matrix for a successful training of the QSVM, where larger values indicate a better model performance.

We run all evaluations on the `state_CPU` simulator. Since the training algorithm of the QSVM is also deterministic, no randomness is involved in the evaluation of the value function with the consequence that $\hat{\Phi}_i^1 = \Phi_i = \phi_i$ according to Eqs. (2), (8) and (10). For $r \in \{1, 2\}$, we therefore use $K = 1$ and $\alpha = 1$, Eq. (41). On the other hand, we choose $K = 1$ and $\alpha = 0.01$ for $r = 3$ to reduce the computational effort, Eq. (11), and perform two independent runs, for each of which we draw different samples. The resulting SVQXs for the feature map circuit, Fig. 4, with different numbers of repetitions r are presented in Fig. 6. The resulting test accuracies read $\text{acc} \approx 0.842$ for $r = 1$, $\text{acc} \approx 0.986$ for $r = 2$, and $\text{acc} \approx 0.913$ for $r = 3$.

We find that the resulting SVQXs exhibit a rich structure. First of all, we can observe that the SVQXs for the last CX gate of a feature map circuit vanish since it has no effect on the measured

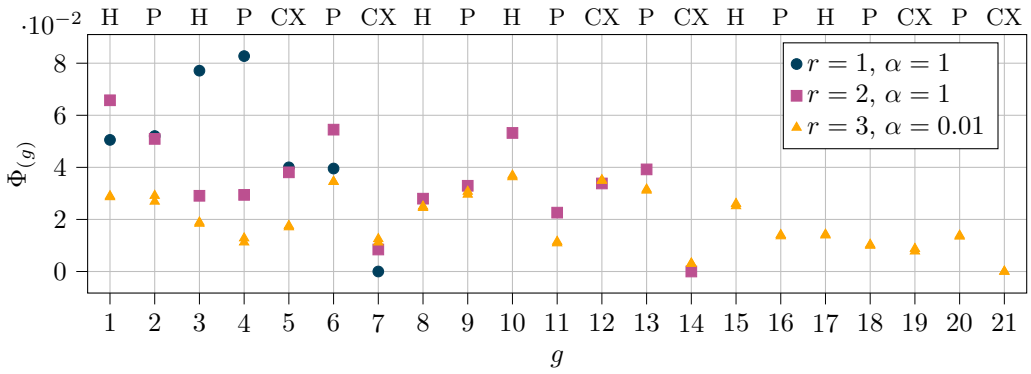


Figure 6: SVQXs $\Phi_{(g)}$, Eq. (40), for the feature map circuit of a QSVM, Fig. 4, with a value function corresponding to the retrained test accuracy of the corresponding model. A larger SVQX is consequently more favorable. We consider three different numbers of repetitions $r \in \{1, 2, 3\}$. The gate index g, Eq. (22), is shown on the bottom, whereas the corresponding gate name is shown on the top of the plot. All evaluations are run on the `state_CPU` simulator, which leads to deterministic value functions. We therefore use $K = 1$ repetitions for their evaluation. For $r \in \{1, 2\}$, we decide not to sample value functions by choosing a sample fraction $\alpha = 1$, Eq. (41), which leads to deterministic SVQXs via Eq. (10). For $r = 3$, we choose a value function sample with $\alpha = 0.01$ and perform two independent runs, for each of which we draw different samples using Eq. (11). We plot the results of both runs with the same symbols.

number of all-zero strings and therefore on the value function. Intermediate CX gates, on the other hand, have non-vanishing SVQXs. Overall, the CX gates at the end of a repetition (i.e., $g \in \{7, 14, 21\}$) have small SVQXs. Supplementary, we show the corresponding distributions of marginal contributions for the evaluation of the SVQXs from Fig. 6 in App. G.

Another perspective on SVQXs is to analyze the corresponding value function distributions for subsets of gates. For this purpose, we consider the multiset of value functions

$$\mathcal{W}_k := \{v \,|\, v = v(S) \in \mathbb{R} \,\forall\, S \subseteq \mathfrak{S} \wedge |S| = k\} \tag{45}$$

given k (active) gates. That is, every element in $\mathcal{W}_k$ corresponds to a retrained test accuracy for a QSVM with a specific feature map circuit (consisting of k gates in total). To determine $\mathcal{W}_k$, all value functions have to be evaluated, which corresponds to $\alpha = 1$. In case of $\alpha < 1$, we consider the multiset of sampled value functions $\widehat{\mathcal{W}}_k$ instead. We plot the distribution of $\mathcal{W}_k$ and $\widehat{\mathcal{W}}_k$, respectively, for the results from Fig. 6 as box plots in Fig. 7.

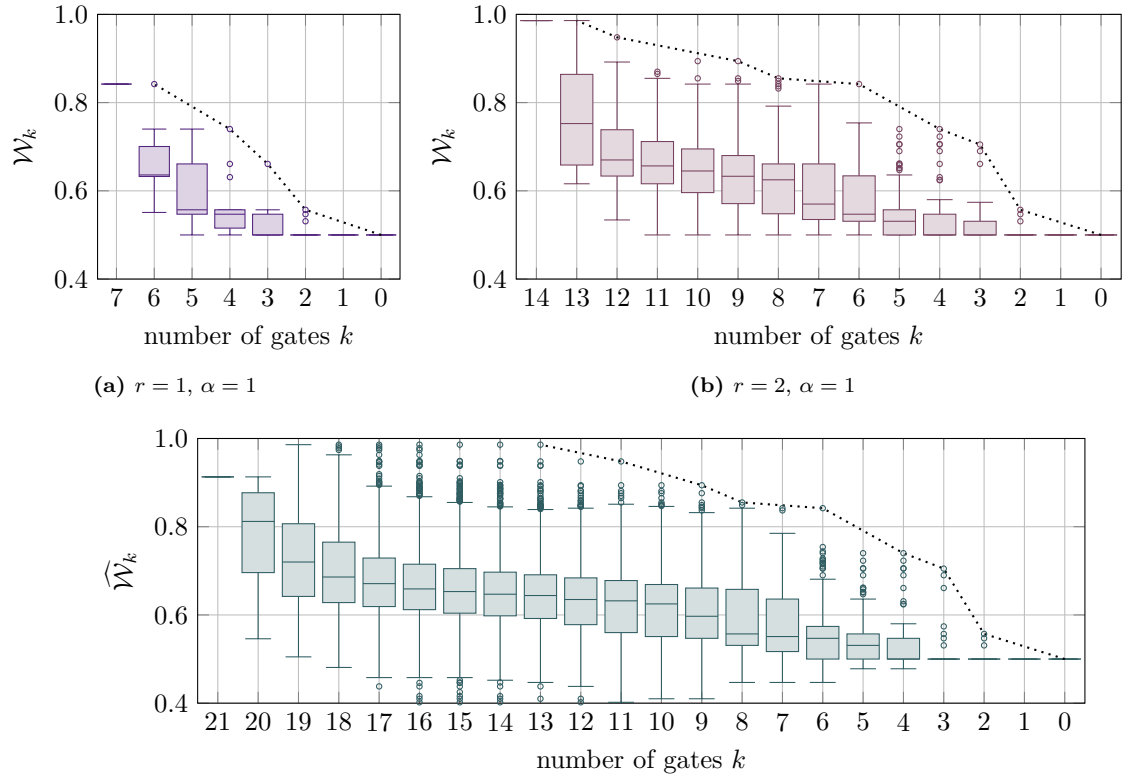
The plots confirm that one gate (the last CX) can be removed without lowering the test accuracy. Furthermore, for $r = 3$ removing up to eight gates can even lead to an improvement of the score. The trade-off between the number of gates and the test accuracy can be considered as a multi-criteria optimization problem for circuit pruning [87]. The corresponding Pareto frontier is indicated as a dotted line in Fig. 7. This analysis can be understood as a very simple form of QAS that allows to select the optimal gate configuration for a certain task.

So far, SVQXs have allowed us to gain certain insights into the influence of the gates of a feature map circuit on the test accuracy of a corresponding QSVM. In order to investigate the significance of these insights and the general robustness of SVQXs, we conduct three additional tests in the following.

First, we determine the influence of the sampling fraction α, Eq. (41), that determines the number of sampled value functions. For this purpose, we perform three evaluations of the SVQXs from Fig. 6 using different values of α and $K = 1$ for each of the repetitions $r \in \{2, 3\}$. Since the evaluations are performed on a `state_CPU` simulator, the only source of randomness originates from the sampling of value functions. The results are presented in Fig. 8.

We find that the mean values of the SVQXs converge for increased values of α, whereas their standard deviation decreases, as expected. The standard deviation is in particular different for each gate.

As a second experiment, we test the generality and robustness of the result by evaluating the SVQXs from Fig. 6 for different data sets. Specifically, we consider the previously used training and test data sets and, in addition, four other training and test data sets of the same size that have been sampled from the same data distribution with different random seeds. We perform the evaluations



(a) $r = 1$, $\alpha = 1$

(b) $r = 2$, $\alpha = 1$

(c) $r = 3$, $\alpha = 0.01$

Figure 7: Distribution of value functions $\mathcal{W}_k$, Eq. (45), for different number of gates k in form of box plots. The results are obtained for the experiments from Fig. 6. We show three different numbers of repetitions $r \in 1, 2, 3$. All evaluations are run on a `state_CPU` simulator with $K = 1$. The dotted line indicates the Pareto frontier for a multi-criteria circuit pruning problem with the goal to minimize the number of gates and to maximize the test accuracy of the QSVM. For $r = 3$, we consider the distribution of sampled value functions $\widehat{\mathcal{W}}_k$ for one run. The corresponding Pareto frontier is an approximation since not all value functions have been evaluated.

on a `state_CPU` simulator with $K = 1$ and $\alpha = 1$, which eliminates all randomness. The results are presented in Fig. 9.

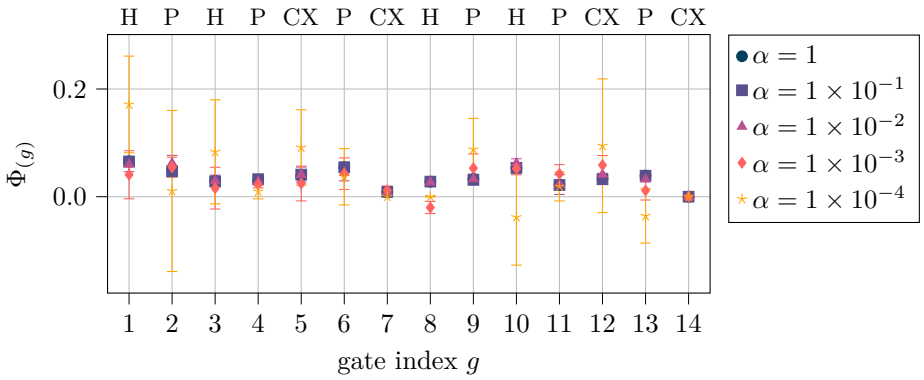
From these results it becomes apparent that the SVQXs for some gates are more susceptible to a change of the data than others. In particular, for $g = 4$, $g = 11$, and $g = 13$ we find a large standard deviation. This is no surprise since all of these gates are P gates that are directly influenced by the choice of data. Moreover, the initial H gates with $g = 1$ and $g = 3$ show a similarly large standard deviation, indicating that their contribution is also data-dependent.

Finally, in a third experiment we investigate the influence of noise from a NISQ QPU. To this end, we choose $r = 1$ and perform six evaluations on the `ibmq_ehningen` QPU with $K = 1$ and $\alpha = 1$. Therefore, the only source of randomness are errors from the QPU. These errors (unavoidably) lead to uncertain value functions in the sense of Eq. (7). For comparison, we also perform an evaluation on the `state_CPU` simulator. The results are presented in Fig. 10.

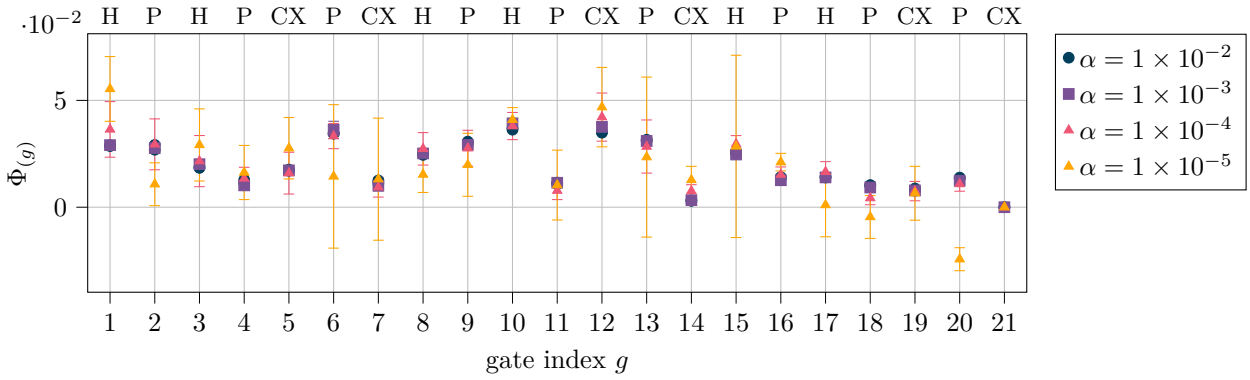
Clearly, the results show that the influence of the errors from the NISQ QPU on the SVQXs is not significant.

4.2 Quantum neural network

As our second use case, we consider a quantum neural network (QNN) with the task to perform a binary classification of a toy data set with two-dimensional features. The data set contains 20 data points and is visualized in Fig. 11. By construction, it is linearly separable in the feature space. For reasons of simplicity, we use the same data set for test and training purposes. The chosen quantum circuit of the QNN is shown in Fig. 12. It is parameterized both by feature vectors $\vec{x} \in \mathbb{R}^2$ and trainable



(a) $r = 2$



(b) $r = 3$

Figure 8: Influence of the number of sampled value functions that is determined by the sampling fraction α, Eq. (41), on the SVQXs from Fig. 6 for two repetitions $r \in \{2, 3\}$. For $r = 2$ and $\alpha = 1$, no randomness is involved so that we can show the exact results. For $r = 3$ and $\alpha = 1 \times 10^{-2}$, we plot both results of two independent runs for each case with the same symbols. In all other cases, we show the mean and one standard deviation over three independent runs. The evaluations are performed on a `state_CPU` simulator with $K = 1$.

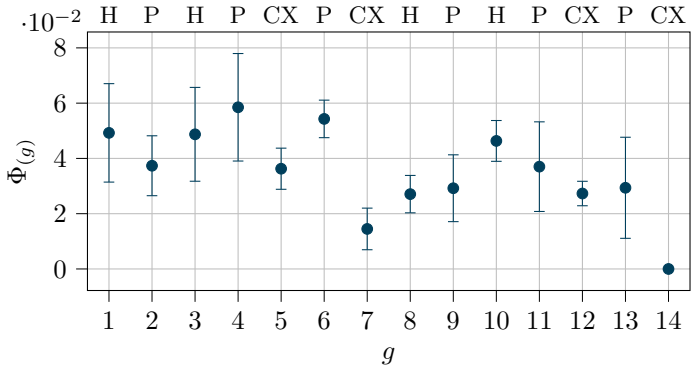


Figure 9: Influence of the data sampling on the resulting SVQXs. We plot SVQXs for the QSVM with $r = 2$ in analogy to Fig. 6 based on five different test and training data sets (obtained from sampling the data distribution with different random seeds). We show the mean and one standard deviation. The evaluations are performed on a `state_CPU` simulator with $K = 1$ and $\alpha = 1$.

parameters $\vec{\theta} \in \mathbb{R}^4$. The prediction of the class label is determined by the binary measurement result of the first qubit.

We train the QNN once on the proposed data set with a COBYLA optimizer [88] to determine a fixed value for the parameters $\vec{\theta} = \vec{\theta}_{\text{trained}} \approx (3.860, -1.070, -1.583, 0.860)$. The resulting accuracy of the classifier, Eq. (44), is 80%. For the evaluation of SVQXs, we choose the active gate set

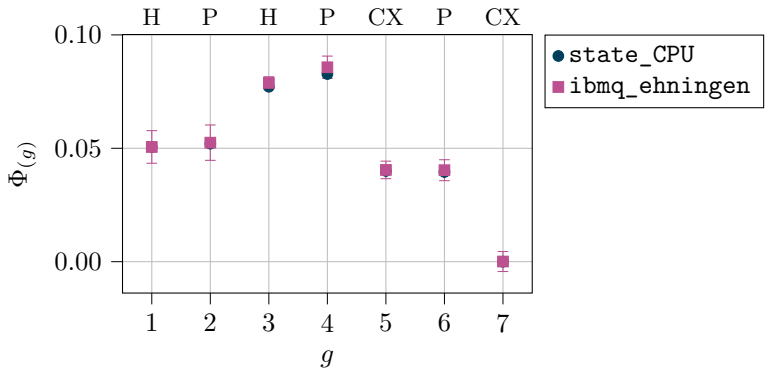


Figure 10: Influence of uncertainty from NISQ hardware on the resulting SVQXs. We plot SVQXs for the QSVM with $r = 1$ in analogy to Fig. 6. All results are evaluated either on a `state_CPU` simulator or the `ibmq_ehningen` QPU with $K = 1$ and $\alpha = 1$. For `ibmq_ehningen`, we show the mean value and five standard deviations over six independent runs.

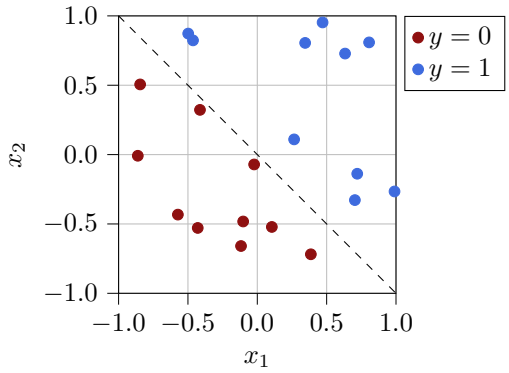


Figure 11: Toy data for the supervised classification problem consisting of 20 data points with two-dimensional features $\vec{x} \in [-1, 1]^2$ and binary labels $y \in \{0, 1\}$. We use a QNN to solve this problem.

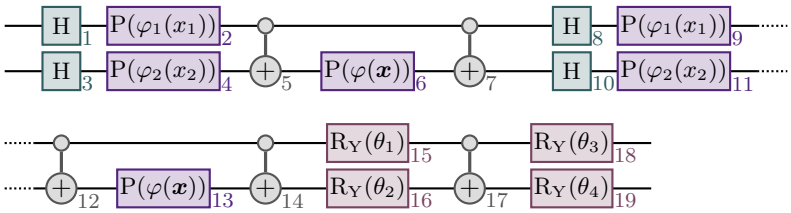


Figure 12: Circuit for the QNN parameterized by trainable parameters $\vec{\theta} \in \mathbb{R}^4$ and feature vectors $\vec{x} \in \mathbb{R}^2$. We use the circuit symbols from App. F and the abbreviations from Eqs. (42) and (43). The number near each gate represents its gate index g, Eq. (22).

$A = \{2, 4, 5, \ldots, 19\}$, Eq. (23), and a set of remaining gates $R = \{1, 3\}$, Eq. (24). Hence, all gates except for the initial H gates are treated as active. The considered value function $v(S)$ is defined by the one-shot test accuracy of the QNN circuit that contains only gates from $S \cup R$, Eq. (25). That is, we perform one shot for every data point to predict the class label (by parameterizing the QNN circuit with the current feature vector as $\vec{x}$ and the trained parameters $\vec{\theta}_{\text{trained}}$) and evaluate the accuracy via Eq. (44). Using the one-shot test accuracy allows us to investigate the shot noise further below.

We run all evaluations on the `shot_CPU` simulator. Therefore, the only source of randomness is the (simulated) shot noise. The shot noise leads to uncertain value functions in the sense of Eq. (7). The resulting SVQXs obtained for $K = 32$ and $\alpha = 0.01$, Eq. (41), are presented in Fig. 13.

We find that SVQXs show significant differences between certain gates. In particular, the gates with $g \in \{7, 8\}$ attain particularly large SVQXs, whereas the gates with $g \in \{6, 11, 14, 19\}$ attain particularly low values. Specifically, for $g = 19$ this is no surprise since the prediction of the QNN is

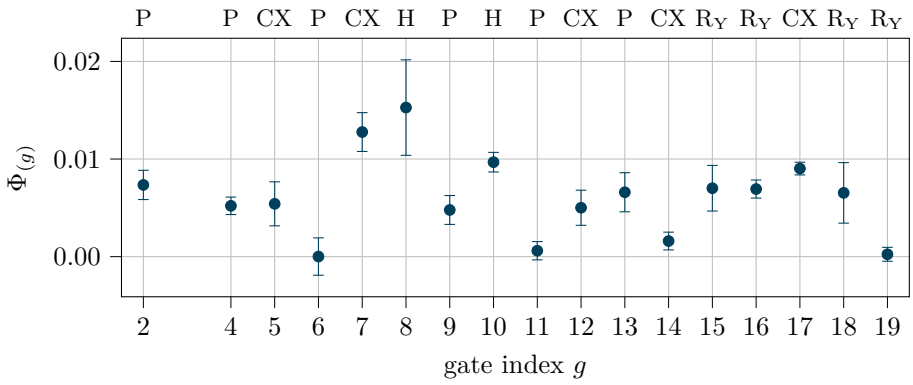


Figure 13: SVQXs $\Phi_{(g)}$, Eq. (40), for the feature map circuit of a QNN, Fig. 12, with a value function corresponding to the one-shot accuracy of the QNN without retraining. A larger SVQX is consequently more favorable. In analogy to Fig. 6, the gate index g, Eq. (22), is shown on the bottom, whereas the corresponding gate name is shown on the top of the plot. For the gates with $g \in R = \{1, 3\}$, no SVQX is determined. All evaluations are run on the `shot_CPU` simulator, which includes (simulated) shot noise. We use $K = 32$, $\alpha = 0.01$, Eq. (41), and show mean and one standard deviation over five independent runs.

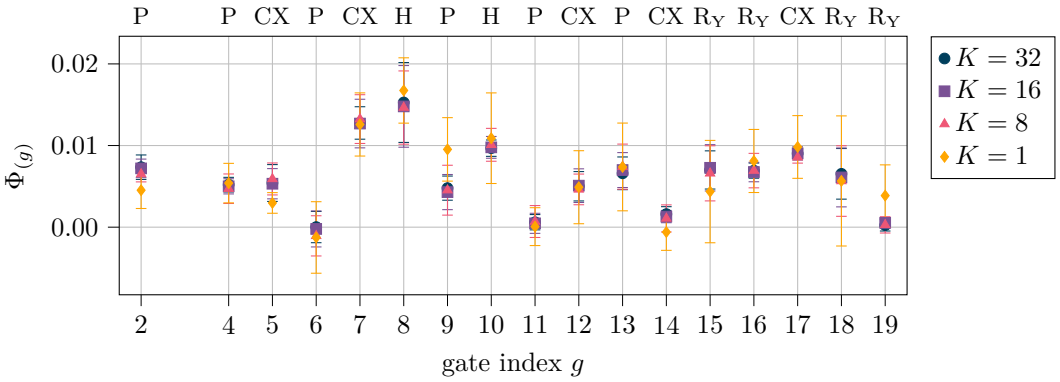


Figure 14: SVQXs $\Phi_{(g)}$, Eq. (40), for the feature map circuit of a QNN in analogy to Fig. 13 with the goal to compare different numbers of repetitions $K \in \{1, 8, 16, 32\}$ to investigate the influence of shot noise. We choose a sample fraction of $\alpha = 0.01$, Eq. (41). All evaluations are run on the `shot_CPU` simulator. Shown are mean and one standard deviation over five independent runs.

based on the measurement of the first qubit such that the last rotation gate on the second qubit has no contribution to the prediction, i.e., the accuracy of the model.

Our choice of value function and QPU allows us to investigate the shot noise of this experiment. For this purpose, we perform several additional runs with $K \in \{1, 8, 16, 32\}$ repetitions. The resulting SVQXs are shown in Fig. 14.

For $K = 1$ the value function is based on a single one-shot prediction for each data point, which leads to a large uncertainty. As expected, the shot noise becomes less significant for a higher number of repetitions K. The plot indicates that between $K = 16$ and $K = 32$, no major improvement of the results can be observed, which implies that $K = 16$ repetitions are already sufficient to capture the uncertainty of the shot noise in this experiment.

4.3 Quantum generative adversarial network

As third use case, we consider a quantum generative adversarial network (QGAN) with the task of generating a discretized log-normal distribution as described in [89]. For a recent review of QGANs we refer to [90]. We use the circuit shown in Fig. 15 as our generative model. It consists of three qubits and can therefore produce $2^3 = 8$ different amplitudes.

First of all, we train the QGAN by suitably optimizing its trainable parameters. As a result, we find $\vec{\theta} = \vec{\theta}_{\text{trained}} \approx (-1.328, -0.155, 3.025, 1.424, 0.427, 1.620, 0.980, 1.495, 1.461)$. The QGAN output

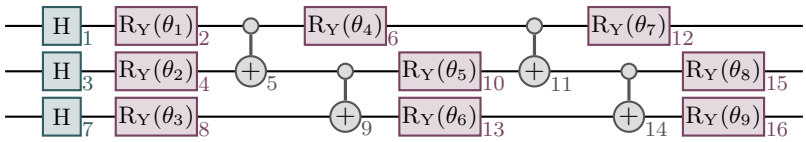


Figure 15: Circuit for the QGAN with trainable parameters $\vec{\theta} \in \mathbb{R}^9$. We use the circuit symbols from App. F and the abbreviations from Eqs. (42) and (43). The number near each gate represents its gate index g, Eq. (22).

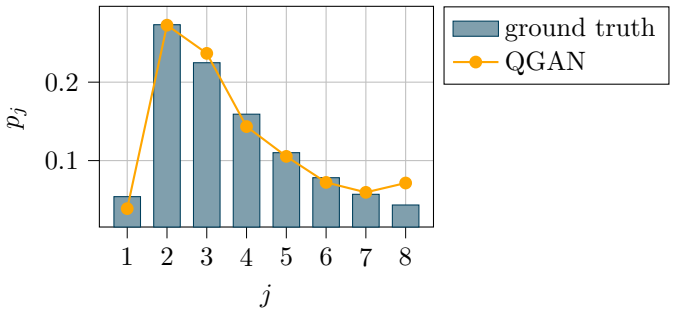


Figure 16: Output of the trained QGAN and corresponding ground truth for the discretized log-normal distribution p_j with a resolution of eight values j. The QGAN is evaluated on a `state_CPU` simulator.

in comparison with the ground truth is shown in Fig. 16. In the following, we only consider the trained QGAN.

As active gates for the evaluation of SVQXs, we choose all gates except for the initial three H gates $R = \{1, 3, 7\}$, Eq. (24), i.e., $A = \{2, 4, 5, 6, 8, 9, \ldots, 16\}$, Eq. (23). For the value function $v(S)$, we choose the Hellinger fidelity, Eq. (38), that results from a comparison of the QGAN output from the `state_CPU` simulator versus the QGAN output when evaluated on the NISQ QPU `ibm_oslo`. The value function is consequently uncertain in the sense of Eq. (7) and allows us to asses hardware errors. For `ibm_oslo`, we choose four different configurations. Specifically, we map the logical qubits to the physical qubits of the backend in two different ways. First, the three logical qubits of the QGAN circuit (from top to bottom in Fig. 15) are mapped to the physical qubits 0-1-2 of `ibm_oslo`.

Second, the physical qubits 3-5-4 are chosen. The two sets of physical qubits are also shown in Fig. 18a. For each set, we perform an evaluation with and without measurement error mitigation (MEM). The MEM method we use here requires a preprocessing step (independent of the QGAN circuit) in which the fraction of bit flips for each combination of measured ones and zeros is determined such that a linear transformation to any subsequent measurement result realizes the MEM [78, 91]. We perform three independent runs with $K = 3$ and $\alpha = 1$, Eq. (41). The results are shown in Fig. 17.

Comparing the two instances with and without MEM, we find that the chosen mitigation approach does not consistently increase the SVQXs but may in some cases also lead to a decrease. To further analyze our results, we require a frame of reference. For this purpose, we can use the error information about the hardware device that is provided by IBM [82]. For technical reasons, these errors may change over time and are therefore updated regularly [92], which is why we collect a set of errors over the duration of the experiments. Specifically, for each of the three runs from the four configurations shown in Fig. 17, we fetch the error information every hour during the total runtime (including waiting times and the execution of MEM) as summarized in App. H. The results are shown in Figs. 18b and 18c.

Clearly, the physical qubits 0, 1, and 2 exhibit a smaller mean (and standard deviation) of the single gate errors than 3, 4, and 5. Similarly, the CX errors of the physical qubit pairs 0-1, 1-2 and 3-5, show a considerably smaller mean (and standard deviation) than 5-4. From these results one would expect that the configuration 0-1-2 should lead to significantly better results than the configuration 3-5-4.

If we compare these observations to our results from Fig. 17, it becomes apparent that the ordering of the errors provided by IBM is not always reflected by the ordering of the corresponding SVQXs. This can be seen for example when comparing $g = 2$ with $g = 6$ and $g = 9$ with $g = 14$, respectively. This result might be a fact of the fundamental difference between the two error definitions: Our proposed error measure via SVQXs is context-aware in the sense that it is defined for specific gates

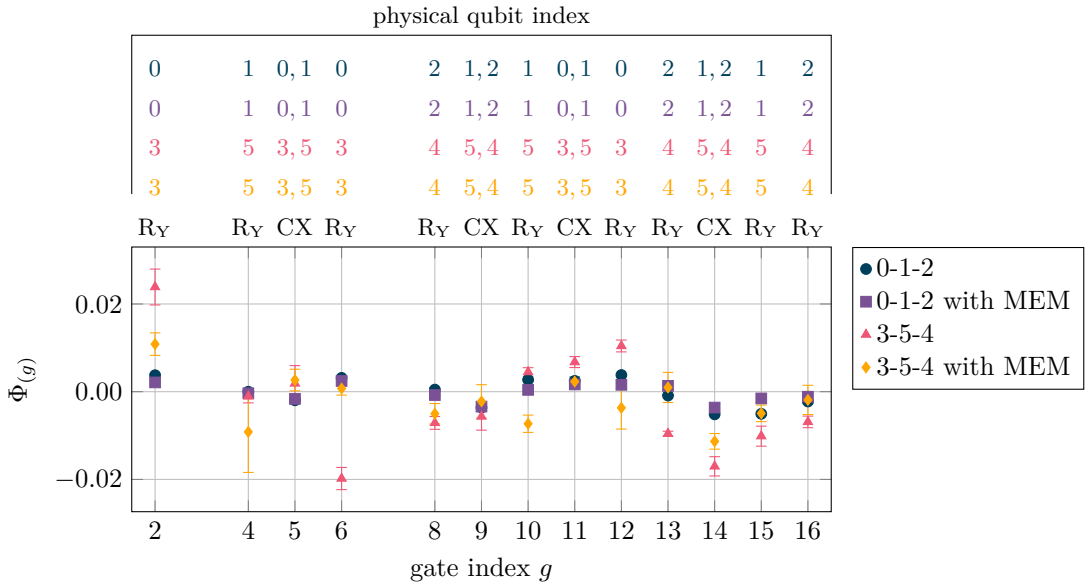
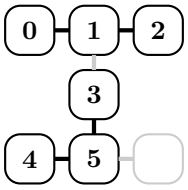
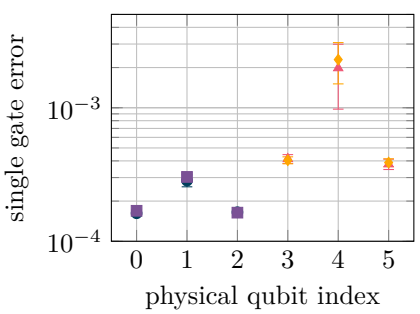


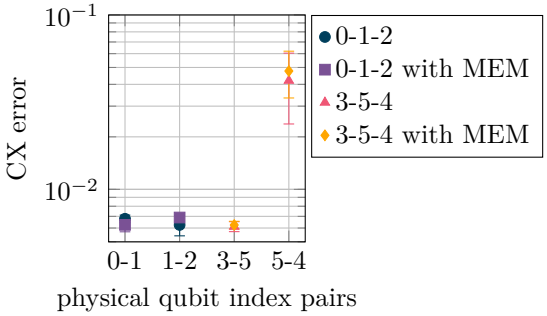
Figure 17: SVQXs $\Phi_{(g)}$, Eq. (40), for the QGAN circuit from Fig. 15 with respect to a value function that corresponds to the Hellinger fidelity, Eq. (38), of the QGAN output on the `state_CPU` simulator versus the QGAN output on `ibm_oslo` for different configurations. First, using the physical qubits 0-1-2 and second, using the physical qubits 3-5-4, each with and without MEM. We show the mean and one standard deviation over three independent runs for each configuration. A larger SVQX is more favorable. For all instances, we choose $K = 3$ and $\alpha = 1$. In analogy to the previous experiments, the gate index g, Eq. (22), is shown on the bottom, whereas the corresponding gate name is shown on the top of the plot. In addition, we also list the physical qubit indices over each gate to indicate on which physical qubits it acts. For the gates with $g \in R = \{1, 3, 7\}$, no SVQX is determined.



(a) Connectivity map of `ibm_oslo`, also see App. E.

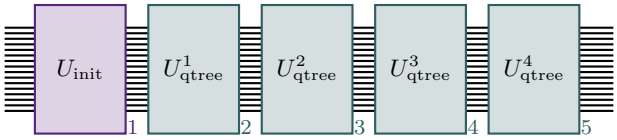


(b) Single gate errors for `ibm_oslo` as provided by IBM.

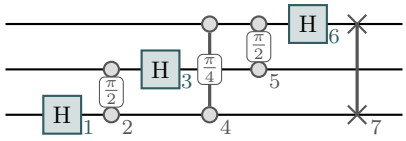


(c) CX errors for `ibm_oslo` as provided by IBM.

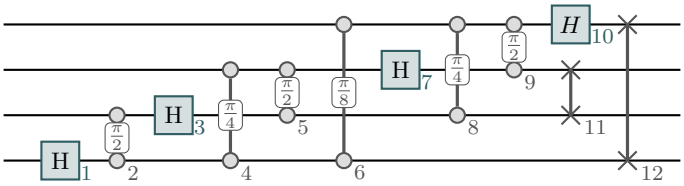
Figure 18: Gate errors as provided by IBM for `ibm_oslo` (plotted with logarithmic scale). In (a) we show the connectivity map of `ibm_oslo` with physical qubit indices and connections between qubits that allow the realization of joint CX gates, where we highlight the two sets of physical qubits (0-1-2 and 3-5-4) that are used for the results in Fig. 17. For each run from Fig. 17, we collect the respective errors that are provided by IBM [82]. Specifically, we fetch the error information for the hardware device every hour during the total runtime; also see App. H. The results are presented in (b) and (c), respectively, where we show the mean and one standard deviation over all error samples.



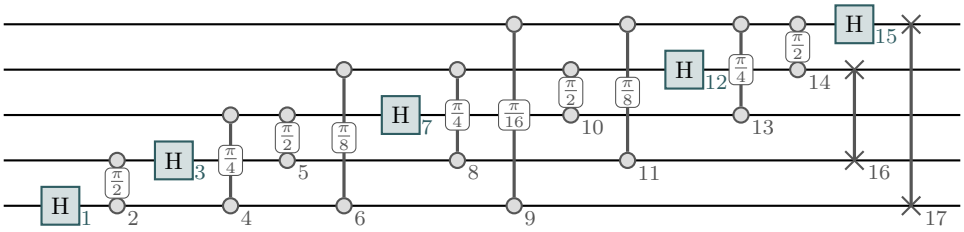
(a) QTree [94] containing four subsequent branches, each represented by a decision layer U^i_{qtree}. The circuit operates on 16 qubits. We choose $A = \{2, 3, 4, 5\}$ and $R = \{1\}$.



(b) QFT for three qubits. We choose $A = \{1, \ldots, 7\}$ and $R = \varnothing$.



(c) QFT for four qubits. We choose $A = \{1, \ldots, 12\}$ and $R = \varnothing$.



(d) QFT for five qubits. We choose $A = \{1, \ldots, 17\}$ and $R = \varnothing$.

Figure 19: Circuits for the purpose of evaluating the estimated execution efficiency. We specifically consider a QTree and three QFTs for a different number of qubits. We use the circuit symbols from App. F. The QTree circuit contains an initialization layer (U_{init}) and decision layers (U^i_{qtree} for $i \in \{1, \ldots, 4\}$). The number near each gate represents its gate index g, Eq. (22). For each circuit, we list the choice of active gates A, Eq. (23), and remaining gates R, Eq. (24).

within a specific circuit, whereas the errors provided by IBM can in contrast be considered as generic. A more detailed comparison is a possible starting point for further studies, which may also take into account recent error characterizations based on discrete time crystals that suggest that "benchmarks computed infrequently are not indicative of the real-world performance of a quantum processor" [93].

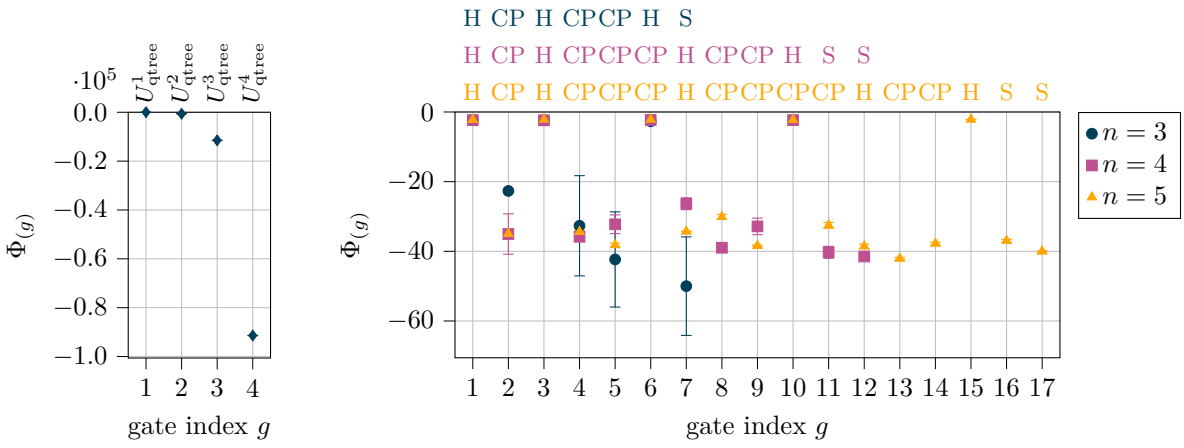
4.4 Transpilation

The fourth use case addresses circuit transpilation as a form of classical preprocessing. Specifically, we consider the transpilation of two types of circuits:

- A quantum decision tree classifier (QTree) with four decision branches trained on the binarized tic-tac-toe endgame data set [81] as described in [94]. The model requires 16 qubits. For the implementation, we use [95].
- Quantum Fourier transforms (QFTs) [96] for three to five qubits.

The respective circuits are shown in Fig. 19, where we also list the choice of active gates A, Eq. (23), and remaining gates R, Eq. (24), respectively. A QTree consists of an initialization layer and subsequent decision layers that represent the binary decisions within the tree model from the root to the branches.

As value function $v(S)$, we choose the transpilation-based estimated execution efficiency, Eq. (39), with penalty parameters $s_1 = -1$ and $s_2 = -10$. Its evaluation is performed exclusively on a classical host, where we repeat the transpilation process 50 times and take the best result. No measurements of the circuits take place for the evaluation. Transpilation is performed with respect to the



(a) QTree **(b)** QFT for n qubits

Figure 20: SVQXs $\Phi_{(g)}$, Eq. (40), for the QTree and QFT circuits, respectively, from Fig. 19 with a value function that corresponds to the transpilation-based estimated execution efficiency, Eq. (39), with respect to the `ibmq_ehningen` QPU. A larger SVQX is more favorable. In analogy to the previous experiments, the gate index g, Eq. (22), is shown on the bottom, whereas the corresponding gate (or layer) name is shown on the top of the plot. All evaluations are run on a classical host with $K = 1$. We choose $\alpha = 1$ for QTree and $\alpha = 0.01$ for QFT, respectively. In total, we perform three independent runs and show the mean value and one standard deviation (not visible in (a)).

`ibmq_ehningen` QPU using the Qiskit transpiler. Randomness is introduced by the heuristic transpilation algorithm that leads to an uncertain value function in the sense of Eq. (7). This is mitigated, to some extend, by our proposed approach of selecting the best transpilation result out of multiple repetitions. We choose $K = 1$ as well as $\alpha = 1$ for QTree and $\alpha = 0.01$ for QFT, Eq. (41). In total, we perform three independent runs for each QFT and the QTree circuit, respectively, with random sampling of value functions and random transpiler seeds. The resulting SVQXs are shown in Fig. 20.

The QTree results indicate, as expected, that deeper decision tree layers have smaller SVQXs since they involve an exponentially growing number of branches and therefore become increasingly more complex. Specifically, the plot suggests an approximately exponential decrease of SVQXs for deeper layers. For the QFT, we find that specific H and CP gates for $g \in \{1, 3, 6, 10, 15\}$ exhibit larger SVQXs. On the other hand, all remaining gates (which particularly includes all S gates) show smaller SVQXs. Interestingly, for $n = 3$ the SVQXs of the gates with $g \in \{4, 5, 7\}$ have a comparably large standard deviation, whereas the standard deviation of almost all other gates is negligible.

4.5 Variational quantum eigensolver

In this final use case, we empirically investigate the usage of SVQXs for a VQE to solve an optimization task that is not from the domain of QML. Specifically, we consider a max-cut problem for the exemplary graph shown in Fig. 21. This binary optimization problem has the goal of finding a bipartite subgraph of the original graph with as many edges as possible. It is NP-hard by definition and therefore no polynomial-time algorithms are known. We use the Quantum Approximate Optimization Algorithm (QAOA) to solve this problem on a QPU [97–99]. The respective circuit is shown in Fig. 22 and depends on the chosen depth r that defines the number of repetitions for cost layers (or cost unitaries) and mixing layers (or mixing unitaries). By design, each layer consists of multiple gates, but we do not consider the internal structure for this experiment.

As active gates for the evaluation of SVQXs, we choose all problem and mixing layers. Thus, the active gate set is formally given by $A = \{8 + 2(i-1) + k \,|\, i \in r \wedge k \in \{0, 1\}\}$, Eq. (23), whereas the set of remaining gates reads $R = \{1, \ldots, 7\}$, Eq. (24). We consider QAOA circuits of different depth $r \in \{1, \ldots, 7\}$. To evaluate SVQXs, we first solve the optimization problem with a COBYLA optimizer for each depth to obtain parameters $\vec{\theta} = \vec{\theta}_{\text{optimal}}$. To investigate the generality of the results, we perform three independent optimization runs in total for each depth with different random seeds. That is, we obtain three different sets of optimal parameters for each depth. The resulting

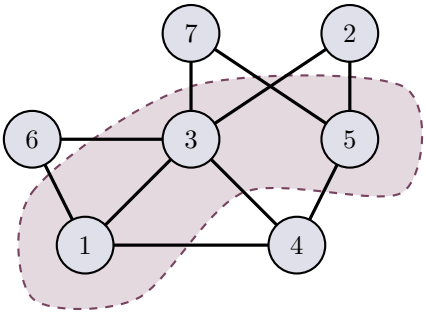


Figure 21: Exemplary graph for the max-cut problem consisting of seven vertices (each labeled by its unique index) and ten edges. The dashed line represents the optimal cut that leads to the unique solution to the problem in form of two subgraphs that are connected by nine edges.

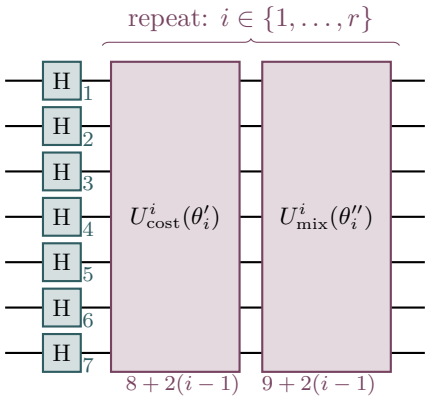


Figure 22: Circuit for QAOA of depth r, which is parameterized by the parameters $\vec{\theta} \in \mathbb{R}^{2r}$. This circuit contains H gates, a cost layer $U^i_{\text{cost}}(\theta'_i)$ depending on the parameter $\theta'_i := \theta_{2(i-1)+1} \in \mathbb{R}$, and a mixing layer $U^i_{\text{mix}}(\theta''_i)$ depending on the parameter $\theta''_i := \theta_{2(i-1)+2} \in \mathbb{R}$. The number near each gate represents its gate index g, Eq. (22).

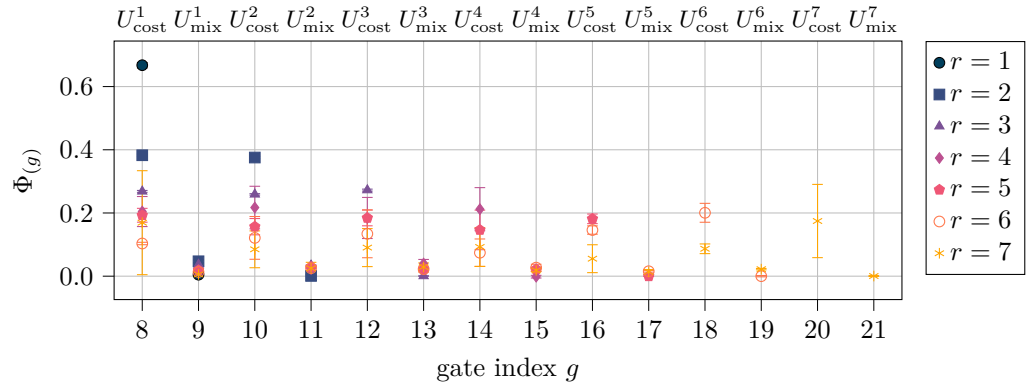
optimization goals are listed in App. I. Based on these fixed parameters, we evaluate the SVQXs for the respective circuits. In total, we consider three different value functions $v(S)$ for this purpose: (i) expressibility, Eq. (36), (ii) entangling capability, Eq. (37), and (iii) energy corresponding to the expectation value of the cost Hamiltonian

$$v(S) = \langle \Psi(S, R, \vec{\theta}) | \mathcal{H} | \Psi(S, R, \vec{\theta}) \rangle \,, \tag{46}$$

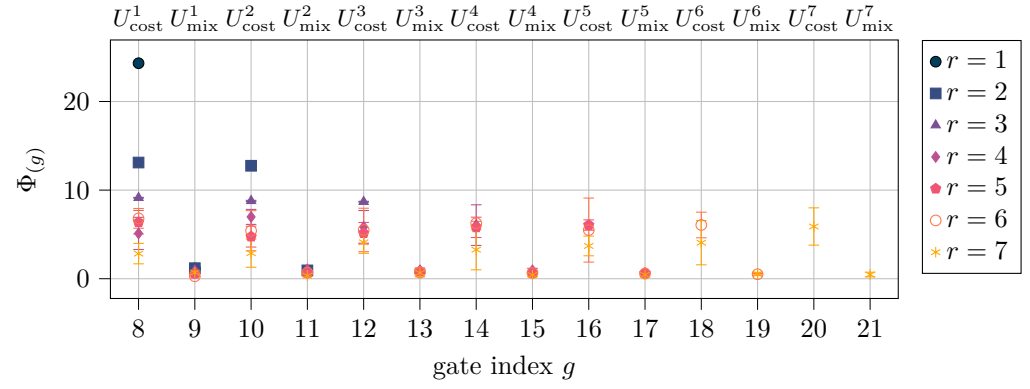
where we recall Eq. (26). The problem Hamiltonian $\mathcal{H}$ is defined in the usual way such that its ground state encodes the solution to the proposed max-cut problem.

We run all evaluations on the `state_CPU` simulator and therefore no randomness is involved in the evaluation of the value functions. We choose $K = 1$ and $\alpha = 1$ for $r \in \{1, 2, 3\}$, $\alpha = 0.1$ for $r \in \{4, 5\}$, $\alpha = 0.01$ for $r = 6$, and $\alpha = 0.001$ for $r = 7$, Eq. (41). For each repetition number, we conduct three independent runs with different random seeds for the algorithm optimizer as well as a different sample in the sense of Eq. (11). The resulting SVQXs are presented in Fig. 23.

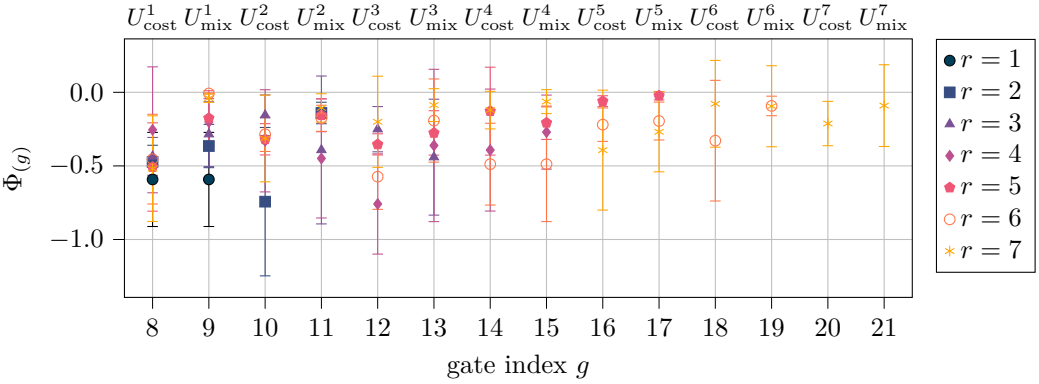
The SVQXs for the expressibility and entangling capability both reveal a similar structure, which clearly shows that cost layers have a dominating contribution, whereas mixing layers have a vanishing contribution. For the energy, no clear structure of the respective SVQXs can be identified and, in addition, the standard deviations over different independent optimization runs are also relatively large. Hence, we can conclude that all gates attribute similarly to the expectation value independent of the depth and their specific contribution might vary depending on the optimized parameters. This in particular means that the mixing layers contribute similarly to cost layers and therefore, pruning mixing layers from the circuit is not expected to bring a general benefit of the algorithm as can be expected from the theory. Summarized, we find from our observations that the exploration of the unitary space with entangled qubits—i.e., the search space—is entirely driven by the cost layers (as



(a) entangling capability

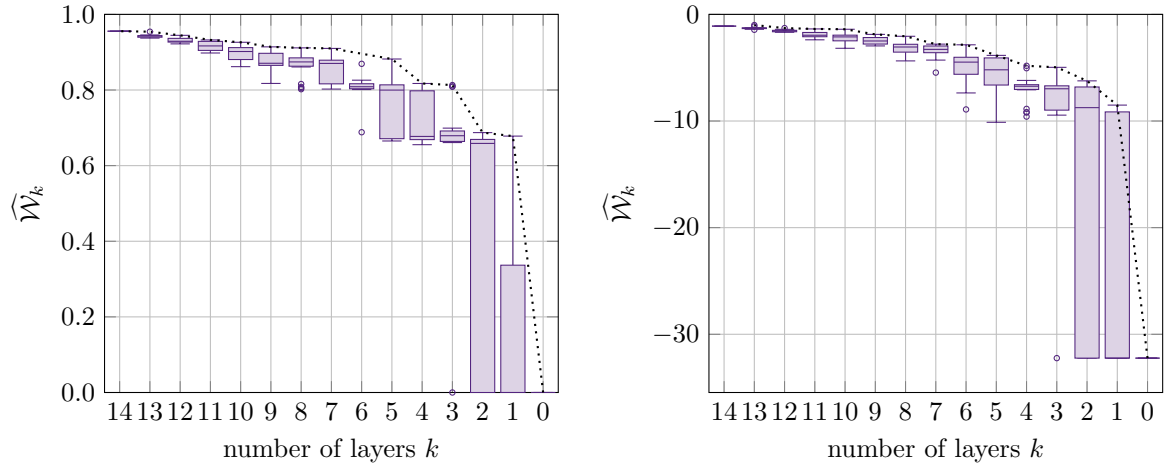


(b) expressibility



(c) energy

Figure 23: SVQXs $\Phi_{(g)}$, Eq. (40), for the QAOA circuit of depth $r \in \{1, \ldots, 7\}$, Fig. 22, with three value functions of interest: (a) entangling capability, Eq. (37), (b) expressibility, Eq. (36), and (c) energy, Eq. (46). For (a) and (b), larger SVQXs are more favorable, whereas smaller SVQXs are more favorable for (c). In analogy to the previous experiments, the gate index g, Eq. (22), is shown on the bottom, whereas the corresponding gate (i.e., layer) name is shown on the top of the plot. For the gates with $g \in R = \{1, \ldots, 7\}$, no SVQX is determined. All evaluations are run on the `state_CPU` simulator. We choose $K = 1$ and $\alpha = 1$ for $r \in \{1, 2, 3\}$, $\alpha = 0.1$ for $r \in \{4, 5\}$, $\alpha = 0.01$ for $r = 6$, and $\alpha = 0.001$ for $r = 7$. For each depth r, we show the mean and one standard deviation over three independent runs.



(a) entangling capability

(b) expressiblity

Figure 24: Distribution of sampled value functions $\widehat{\mathcal{W}}_k$, Eq. (45), for different number of layers k in form of box plots in analogy to Fig. 7. The results are obtained from one optimization run from Fig. 23 with depth $r = 7$. All evaluations are run on a `state_CPU` simulator with $K = 1$ and $\alpha = 0.001$. The dotted line indicates an approximation of the Pareto frontier for a multi-criteria circuit pruning problem with the goal to minimize the number of layers and to maximize the corresponding value function (expressibility and entangling capability, respectively).

the informative part of the algorithm containing the problem structure), whereas the mixing layers (as the uninformative part containing a generic structure) help to minimize the expectation value without contributing to an extension of the search space. Based on these results one could also argue that expressibility and entangling capability are both not necessarily required for energy minimization. And indeed, high expressibility can lead to flatter cost landscapes that hamper optimization [100].

To further investigate the importance of the circuit depth for the entangling capability and expressibility, we conduct a brief analysis of the sampled value function distribution depending on the number of active gates (i.e., layers) according to Eq. (45) in analogy to Fig. 7. For this purpose, we limit ourselves to the case $r = 7$. In this analysis, we do not consider the energy function since (in contrast to the entangling capability and the expressibility) it depends on the optimization parameters which have been determined beforehand based on the complete circuit such that starting with a pruned circuit might converge to a different set of optimization parameters with a different energy. The results are shown in Fig. 24.

As expected, we find that there is a clear tradeoff between the number of layers and the expressibility and entangling capability, respectively. For a smaller number of layers, the distribution also becomes significantly broader.

5 Conclusions

In this work, the important XAI tool of Shapley values (SVs) has been adopted to explain quantum circuits by analyzing the influence of gates (or groups of gates) on quantum computations. We use the term *Shapley values for quantum circuit explanations* (SVQXs) for our approach. While we have mostly focused on its use for approaching model architecture investigations in an XQML context, our method is in fact applicable beyond QML to gate-based quantum computations in general.

The value function for the evaluation of SVQXs can be freely chosen, which allows us to study the impact of quantum gates on a wide range of properties, also within quantum-classical hybrid systems such as a quantum-enhanced ML pipeline. We demonstrate this versatility experimentally with five detailed use cases including QML models for classification and generative modeling as well as quantum optimization tasks with variational quantum circuits. Within these use cases, we perform various explanations with different value functions ranging from model performance over hardware noise and transpilation efficiency to circuit expressibility and entanglement capability. To realize the

experiments, we have used both simulators running on a classical host and two different QPUs based on superconducting quantum hardware.

Given this broad applicability, our method opens up opportunities for monitoring and understanding the behavior of real-world quantum computing systems. At the same time, new research questions emerge, e.g., understanding the relation between qubit error metrics or crosstalk [101] and the context-aware (i.e., gate-specific and circuit-specific) quantities derived via SVQXs to quantify hardware uncertainties for generative models. In addition, our findings also indicate that SVQX could be a promising tool within a QAS framework.

The high complexity, inherited from the classical SVs, is arguably the major limitation of the proposed SVQX approach. A high computational effort is required to cover relevant coalitions and to reduce the uncertainty of the estimated quantities. While our results suggest that an estimation with a subset of coalitions on actual QPUs delivers reasonable results, further research might be required to allow a more general analysis, especially for large scale systems.

Moreover, SVQXs could be modified or extended in many ways as a possible starting point for further studies:

- In place of SVs, Owen values [102, 103] or Banzhaf values [104] can be used, concepts that are already used in classical XAI [105, 106].
- Instead of removing gates that are not part of the coalition, other operations could be performed on those gates, e.g., local inversions as proposed recently to quantify the contribution of individual gates to the overall error on NISQ hardware [107].
- Certain aspects of XQML like, e.g., an intuitive explanation of quantum feature spaces [108], go beyond the scope of classical XAI and might require entirely new explainability approaches.

These exemplary aspects indicate that there are still many open research aspects for the explanation of quantum circuits, even within the limited scope of our approach.

So far, XQML is a novel, yet mostly unexplored field that can in the spirit of [14] be considered as an alternative research direction of QML instead of the search for a quantum advantage. In this sense, we see this contribution as one of the first shots at XQML with a focus on QML model architecture and as a promising cornerstone for further research.

Acknowledgments

Parts of this research have been funded by the Ministry of Science and Health of the State of Rhineland-Palatinate as part of the project *AnQuC*. Parts of this research have been funded by the Federal Ministry of Education and Research of Germany and the state of North-Rhine Westphalia as part of the *Lamarr-Institute for Machine Learning and Artificial Intelligence*. Access to quantum computing hardware has been funded by *Fraunhofer Quantum Now*.

References

[1] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 2nd ed. bookdown, 2022. URL: https://christophm.github.io/interpretable-ml-book.

[2] Ričards Marcinkevičs and Julia E. Vogt. *Interpretability and Explainability: A Machine Learning Zoo Mini-tour*. 2020. DOI: 10.48550/ARXIV.2012.01805. URL: https://arxiv.org/abs/2012.01805.

[3] Ravil I. Mukhamediev et al. "Review of Artificial Intelligence and Machine Learning Technologies: Classification, Restrictions, Opportunities and Challenges". In: *Mathematics* 10.15 (2022). ISSN: 2227-7390. DOI: 10.3390/math10152552. URL: https://www.mdpi.com/2227-7390/10/15/2552.

[4] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. "Explainable AI: A Review of Machine Learning Interpretability Methods". In: *Entropy* 23.1 (2020). ISSN: 1099-4300. DOI: 10.3390/e23010018. URL: https://www.mdpi.com/1099-4300/23/1/18.

[5] Dang Minh et al. "Explainable artificial intelligence: a comprehensive review". In: *Artificial Intelligence Review* 55.5 (June 2022), pp. 3503–3568. ISSN: 1573-7462. DOI: 10.1007/s10462-021-10088-y. URL: https://doi.org/10.1007/s10462-021-10088-y.

[6] Waddah Saeed and Christian Omlin. "Explainable AI (XAI): A systematic meta-survey of current challenges and future opportunities". In: *Knowledge-Based Systems* 263 (2023), p. 110273. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2023.110273. URL: https://www.sciencedirect.com/science/article/pii/S0950705123000230.

[7] Jacob Biamonte et al. "Quantum machine learning". In: *Nature* 549.7671 (Sept. 2017), pp. 195–202. ISSN: 1476-4687. DOI: 10.1038/nature23474. URL: https://doi.org/10.1038/nature23474.

[8] Francesco Petruccione Maria Schuld. *Supervised Learning with Quantum Computers*. 1st ed. Quantum Science and Technology. Springer Cham, 2018. ISBN: 978-3-319-96423-2. DOI: 10.1007/978-3-319-96424-9.

[9] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 10th. USA: Cambridge University Press, 2011. ISBN: 1107002176.

[10] Hsin-Yuan Huang et al. "Power of data in quantum machine learning". In: *Nature Communications* 12.1 (May 2021), pp. 1–9. ISSN: 2041-1723. DOI: 10.1038/s41467-021-22539-9. URL: https://doi.org/10.1038/s41467-021-22539-9.

[11] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. "A rigorous and robust quantum speed-up in supervised machine learning". In: *Nature Physics* 17.9 (July 2021), pp. 1013–1017. DOI: 10.1038/s41567-021-01287-z. URL: https://doi.org/10.1038%2Fs41567-021-01287-z.

[12] Matthias C. Caro et al. "Generalization in quantum machine learning from few training data". In: *Nature Communications* 13.1 (Aug. 2022), pp. 1–11. ISSN: 2041-1723. DOI: 10.1038/s41467-022-32550-3. URL: https://doi.org/10.1038/s41467-022-32550-3.

[13] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (Aug. 2018), p. 79. DOI: 10.22331/q-2018-08-06-79. URL: https://doi.org/10.22331%2Fq-2018-08-06-79.

[14] Maria Schuld and Nathan Killoran. *Is quantum advantage the right goal for quantum machine learning?* 2022. DOI: 10.48550/ARXIV.2203.01340. URL: https://arxiv.org/abs/2203.01340.

[15] Francesco Mercaldo et al. "Towards Explainable Quantum Machine Learning for Mobile Malware Detection and Classification". In: *Applied Sciences* 12.23 (2022). ISSN: 2076–3417. DOI: 10.3390/app122312025. URL: https://www.mdpi.com/2076-3417/12/23/12025.

[16] Luke Power and Krishnendu Guha. *Feature Importance and Explainability in Quantum Machine Learning*. 2024. arXiv: 2405.08917 [cs.LG]. URL: https://arxiv.org/abs/2405.08917.

[17] Yaswitha Gujju, Atsushi Matsuo, and Rudy Raymond. "Quantum machine learning on near-term quantum devices: Current state of supervised and unsupervised techniques for real-world applications". In: *Phys. Rev. Appl.* 21 (6 June 2024), p. 067001. DOI: 10.1103/PhysRevApplied.21.067001. URL: https://link.aps.org/doi/10.1103/PhysRevApplied.21.067001.

[18] Elies Gil-Fuster et al. *Opportunities and limitations of explaining quantum machine learning*. 2024. arXiv: 2412.14753 [quant-ph]. URL: https://arxiv.org/abs/2412.14753.

[19] Giovanni Ciaramella et al. "Exploring Quantum Machine Learning for Explainable Malware Detection". In: *2023 International Joint Conference on Neural Networks (IJCNN)*. Ed. by IJCNN. IEEE, 2023, pp. 1–6. DOI: 10.1109/IJCNN54540.2023.10191964.

[20] Jinkai Tian and Wenjing Yang. "Explainable Quantum Neural Networks: Example-Based and Feature-Based Methods". In: *Electronics* 13.20 (2024). ISSN: 2079-9292. DOI: 10.3390/electronics13204136. URL: https://www.mdpi.com/2079-9292/13/20/4136.

[21] Lirandë Pira and Chris Ferrie. "On the interpretability of quantum neural networks". In: *Quantum Machine Intelligence* 6.2 (Aug. 2024), p. 52. ISSN: 2524-4914. DOI: 10.1007/s42484-024-00191-y. URL: https://doi.org/10.1007/s42484-024-00191-y.

[22] Shaolun Ruan et al. “VIOLET: Visual Analytics for Explainable Quantum Neural Networks”. In: *IEEE Transactions on Visualization and Computer Graphics* 30.6 (2024), pp. 2862–2874. DOI: 10.1109/TVCG.2024.3388557.

[23] Hsin-Yi Lin et al. “Quantum Gradient Class Activation Map for Model Interpretability”. In: *2024 IEEE Workshop on Signal Processing Systems (SiPS)*. Ed. by Javier Gurrola. IEEE, 2024, pp. 165–170. DOI: 10.1109/SiPS62058.2024.00037.

[24] Manuel P. Cuéllar, M. C. Pegalajar, and C. Cano. “Automatic evolutionary design of quantum rule-based systems and applications to quantum reinforcement learning”. In: *Quantum Information Processing* 23.5 (May 2024), p. 179. ISSN: 1573-1332. DOI: 10.1007/s11128-024-04391-0. URL: https://doi.org/10.1007/s11128-024-04391-0.

[25] Fulvio Flamini et al. “Towards interpretable quantum machine learning via single-photon quantum walks”. In: *Quantum Science and Technology* 9.4 (July 2024), p. 045011. DOI: 10.1088/2058-9565/ad5907. URL: https://dx.doi.org/10.1088/2058-9565/ad5907.

[26] Lloyd S Shapley. “A value for n-person games”. In: *Contributions to the Theory of Games* 2.28 (1953), pp. 307–317.

[27] Robert J Aumann and Lloyd S Shapley. *Values of Non-Atomic Games*. Princeton University Press, 1974. URL: http://www.jstor.org/stable/j.ctt13x149m.

[28] Erik Strumbelj and Igor Kononenko. “An efficient explanation of individual classifications using game theory”. In: *The Journal of Machine Learning Research* 11 (2010). Publisher: JMLR. org, pp. 1–18.

[29] Scott M Lundberg and Su-In Lee. “A Unified Approach to Interpreting Model Predictions”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 4765–4774. URL: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf (visited on 02/06/2020).

[30] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. “Axiomatic Attribution for Deep Networks”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. Sydney, NSW, Australia: PMLR, Aug. 2017, pp. 3319–3328. (Visited on 02/18/2020).

[31] Quan-shi Zhang and Song-chun Zhu. “Visual interpretability for deep learning: a survey”. In: *Frontiers of Information Technology & Electronic Engineering* 19.1 (Jan. 2018), pp. 27–39. ISSN: 2095-9230. DOI: 10.1631/FITEE.1700808. URL: https://doi.org/10.1631/FITEE.1700808.

[32] Imrus Salehin et al. “AutoML: A systematic review on automated machine learning with neural architecture search”. In: *Journal of Information and Intelligence* 2.1 (2024), pp. 52–81. ISSN: 2949-7159. DOI: https://doi.org/10.1016/j.jiixd.2023.10.002. URL: https://www.sciencedirect.com/science/article/pii/S2949715923000604.

[33] Sasan Salmani Pour Avval et al. “Systematic review on neural architecture search”. In: *Artificial Intelligence Review* 58.3 (Jan. 2025), p. 73. ISSN: 1573-7462. DOI: 10.1007/s10462-024-11058-w. URL: https://doi.org/10.1007/s10462-024-11058-w.

[34] Florin Leon. “Optimizing neural network topology using Shapley value”. In: *2014 18th International Conference on System Theory, Control and Computing (ICSTCC)*. Ed. by ICSTCC. 2014, pp. 862–867. DOI: 10.1109/ICSTCC.2014.6982527.

[35] Julian Stier et al. “Analysing Neural Network Topologies: a Game Theoretic Approach”. In: *Procedia Computer Science* 126 (2018). Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 22nd International Conference, KES-2018, Belgrade, Serbia, pp. 234–243. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.07.257. URL: https://www.sciencedirect.com/science/article/pii/S187705091831233X.

[36] Han Xiao et al. *Shapley-NAS: Discovering Operation Contribution for Neural Architecture Search*. 2022. arXiv: 2206.09811 [cs.LG]. URL: https://arxiv.org/abs/2206.09811.

[37] Mauricio Diaz-Ortiz Jr et al. *Using Cooperative Game Theory to Prune Neural Networks*. 2023. arXiv: 2311.10468 [cs.LG]. URL: https://arxiv.org/abs/2311.10468.

[38] Babatounde Moctard Oloulade et al. “Shapley-guided pruning for efficient graph neural architecture prediction in distributed learning environments”. In: *Information Sciences* 695 (2025), p. 121695. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2024.121695. URL: https://www.sciencedirect.com/science/article/pii/S0020025524016098.

[39] Rafael Barbudo, Sebastián Ventura, and José Raúl Romero. “Eight years of AutoML: categorisation, review and trends”. In: *Knowledge and Information Systems* 65.12 (Dec. 2023), pp. 5097–5149. ISSN: 0219-3116. DOI: 10.1007/s10115-023-01935-1. URL: https://doi.org/10.1007/s10115-023-01935-1.

[40] Dennis Klau, Marc Zöller, and Christian Tutschku. *Bringing Quantum Algorithms to Automated Machine Learning: A Systematic Review of AutoML Frameworks Regarding Extensibility for QML Algorithms*. 2023. arXiv: 2310.04238 [cs.LG]. URL: https://arxiv.org/abs/2310.04238.

[41] Tomasz Rybotycki and Piotr Gawron. *AQMLator – An Auto Quantum Machine Learning E-Platform*. 2024. arXiv: 2409.18338 [quant-ph]. URL: https://arxiv.org/abs/2409.18338.

[42] Li Ding and Lee Spector. “Evolutionary quantum architecture search for parametrized quantum circuits”. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. Ed. by GECCO. GECCO ’22. Boston, Massachusetts: Association for Computing Machinery, 2022, pp. 2190–2195. ISBN: 9781450392686. DOI: 10.1145/3520304.3534012. URL: https://doi.org/10.1145/3520304.3534012.

[43] Yuxuan Du et al. “Quantum circuit architecture search for variational quantum algorithms”. In: *npj Quantum Information* 8.1 (May 2022), p. 62. ISSN: 2056-6387. DOI: 10.1038/s41534-022-00570-y. URL: https://doi.org/10.1038/s41534-022-00570-y.

[44] Lukas Franken et al. “Quantum Circuit Evolution on NISQ Devices”. In: *2022 IEEE Congress on Evolutionary Computation (CEC)*. Ed. by CEC. IEEE, July 2022, pp. 1–8. DOI: 10.1109/cec55065.2022.9870269. URL: http://dx.doi.org/10.1109/CEC55065.2022.9870269.

[45] Alessandro Giovagnoli et al. “QNEAT: Natural Evolution of Variational Quantum Circuit Architecture”. In: *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*. Ed. by GECCO. GECCO ’23 Companion. Lisbon, Portugal: Association for Computing Machinery, 2023, pp. 647–650. ISBN: 9798400701207. DOI: 10.1145/3583133.3590675. URL: https://doi.org/10.1145/3583133.3590675.

[46] Samuel Yen-Chi Chen. *Differentiable Quantum Architecture Search in Asynchronous Quantum Reinforcement Learning*. 2024. arXiv: 2407.18202 [quant-ph]. URL: https://arxiv.org/abs/2407.18202.

[47] Robert J. J. Aumann. “Economic Applications of the Shapley Value”. In: *Game-Theoretic Methods in General Equilibrium Analysis*. Ed. by Jean-François Mertens and Sylvain Sorin. Dordrecht: Springer Netherlands, 1994, pp. 121–133. ISBN: 978-94-017-1656-7. DOI: 10.1007/978-94-017-1656-7_12. URL: https://doi.org/10.1007/978-94-017-1656-7_12.

[48] Eyal Winter. “The Shapley value”. In: ed. by Robert Aumann and Sergiu Hart. Vol. 3. Handbook of Game Theory with Economic Applications. Elsevier, 2002, pp. 2025–2054. DOI: 10.1016/S1574-0005(02)03016-3. URL: https://www.sciencedirect.com/science/article/pii/S1574000502030163.

[49] Shaheen S. Fatima, Michael Wooldridge, and Nicholas R. Jennings. “A randomized method for the Shapley value for the voting game”. In: *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*. Ed. by AAMAS. AAMAS ’07. Honolulu, Hawaii: Association for Computing Machinery, 2007. ISBN: 9788190426275. DOI: 10.1145/1329125.1329316. URL: https://doi.org/10.1145/1329125.1329316.

[50] Iain Burge, Michel Barbeau, and Joaquin Garcia-Alfaro. *A Shapley Value Estimation Speedup for Efficient Explainable Quantum AI*. 2024. arXiv: 2412.14639 [cs.CR]. URL: https://arxiv.org/abs/2412.14639.

[51] Jonathan J. Burnett et al. “Decoherence benchmarking of superconducting qubits”. In: *npj Quantum Information* 5.1 (June 2019), p. 54. ISSN: 2056-6387. DOI: 10.1038/s41534-019-0168-5. URL: https://doi.org/10.1038/s41534-019-0168-5.

[52] Mingyu Sun and Michael R. Geller. *Efficient characterization of correlated SPAM errors*. 2018. DOI: 10.48550/ARXIV.1810.10523. URL: https://arxiv.org/abs/1810.10523.

[53] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[54] Raoul Heese et al. *Shapley Values with Uncertain Value Functions*. 2023. DOI: 10.48550/ARXIV.2301.08086. URL: https://arxiv.org/abs/2301.08086.

[55] Benedek Rozemberczki et al. *The Shapley Value in Machine Learning*. 2022. arXiv: 2202.05594 [cs.LG]. URL: https://arxiv.org/abs/2202.05594.

[56] Mukund Sundararajan and Amir Najmi. "The Many Shapley Values for Model Explanation". In: *International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 9269–9278.

[57] Guy Van den Broeck et al. "On the tractability of SHAP explanations". In: *Journal of Artificial Intelligence Research* 74 (2022), pp. 851–886.

[58] I. Elizabeth Kumar et al. "Problems with Shapley-value-based explanations as feature importance measures". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 5491–5500. URL: https://proceedings.mlr.press/v119/kumar20e.html.

[59] Xuanxiang Huang and Joao Marques-Silva. *The Inadequacy of Shapley Values for Explainability*. 2023. arXiv: 2302.08160 [cs.LG]. URL: https://arxiv.org/abs/2302.08160.

[60] Vedran Dunjko, Jacob M. Taylor, and Hans J. Briegel. "Quantum-Enhanced Machine Learning". In: *Physical Review Letters* 117.13 (Sept. 2016). ISSN: 1079-7114. DOI: 10.1103/physrevlett.117.130501. URL: http://dx.doi.org/10.1103/PhysRevLett.117.130501.

[61] Manuela Weigold et al. "Data Encoding Patterns for Quantum Computing". In: *Proceedings of the 27th Conference on Pattern Languages of Programs*. Ed. by Rebecca Wirfs-Brock. PLoP '20. Virtual Event: The Hillside Group, 2020, pp. 1–11. ISBN: 9781941652169.

[62] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. "Effect of data encoding on the expressive power of variational quantum-machine-learning models". In: *Physical Review A* 103.3 (Mar. 2021). DOI: 10.1103/physreva.103.032430. URL: https://doi.org/10.1103%2Fphysreva.103.032430.

[63] Jay Gambetta. *Expanding the IBM Quantum roadmap to anticipate the future of quantum-centric supercomputing*. 2022. URL: https://research.ibm.com/blog/ibm-quantum-roadmap-2025 (visited on 06/22/2022).

[64] M. Cerezo et al. "Variational quantum algorithms". In: *Nature Reviews Physics* 3.9 (Aug. 2021), pp. 625–644. DOI: 10.1038/s42254-021-00348-9. URL: https://doi.org/10.1038%2Fs42254-021-00348-9.

[65] Maria Schuld. *Supervised quantum machine learning models are kernel methods*. 2021. arXiv: 2101.11020 [quant-ph].

[66] Oleksandr Kyriienko and Einar B. Magnusson. *Unsupervised quantum machine learning for fraud detection*. 2022. arXiv: 2208.01203 [quant-ph].

[67] Nico Meyer et al. *A Survey on Quantum Reinforcement Learning*. 2022. arXiv: 2211.03464 [quant-ph].

[68] Faisal Shah Khan et al. "Quantum games: a review of the history, current state, and interpretation". In: *Quantum Information Processing* 17.11 (Oct. 2018). DOI: 10.1007/s11128-018-2082-8. URL: https://doi.org/10.1007%2Fs11128-018-2082-8.

[69] Tobias Haug, Kishor Bharti, and M.S. Kim. "Capacity and Quantum Geometry of Parametrized Quantum Circuits". In: *PRX Quantum* 2.4 (Oct. 2021). DOI: 10.1103/prxquantum.2.040309. URL: https://doi.org/10.1103%2Fprxquantum.2.040309.

[70] Adriano Barenco et al. “Stabilization of Quantum Computations by Symmetrization”. In: *SIAM Journal on Computing* 26.5 (1997), pp. 1541–1557. DOI: 10.1137/S0097539796302452. eprint: https://doi.org/10.1137/S0097539796302452. URL: https://doi.org/10.1137/S0097539796302452.

[71] Harry Buhrman et al. “Quantum Fingerprinting”. In: *Physical Review Letters* 87.16 (Sept. 2001), p. 167902. DOI: 10.1103/PhysRevLett.87.167902. URL: https://link.aps.org/doi/10.1103/PhysRevLett.87.167902.

[72] Leonardo Zambrano et al. “Estimation of pure quantum states in high dimension at the limit of quantum accuracy through complex optimization and statistical inference”. In: *Scientific Reports* 10.1 (July 2020), p. 12781. ISSN: 2045-2322. DOI: 10.1038/s41598-020-69646-z. URL: https://doi.org/10.1038/s41598-020-69646-z.

[73] Yiqing Zhou, E. Miles Stoudenmire, and Xavier Waintal. “What Limits the Simulation of Quantum Computers?” In: *Physical Review X* 10.4 (Nov. 2020), p. 041038. DOI: 10.1103/PhysRevX.10.041038. URL: https://link.aps.org/doi/10.1103/PhysRevX.10.041038.

[74] Konstantinos Georgopoulos, Clive Emary, and Paolo Zuliani. “Modeling and simulating the noisy behavior of near-term quantum computers”. In: *Physical Review A* 104.6 (Dec. 2021). DOI: 10.1103/physreva.104.062432. URL: https://doi.org/10.1103%2Fphysreva.104.062432.

[75] Sukin Sim, Peter D. Johnson, and Alán Aspuru-Guzik. “Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms”. In: *Advanced Quantum Technologies* 2.12 (Oct. 2019), p. 1900070. DOI: 10.1002/qute.201900070. URL: https://doi.org/10.1002%2Fqute.201900070.

[76] David A. Meyer and Nolan R. Wallach. “Global entanglement in multiparticle systems”. In: *Journal of Mathematical Physics* 43.9 (2002), pp. 4273–4278. DOI: 10.1063/1.1497700. eprint: https://doi.org/10.1063/1.1497700. URL: https://doi.org/10.1063/1.1497700.

[77] E. Hellinger. “Neue Begründung der Theorie quadratischer Formen von unendlichvielen Veränderlichen”. In: *Journal für die reine und angewandte Mathematik* 1909.136 (1909), pp. 210–271. DOI: doi:10.1515/crll.1909.136.210. URL: https://doi.org/10.1515/crll.1909.136.210.

[78] MD Sajid Anis et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2019. DOI: 10.5281/zenodo.2573505.

[79] Raoul Heese. *Quantum Shapley Value Toolbox*. 2023. URL: https://github.com/RaoulHeese/qshaptools (visited on 04/02/2023).

[80] Raoul Heese et al. *Data for: Explainable Quantum Machine Learning*. Version 1.0. Feb. 2023. DOI: 10.5281/zenodo.7603939. URL: https://doi.org/10.5281/zenodo.7603939 (visited on 03/02/2023).

[81] David W. Aha. *Tic-Tac-Toe Endgame Data Set*. 1991. URL: https://archive.ics.uci.edu/ml/datasets/Tic-Tac-Toe+Endgame (visited on 06/01/2021).

[82] IBM. *IBM Quantum Germany*. 2022. URL: https://de.quantum-computing.ibm.com (visited on 12/21/2022).

[83] IBM. *IBM Quantum*. 2022. URL: https://quantum-computing.ibm.com (visited on 12/21/2022).

[84] Vojtěch Havlíček et al. “Supervised learning with quantum-enhanced feature spaces”. In: *Nature* 567.7747 (Mar. 2019), pp. 209–212. DOI: 10.1038/s41586-019-0980-2. URL: https://doi.org/10.1038%2Fs41586-019-0980-2.

[85] Corinna Cortes and Vladimir Vapnik. “Support-vector networks”. In: *Machine Learning* 20.3 (Sept. 1995), pp. 273–297. ISSN: 1573-0565. DOI: 10.1007/BF00994018. URL: https://doi.org/10.1007/BF00994018.

[86] Gian Gentinetta et al. *The complexity of quantum support vector machines*. 2022. DOI: 10.48550/ARXIV.2203.00031. URL: https://arxiv.org/abs/2203.00031.

[87] Sukin Sim et al. “Adaptive pruning-based optimization of parameterized quantum circuits”. In: *Quantum Science and Technology* 6.2 (Mar. 2021), p. 025019. DOI: 10.1088/2058-9565/abe107. URL: https://doi.org/10.1088/2058-9565/abe107.

[88] M. J. D. Powell. *A View of Algorithms for Optimization without Derivatives*. Tech. rep. 5. Institute of Mathematics and its Applications, 2007, pp. 170–174.

[89] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. "Quantum Generative Adversarial Networks for learning and loading random distributions". In: *npj Quantum Information* 5.1 (Nov. 2019), p. 103. ISSN: 2056-6387. DOI: 10.1038/s41534-019-0223-2. URL: https://doi.org/10.1038/s41534-019-0223-2.

[90] Bobak Toussi Kiani et al. "Learning quantum data with the quantum earth mover's distance". In: *Quantum Science and Technology* 7.4 (June 2022), p. 045002. DOI: 10.1088/2058-9565/ac79c9. URL: https://doi.org/10.1088%2F2058-9565%2Fac79c9.

[91] Paul D. Nation et al. "Scalable Mitigation of Measurement Errors on Quantum Computers". In: *PRX Quantum* 2.4 (Nov. 2021). DOI: 10.1103/prxquantum.2.040326. URL: https://doi.org/10.1103%2Fprxquantum.2.040326.

[92] Betis Baheri et al. "Quantum Noise in the Flow of Time: A Temporal Study of the Noise in Quantum Computers". In: *2022 IEEE 28th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. Ed. by Alessandro Savino et al. Sept. 2022, pp. 1–5. DOI: 10.1109/IOLTS56730.2022.9897404.

[93] Victoria Zhang and Paul D. Nation. *Characterizing quantum processors using discrete time crystals*. 2023. DOI: 10.48550/ARXIV.2301.07625. URL: https://arxiv.org/abs/2301.07625.

[94] Raoul Heese, Patricia Bickert, and Astrid Elisa Niederle. "Representation of binary classification trees with binary features by quantum circuits". In: *Quantum* 6 (Mar. 2022), p. 676. ISSN: 2521-327X. DOI: 10.22331/q-2022-03-30-676. URL: https://doi.org/10.22331/q-2022-03-30-676.

[95] Raoul Heese. *Quantum Decision Trees (qtree)*. 2022. URL: https://github.com/RaoulHeese/qtree (visited on 09/22/2022).

[96] D. Coppersmith. *An approximate Fourier transform useful in quantum factoring*. 2002. DOI: 10.48550/ARXIV.QUANT-PH/0201067. URL: https://arxiv.org/abs/quant-ph/0201067.

[97] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. 2014. DOI: 10.48550/ARXIV.1411.4028. URL: https://arxiv.org/abs/1411.4028.

[98] Nikolaj Moll et al. "Quantum optimization using variational algorithms on near-term quantum devices". In: *Quantum Science and Technology* 3.3 (June 2018), p. 030503. DOI: 10.1088/2058-9565/aab822. URL: https://doi.org/10.1088%2F2058-9565%2Faab822.

[99] Stuart Hadfield et al. "From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz". In: *Algorithms* 12.2 (Feb. 2019), p. 34. DOI: 10.3390/a12020034. URL: https://doi.org/10.3390%2Fa12020034.

[100] Zoë Holmes et al. "Connecting Ansatz Expressibility to Gradient Magnitudes and Barren Plateaus". In: *PRX Quantum* 3.1 (Jan. 2022). DOI: 10.1103/prxquantum.3.010313. URL: https://doi.org/10.1103%2Fprxquantum.3.010313.

[101] Mohan Sarovar et al. "Detecting crosstalk errors in quantum information processors". In: *Quantum* 4 (Sept. 2020), p. 321. DOI: 10.22331/q-2020-09-11-321. URL: https://doi.org/10.22331%2Fq-2020-09-11-321.

[102] Guilliermo Owen. "Values of Games with a Priori Unions". In: *Mathematical Economics and Game Theory*. Ed. by Rudolf Henn and Otto Moeschlin. Berlin, Heidelberg: Springer Berlin Heidelberg, 1977, pp. 76–88. ISBN: 978-3-642-45494-3.

[103] Susana López and Martha Saboya. "On the relationship between Shapley and Owen values". In: *Central European Journal of Operations Research* 17.4 (Dec. 2009), pp. 415–423. ISSN: 1613-9178. DOI: 10.1007/s10100-009-0100-8. URL: https://doi.org/10.1007/s10100-009-0100-8.

[104] J.F. Banzhaf. "Weighted voting doesn't work: A mathematical analysis". In: *Rutgers Law Review* 19.2 (1965), pp. 317–343.

[105] Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. *Generating Hierarchical Explanations on Text Classification via Feature Interaction Detection*. 2020. arXiv: 2004.02015 [cs.CL].

[106] Adam Karczmarz et al. *Improved Feature Importance Computations for Tree Models: Shapley vs. Banzhaf*. 2021. arXiv: 2108.04126 [cs.LG]. URL: https://arxiv.org/abs/2108.04126.

[107] Fernando A. Calderon-Vargas et al. "Quantum circuit debugging and sensitivity analysis via local inversions". In: *Quantum* 7 (Feb. 2023), p. 921. DOI: 10.22331/q-2023-02-09-921. URL: https://doi.org/10.22331%2Fq-2023-02-09-921.

[108] Seth Lloyd et al. *Quantum embeddings for machine learning*. 2020. DOI: 10.48550/ARXIV.2001.03622. URL: https://arxiv.org/abs/2001.03622.

[109] Aram W. Harrow and Richard A. Low. "Random Quantum Circuits are Approximate 2-designs". In: *Communications in Mathematical Physics* 291.1 (July 2009), pp. 257–302. DOI: 10.1007/s00220-009-0873-6. URL: https://doi.org/10.1007%2Fs00220-009-0873-6.

[110] Kouhei Nakaji and Naoki Yamamoto. "Expressibility of the alternating layered ansatz for quantum computation". In: *Quantum* 5 (Apr. 2021), p. 434. DOI: 10.22331/q-2021-04-19-434. URL: https://doi.org/10.22331%2Fq-2021-04-19-434.

[111] Yuxuan Du et al. "Efficient Measure for the Expressivity of Variational Quantum Algorithms". In: *Physical Review Letters* 128.8 (Feb. 2022). DOI: 10.1103/physrevlett.128.080506. URL: https://doi.org/10.1103%2Fphysrevlett.128.080506.

[112] Gavin K. Brennen. *An observable measure of entanglement for pure states of multi-qubit systems*. 2003. DOI: 10.48550/ARXIV.QUANT-PH/0305094. URL: https://arxiv.org/abs/quant-ph/0305094.

[113] Robert R. Tucci. *An Introduction to Cartan's KAK Decomposition for QC Programmers*. 2005. DOI: 10.48550/ARXIV.QUANT-PH/0507171. URL: https://arxiv.org/abs/quant-ph/0507171.

[114] Swamit S. Tannu and Moinuddin K. Qureshi. *A Case for Variability-Aware Policies for NISQ-Era Quantum Computers*. 2018. DOI: 10.48550/ARXIV.1805.10224. URL: https://arxiv.org/abs/1805.10224.

[115] Prakash Murali et al. *Noise-Adaptive Compiler Mappings for Noisy Intermediate-Scale Quantum Computers*. 2019. DOI: 10.48550/ARXIV.1901.11054. URL: https://arxiv.org/abs/1901.11054.

[116] Ellis Wilson, Sudhakar Singh, and Frank Mueller. "Just-in-time Quantum Circuit Transpilation Reduces Noise". In: *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Ed. by Hausi A. Müller and Greg Byrd et al. 2020, pp. 345–355. DOI: 10.1109/QCE49297.2020.00050.

[117] Nathan Earnest, Caroline Tornow, and Daniel J. Egger. "Pulse-efficient circuit transpilation for quantum applications on cross-resonance-based hardware". In: *Physical Review Research* 3 (4 Oct. 2021), p. 043088. DOI: 10.1103/PhysRevResearch.3.043088. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.3.043088.

[118] Paul Nation, Matthew Treinish, and Clemens Possel. *mapomatic*. 2022. URL: https://github.com/Qiskit-Partners/mapomatic (visited on 12/21/2022).

[119] Andrew W. Cross et al. "Validating quantum computers using randomized model circuits". In: *Physical Review A* 100 (3 Sept. 2019), p. 032328. DOI: 10.1103/PhysRevA.100.032328. URL: https://link.aps.org/doi/10.1103/PhysRevA.100.032328.

[120] Andrew Wack et al. *Quality, Speed, and Scale: three key attributes to measure the performance of near-term quantum computers*. 2021. DOI: 10.48550/ARXIV.2110.14108. URL: https://arxiv.org/abs/2110.14108.

Appendices

A Expressibility

In this appendix section, we briefly present an estimator for the expressibility along the lines of [75] with the minor extension that we focus on a formulation that can actually be measured on a QPU. For this purpose, we consider an ansatz circuit consisting of q qubits and a corresponding unitary, Eq. (26), that is fully determined by a parameter vector $\vec{\theta} \in \mathcal{P}$, whereas a constant feature vector $\vec{x} \in \mathcal{X}$ (i.e., $\vec{x} = \text{const.}$) is presumed. An estimator of the expressibility of this circuit is given by the Kullback-Leibler divergence

$$\eta(S, R, \vec{x}, c_p, c_b, \mathcal{F}) := D_{\mathrm{KL}}[\hat{P}_{\mathrm{circ}}^{c_p,c_b}(F, S, R, \vec{x}, \mathcal{F}) \| P_{\mathrm{Haar}}(F)]_F \geq 0 \tag{47}$$

with respect to the fidelity $F \in [0, 1]$.

The first probability distribution in Eq. (47) is an estimation of the probability distribution of fidelities for the circuit of interest. It can be obtained by introducing two i.i.d. random variables $\vec{\boldsymbol{\vartheta}}_1 \sim u(\mathcal{P})$ and $\vec{\boldsymbol{\vartheta}}_2 \sim u(\mathcal{P})$, where $u(\mathcal{P})$ denotes the uniform distribution on $\mathcal{P}$. These random variables represent randomly chosen parameter vectors. In total, c_p pairs $\{(\vec{\vartheta}_1^1, \vec{\vartheta}_2^1), \ldots, (\vec{\vartheta}_1^{c_p}, \vec{\vartheta}_2^{c_p})\}$ of those random variables are sampled. For each realization pair $i \in \{1, \ldots, c_p\}$, a SWAP test [70, 71] is performed based on s shots.

From these measurements, we obtain the set of bit strings

$$\vec{B}^i(\vec{x}, \vec{\vartheta}_1^i, \vec{\vartheta}_2^i) \equiv \vec{B}^i := \{b_1^i, \ldots, b_s^i\} \tag{48}$$

in analogy to Eq. (17) with $b_j^i \in \{0, 1\} \, \forall \, j \in \{1, \ldots, s\}$. Furthermore, $b_j^i \sim p_{\mathrm{SWAP}}(b_j^i)$ with

$$p_{\mathrm{SWAP}}(b_j^i = 1) = \frac{1}{2} - \frac{1}{2} F^i, \tag{49}$$

where

$$F^i \equiv F^i(S, R, \vec{x}, \vec{\vartheta}_1^i, \vec{\vartheta}_2^i) := |\, \langle \Psi(S, R, \vec{x}, \vec{\vartheta}_1^i) | \Psi(S, R, \vec{x}, \vec{\vartheta}_2^i) \rangle \,| \in [0, 1] \tag{50}$$

denotes the fidelity of the two states resulting from the circuit of interest for the randomly sampled parameter vector pair i, where we recall Eq. (16). Thus, an estimate

$$\hat{F}^i := 1 - 2 \frac{|\{b \,|\, b \in \vec{B}^i, b = 1\}|}{s} \tag{51}$$

of F^i can be obtained from the multiset of bit strings.

In total, sc_p shots are necessary to measure the set of fidelities $\mathcal{F} := \{\hat{F}^1, \ldots, \hat{F}^{c_p}\}$ for all pairs. From a histogram of $\mathcal{F}$ based on c_b evenly sized bins, the probability distribution

$$\hat{P}_{\mathrm{circ}}^{c_p,c_b}(F, S, R, \vec{x}, \mathcal{F}) := \frac{1}{c_p} \left| \left\{ F' \,|\, F' \in \mathcal{F}, \left[\frac{j(c_b, F)}{c_b} \leq F' < \frac{j(c_b, F) + 1}{c_b} \right] \right. \right. \\ \left. \left. \vee \, [j(c_b, F) + 1 = c_b \wedge F' = 1] \right\} \right|^2 \tag{52}$$

with

$$j(c_b, F) := \begin{cases} \lfloor F c_b \rfloor & \text{if } 0 \leq F < 1 \\ c_b - 1 & \text{if } F = 1 \end{cases} \tag{53}$$

can be obtained.

The second probability distribution in Eq. (47) represents the probability density function of fidelities for the ensemble of Haar random states with the analytical form

$$P_{\mathrm{Haar}}(F) := (2^q - 1)(1 - F)^{2^q - 2}, \tag{54}$$

which only depends on the number of qubits q.

The presented expressibility measure, Eq. (47), is based on a distance measure such that a high value indicates a low expressibility and vice versa. For a more detailed discussion, we refer to [75, 109, 110] and references therein. An alternative measure can also be found in [111].

B Entangling capability

This appendix section outlines a measure for the entangling capability following [75]. We extend the original concept with a brief description of how it can be measured on a QPU by means of quantum tomography. In analogy to the expressibility calculation in App. A, we consider an ansatz circuit consisting of q qubits and a corresponding unitary, Eq. (26), that is fully determined by a parameter vector $\vec{\theta} \in \mathcal{P}$, whereas a constant feature vector $\vec{x} \in \mathcal{X}$ (i.e., $\vec{x} = \text{const.}$) is presumed. A measure for the entangling capability of this circuit is given by the mean

$$\lambda(S, R, \vec{x}, c_s, \mathcal{T}) := \frac{1}{c_s} \sum_{i=1}^{c_s} Q(|\hat{\Psi}^i\rangle) \in [0, 1] \tag{55}$$

over the Meyer-Wallach entanglement measure [76], that is defined as

$$Q(|\Psi\rangle) := \frac{4}{n} \sum_{j=1}^{q} D(\iota_j(0) |\Psi\rangle, \iota_j(1) |\Psi\rangle) \in [0, 1] \tag{56}$$

for a q-qubit state $|\Psi\rangle$. Here, we use the mapping

$$\iota_j(b) |b_1 \cdots b_q\rangle := \delta_{bb_j} |b_1 \cdots \hat{b}_j \cdots b_q\rangle \tag{57}$$

that acts on the computational basis, where $\hat{b}_j$ denotes the absence of the jth qubit and $b \in \{0, 1\}$. Furthermore, we make use of the generalized distance

$$D(|u\rangle, |v\rangle) := \frac{1}{2} \sum_{i,j=1}^{2^q-2} |u_i v_j - u_j v_i|^2 \tag{58}$$

with $|u\rangle := \sum_{i=1}^{2^q-2} u_i |i\rangle$ and $|v\rangle := \sum_{i=1}^{2^q-2} v_i |i\rangle$, respectively.

In Eq. (55), the Meyer-Wallach entanglement measure is evaluated for all quantum states in $\mathcal{Q} := \{|\hat{\Psi}^i\rangle, \ldots, |\hat{\Psi}^{c_s}\rangle\}$. To determine these states, a random variable $\vec{\boldsymbol{\vartheta}} \sim u(\mathcal{P})$ is introduced and c_s realizations $\mathcal{T} := \{\vec{\vartheta}^1 \ldots, \vec{\vartheta}^{c_s}\}$ are obtained. For each realization $i \in \{1, \ldots, c_s\}$, a quantum state tomography [72] is performed to obtain

$$|\hat{\Psi}^i\rangle \equiv |\hat{\Psi}^i(S, R, \vec{x}, \vec{\vartheta}^i, \vec{B}^i)\rangle \approx |\Psi(S, R, \vec{x}, \vec{\vartheta}^i)\rangle, \tag{59}$$

where we recall Eq. (26). We presume that a multiset

$$\vec{B}^i \equiv \vec{B}^i(S, R, \vec{x}, \mathcal{T}) := \{b_1^i, \ldots, b_s^i\} \tag{60}$$

of s measurement results in form of bit strings in analogy to Eq. (17) with $b_j^i \in \{0, 1\} \, \forall \, j \in \{1, \ldots, s\}$ is necessary to achieve this goal, but do not further specify the tomographic process

$$\vec{B}^i \mapsto |\hat{\Psi}^i\rangle, \tag{61}$$

which can for example correspond to the iterative algorithm presented in [72]. In total, sc_s shots are necessary to estimate all quantum states of interest.

According to the Meyer-Wallach entanglement measure, the entangling capability, Eq. (55), becomes larger for more entangled states in $\mathcal{T}$. For a more detailed discussion, we refer to [75, 76] and references therein. An alternative interpretation and evaluation of Eq. (55) is presented in [112].

C Hellinger fidelity

In this appendix section, we briefly outline a metric to quantify NISQ hardware deficiencies. For this purpose, we consider a circuit of interest consisting of q qubits and a corresponding unitary, Eq. (26), that may optionally depend on a parameter vector $\vec{\theta} \in \mathcal{P}$ and feature vector $\vec{x} \in \mathcal{X}$, respectively, which are both presumed to be constant (i.e., $\vec{\theta} = \text{const.}$ and $\vec{x} = \text{const.}$). The circuit is evaluated independently with two different approaches. First, on a QPU to obtain the measurement results

$\vec{B}(S)$, Eq. (27), as a collection of bit strings. And second, with an idealized (i.e., noise-free) simulator running on classical hardware.

The estimated probability distribution of bit strings $\vec{b} \in \{0,1\}^q$ that can be obtained from the QPU is given by

$$m_{\mathrm{qc}}(\vec{b}, \vec{B}(S)) := \frac{|\{\vec{b}' \,|\, \vec{b}' \in \vec{B}(S), \vec{b}' = \vec{b}\}|}{s} \tag{62}$$

for s shots. On the other hand, the corresponding probability distribution obtained from the classical simulator is given by $m_{\mathrm{sim}}(\vec{b}) \equiv m_{\mathrm{sim}}(\vec{b}, S, R, \vec{x}, \vec{\theta})$ according to Eq. (28). The Hellinger fidelity [77]

$$H(S, R, \vec{x}, \vec{\theta}, \vec{B}(S)) := \left[\sum_{\vec{b} \in \{0,1\}^q} \sqrt{m_{\mathrm{qc}}(\vec{b}, \vec{B}(S)) m_{\mathrm{sim}}(\vec{b})} \right]^2 \in [0,1] \tag{63}$$

is a similarity measure of the two distributions. It can therefore be used as a metric to quantify NISQ hardware deficiencies. By definition, a smaller fidelity indicates larger hardware deficiencies.

D Estimated execution efficiency

This appendix section considers the estimated execution efficiency as a measure of how efficiently a circuit can be executed on a specific NISQ hardware device. The challenge of efficient execution arises because of two typical limitations of NISQ hardware. First, only a limited set of gates can be physically realized on a hardware device. However, presuming that this set of available gates is a universal set of quantum gates, all other gates can be expressed in terms of the available gates [9, 113]. Second, the connectivity of physical qubits in a hardware device is limited in the sense that gates acting on two (or more) qubits can only be applied on certain pairs (or sets) of qubits. However, this limitation can be overcome by introducing additional gates to swap qubits accordingly [9]. Due to these two limitations, any quantum circuit must be subjected to a suitable equivalence transformation before execution such that it contains only gates from the set of available gates $\mathcal{U}$ as defined by the hardware device of interest. This equivalence transformation is also called transpilation [78].

Specifically, we consider a transpilation procedure T as an equivalence transformation of a gate set $P(S,R) := \{A_a \,|\, a \in S\} \cup R$ from Eq. (25) into a new gate set $P'(S,R) := \{1, \ldots, G'\} \cup R$ according to

$$T : P(S,R) \mapsto P'(S,R) \tag{64}$$

such that

$$U(S, R, \vec{x}, \vec{\theta}) = U'(S, R, \vec{x}, \vec{\theta}), \tag{65}$$

where

$$U'(S, R, \vec{x}, \vec{\theta}) := \prod_{g \in P'(S,R)} U_g(\vec{x}^g, \vec{\theta}^g) \tag{66}$$

in analogy to Eq. (25) with unitary operators $U'_g(\vec{x}^g, \vec{\theta}^g) \in \mathcal{U}$ for all $g \in P'(S,R)$. The set $\mathcal{U}$ contains all gates that are available on the hardware device of interest. For the sake of simplicity, we presume that $\mathcal{U}$ contains only one-qubit and two-qubit gates.

Generally, the equivalence transformation of a circuit is ambiguous and can therefore lead to different gate decompositions with a varying number of gates. Since every gate introduces hardware-related uncertainty into the corresponding QPU computation, an efficient transpilation with at least gates as possible in the resulting circuit is desired, which is a non-trivial optimization task. The estimated transpilation efficiency

$$\tau_T(S, R, \mathcal{U}, s_1, s_2) := \sum_{g \in P'(S,R)} \hat{\tau}(U'_g(\vec{x}^g, \vec{\theta}^g), s_1, s_2) \leq 0 \tag{67}$$

is a possible metric to measure the quality of a certain transpilation T for an available gate set $\mathcal{U}$ based on the resulting gate decomposition. The penalty function

$$\hat{\tau}(U, s_1, s_2) := \begin{cases} s_2 & \text{if } U \in \mathcal{U}_2 \\ s_1 & \text{otherwise} \end{cases} \tag{68}$$

Table 1: NISQ device specifications as provided by IBM.

QPU	processor type	version	number of qubits	quantum volume [119]	CLOPS [120]
`ibmq_ehningen`	Falcon r5.11	3.1.22	27	64	1.9×10^3
`ibm_oslo`	Falcon r5.11H	1.0.14	7	32	2.6×10^3
QPU	median CX error	median readout error	median T_1	median T_2	reference
`ibmq_ehningen`	7.218×10^{-3}	9.300×10^{-3}	127.09 µs	143.27 µs	[82]
`ibm_oslo`	9.748×10^{-3}	1.950×10^{-2}	148.03 µs	41.79 µs	[83]

estimates the decrease in efficiency for a higher number of gates by assigning a negative penalty value to each gate, where $\mathcal{U}_2 \subset \mathcal{U}$ contains all available two-qubit gates. Hence, a higher penalty score indicates a higher estimated transpilation efficiency (since less noise can be expected). The penalty parameters $s_1, s_2 \in \mathbb{R}_{<0}$ can be chosen arbitrarily, where typically $0 > s_1 > s_2$ to reflect that two-qubit gates induce more noise than one-qubit gates.

The estimated execution efficiency

$$\tau(S, R, \mathcal{U}, s_1, s_2) := \max_T \tau_T(S, R, \mathcal{U}, s_1, s_2) \leq 0 \tag{69}$$

is defined as the largest estimated transpilation efficiency, Eq. (67), for all possible transpilation procedures. Summarized, the estimated execution efficiency represents the largest possible sum over the penalty scores of all gates and can therefore be used to estimate how efficiently the corresponding circuit can be executed on the hardware device of interest.

More generally, the efficiency of a transpilation procedure may not only depend on the absolute number of gates, but also the noise properties of individual qubits and gates as well as low-level hardware realizations to achieve a better performance [114–118]. We do not consider this more complex case here, which can, however, be brought to the same formalism by defining a suitably modified penalty function.

E NISQ devices

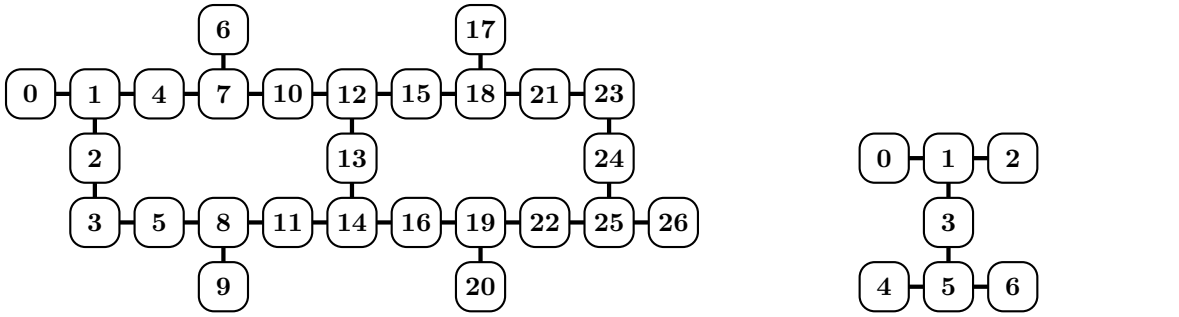
We use two NISQ devices to perform our experiments: `ibmq_ehningen` (accessible via [82]) and `ibm_oslo` (accessible via [83]). The specifications of these devices—as provided by IBM—are listed in Tab. 1. Both QPUs can realize four elementary gates that form a universal gate set: (i) three one-qubit gates, namely the Z-rotation gate $\mathrm{R_Z}(\lambda)$ (parameterized by an angle λ), the NOT gate X, and the SX gate $\sqrt{\mathrm{X}}$, as well as (ii) the two-qubit controlled NOT gate CX. In Fig. 25, we show the hardware connectivity maps. Each node represents a qubit labeled with the corresponding physical qubit index (that allows a unique identification) and each edge indicate the respective qubit connections that allow the realization of joint CX gates.

F Circuit symbols

Throughout this paper, we use the circuit symbols shown in Fig. 26 to denote quantum gates. All gates are used as defined by Qiskit [78].

G QSVM marginal contributions

In this appendix section, we provide supplementary material for the QSVM use case from Sec. 4.1. Specifically, we investigate the distribution of marginal contributions $p_i(\Delta_i v)$, Eq. (9), for each player index i, where i corresponds to the gate index g. The results are based on the evaluation from the `state_CPU` simulator with $K = 1$. For $r \in \{1, 2\}$, we use $\alpha = 1$, whereas for $r = 3$, we use $\alpha = 0.01$ with a single run. That is, we use the same setup as for Fig. 6, but consider only the first run for $r = 3$.



(a) `ibmq_ehningen` **(b)** `ibm_oslo`

Figure 25: Hardware connectivity map for the two NISQ devices of interest. We show the physical qubit indices (unique identifiers) and connections between qubits that allow the realization of joint CX gates.

(a) H **(b)** $\mathrm{P}(\lambda)$ **(c)** $\mathrm{R_Y}(\lambda)$ **(d)** $\mathrm{CP}(\lambda)$ **(e)** CX **(f)** S

Figure 26: Overview over the circuit symbols we use throughout the paper: Hadamard gate H, phase gate $\mathrm{P}(\lambda)$, Y-rotation gate $\mathrm{R_Y}(\lambda)$, controlled phase gate $\mathrm{CP}(\lambda)$, controlled NOT gate CX, and SWAP gate S gate. Here, λ denotes a real parameter.

The results are shown in Fig. 27. Each circle represents a marginal contribution with a size that monotonically depends on the corresponding probability (i.e., the larger, the more probable). In addition, we also show the mean, Eq. (8), and one standard deviation, Eq. (12), for each index i. For the sake of simplicity, we omit values with $p < 0.001$. Furthermore, we renormalize the probabilities to one for $r = 3$ (which are not normalized due to the subsampling).

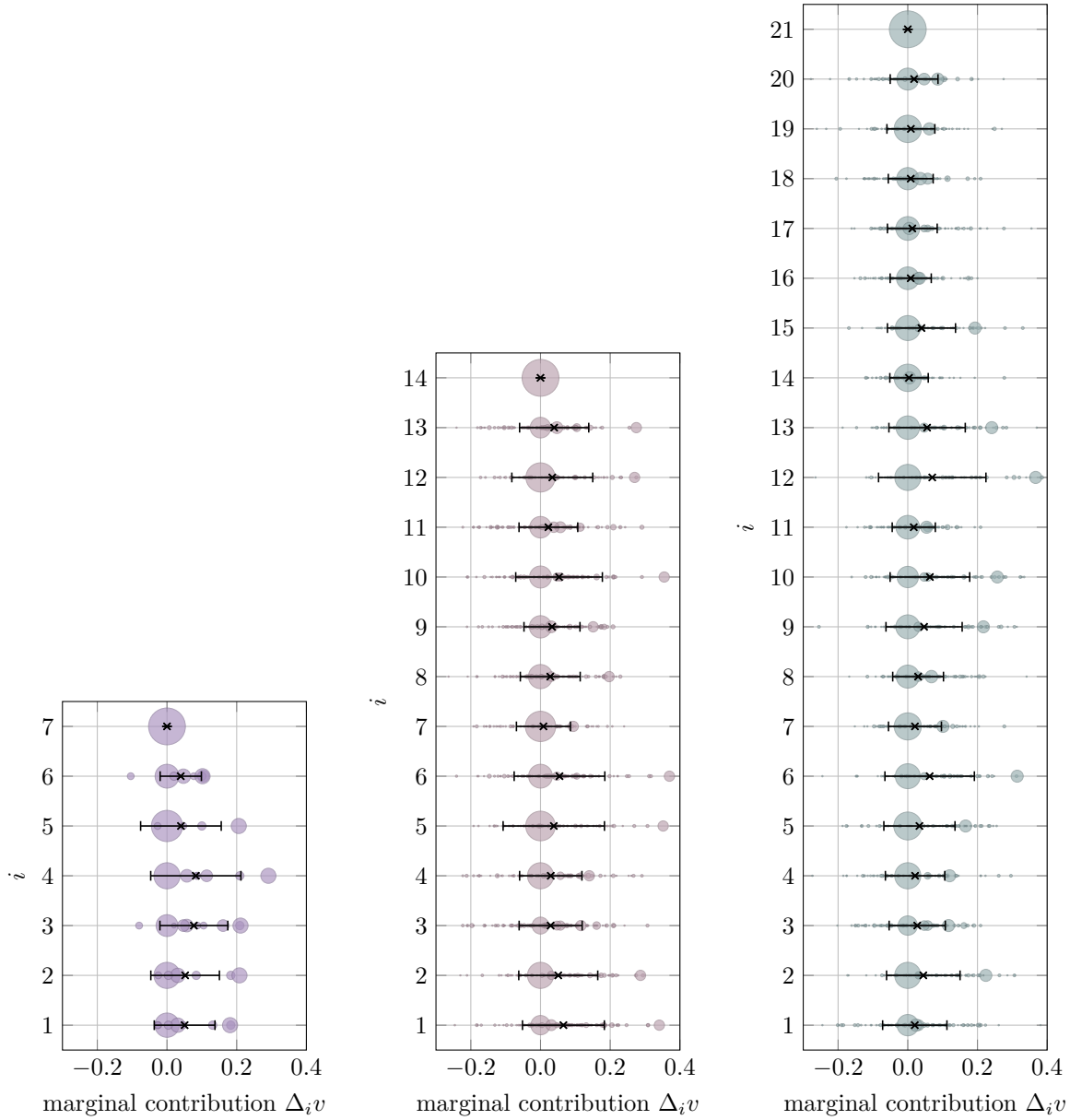
By definition, the mean values from Fig. 27 correspond to the respective SVQXs from Fig. 6. We find that almost all SVQXs are within one standard deviation of the other SVQXs.

H QGAN timing of experiments

The start and end times for the QGAN experiments from Sec. 4.3 are presented in Tab. 2. Here, the configurations and runs refer to the results shown in Fig. 17.

I QAOA optimization results

The optimization results for the max-cut problem from Sec. 4.5 are listed in Tab. 3. Specifically, we consider three independent runs for different circuit depths $r \in \{1, \ldots, 7\}$, each with a different random seed for the classical optimizer. The global optimum has a value of 9.



(a) $r = 1$ **(b)** $r = 2$ **(c)** $r = 3$

Figure 27: Distribution of marginal contributions $p_i(\Delta_i v)$, Eq. (9), for the QSVM use case from Sec. 4.1 with repetition numbers $r \in \{1, 2, 3\}$. For each player index i, we show the marginal contributions as circles with a size that monotonically depends on the corresponding probability. All evaluations are based on the `state_CPU` simulator with $K = 1$. We use $\alpha = 1$ for $r \in \{1, 2\}$ and $\alpha = 0.01$ for $r = 3$. For each player index i, we show the mean, Eq. (8), and one standard deviation, Eq. (12).

Table 2: Timing of the runs from Fig. 17. For each experiment and each run, we present the start time, the end time and the number of hourly collected error samples between start and end time as described in Sec. 4.3. Each sample consists of a single gate error and a CX error, respectively, for each qubit or qubit pair of interest. The time intervals include waiting times and the execution of MEM. All dates are in the format *YYYY-MM-DD HH:MM:SS*.

configuration	run	start time	end time	samples
0-1-2	1	2022-12-10 16:01:01	2022-12-11 19:12:15	28
	2	2022-12-11 19:12:15	2022-12-12 18:54:38	24
	3	2022-12-12 18:54:39	2022-12-13 23:51:04	29
0-1-2 with MEM	1	2022-12-10 16:01:21	2022-12-11 20:23:36	29
	2	2022-12-11 20:23:36	2022-12-13 07:14:06	35
	3	2022-12-13 07:14:07	2022-12-14 11:24:05	29
3-5-4	1	2022-12-21 12:39:12	2022-12-23 02:51:03	39
	2	2022-12-23 02:51:03	2022-12-24 22:27:32	44
	3	2022-12-24 22:27:33	2022-12-26 05:47:31	32
3-5-4 with MEM	1	2022-12-21 12:40:53	2022-12-23 08:54:24	45
	2	2022-12-23 08:54:24	2022-12-25 03:22:59	43
	3	2022-12-25 03:22:59	2022-12-26 07:15:21	28

Table 3: Optimization goals (i.e., number of cuts in the graph from Fig. 21) resulting from QAOA for different depths r with three independent runs per depth. The global optimum is highlighted in bold. All calculations are performed on a `state_CPU` simulator.

r	optimization goal		
	run 1	run 2	run 3
1	**9**	**9**	6
2	**9**	**9**	6
3	6	**9**	**9**
4	**9**	**9**	**9**
5	**9**	**9**	**9**
6	**9**	**9**	**9**
7	**9**	**9**	**9**

Knowledge-Based Systems 267 (2023) 110413

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Explainable quantum clustering method to model medical data

Shradha Deshmukh [a], Bikash K. Behera [b], Preeti Mulay [a], Emad A. Ahmed [c], Saif Al-Kuwari [d], Prayag Tiwari [e,*], Ahmed Farouk [f,*]

[a] Symbiosis Institute of Technology, Symbiosis International (Deemed University), Pune, India
[b] Bikash's Quantum (OPC) Pvt. Ltd., Mohanpur, WB, 741246, India
[c] Department of Computer Science, Faculty of Computers and Information, South Valley University, Qena, Egypt
[d] College of Science and Engineering, Hamad Bin Khalifa University, Qatar Foundation, Doha, Qatar
[e] School of Information Technology, Halmstad University, Sweden
[f] Department of Computer Science, Faculty of Computers and Artificial Intelligence, South Valley University, Hurghada, Egypt

ARTICLE INFO

Article history:
Received 4 September 2022
Received in revised form 20 January 2023
Accepted 18 February 2023
Available online 23 February 2023

Dataset link: https://github.com/shradha-deshmukh/Explainable-Quantum-Clustering.git

Keywords:
Quantum clustering
Quantum machine learning
Quantum computing
qk-means algorithm
Explainable AI
LIME

ABSTRACT

Medical experts are often skeptical of data-driven models due to the lack of their explainability. Several experimental studies commence with wide-ranging unsupervised learning and precisely with clustering to obtain existing patterns without prior knowledge of newly acquired data. Explainable Artificial Intelligence (XAI) increases the trust between virtual assistance by Machine Learning models and medical experts. Awareness about how data is analyzed and what factors are considered during the decision-making process can be confidently answered with the help of XAI. In this paper, we introduce an improved hybrid classical-quantum clustering (improved qk-means algorithm) approach with the additional explainable method. The proposed model uses learning strategies such as the Local Interpretable Model-agnostic Explanations (LIME) method and improved quantum k-means (*qk*-means) algorithm to diagnose abnormal activities based on breast cancer images and Knee Magnetic Resonance Imaging (MRI) datasets to generate an explanation of the predictions. Compared with existing algorithms, the clustering accuracy of the generated clusters increases trust in the model-generated explanations. In practice, the experiment uses 600 breast cancer (BC) patient records with seven features and 510 knee MRI records with five features. The result shows that the improved hybrid approach outperforms the classical one in the BC and Knee MRI datasets.

1. Introduction

Artificial Intelligence (AI) has achieved widespread success in various fields. A tremendous amount of data is generated daily to monitor routine tasks or help humans to improve their life. Deep learning is a significant factor in developing knowledgeable learning from the generated data. Managing complex data containing numerous features is increasingly time-consuming. Hence, models are very often used as black boxes that process input data and output predictions or decisions. This black box approach leads to non-transparent models and it can not be explained how the conclusions or predictions are made. The black box characteristics create problems while presenting predictions or making decisions. These decisions must have some explanations which show the transparent method of predictions. In the endeavor to address non-transparency, Explainable AI (XAI) comes into the picture and is also known as interpretable machine learning (ML) [1]. Several explainable and interpretable methods have been proposed in recent years. A model-based or post-hoc method is discussed and used in [2]. For instance, CT scan data is fed to the trained AI model for explainable prediction of lung cancer, presented in [3]. The XAI helps to determine why a particular CT scan was classified as lung cancer. XAI also explains how the model works and what factors are important in an overall manner. XAI methods focus on global and local explainability to see the relevant features and the decision probability.

1.1. Categorization in explainable AI

XAI is an add-on option when using ML algorithms to generate human understandable decisions [4], but it is rapidly gaining popularity in many applications [5–8]. A lot of techniques have been created for supervised algorithms, but since clustering cannot identify the causes of cluster formation, less work has been done to explain unsupervised procedures. Finding the inherent

* Corresponding authors.
E-mail addresses: shradha.deshmukh.phd2020@sitpune.edu.in (S. Deshmukh), bikas.riki@gmail.com (B.K. Behera), preeti.mulay@sitpune.edu.in (P. Mulay), emad.amer@sci.svu.edu.eg (E.A. Ahmed), smalkuwari@hbku.edu.qa (S. Al-Kuwari), prayag.tiwari@ieee.org (P. Tiwari), ahmed.farouk@sci.svu.edu.eg (A. Farouk).

https://doi.org/10.1016/j.knosys.2023.110413
0950-7051/

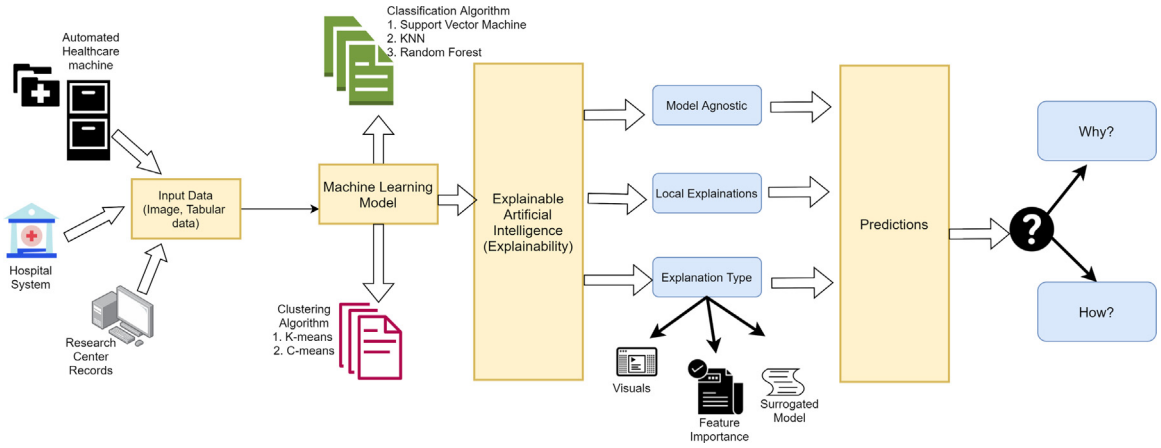


Fig. 1. The overview shows the procedures XAI uses to reach at accurate outcomes produced by the ML model. The input data (all the data types) is accepted in the first stage, after which the appropriate ML model is used to see the outcomes or predictions. The answers to queries like "How and why the predictions are made" are offered by the explanations given by the XAI.

groupings contained within the data is performed using the unsupervised clustering technique. However, it may be challenging to understand why a single row of data is grouped in a certain cluster when clusters are produced using methods like the k-means algorithm. The model can utilize this insight to shift data points from one cluster to a more lucrative cluster by being aware of these boundary constraints for transition from one cluster to another. For decision-makers, explainable clustering is typically advantageous. Some methods can be used to interpret the clustering outcomes of any unsupervised approaches because it is model agnostic [9,10] (which means that these methods can be applied to any model). The classification algorithms like Support Vector Machine (SVM), Random Forest, and K Nearest Neighbor (KNN) are used to categorize the acquired cluster labels and the associated datasets.

Regarding the scope of the provided explanation, we can categorize the methods into global and local approaches. Scope refers to either aiming to explain the whole model or just parts of the model, such as individual predictions. To explain, let us recall that the decision boundary of a complex model can be highly non-linear. For instance, we have a classification problem here, and only a complex function can separate the two classes. Sometimes it cannot explain the global model; instead, many approaches assume a local area. Then we need to explain the individual predictions made at the specified decision boundary. Fig. 1 further differentiates according to the data type a method can handle. A few explainability algorithms can work with all types of data [11]. Finally, the models produce different explanations, starting with correlation plots or other visual methods. Also, obtaining information about the feature's importance is called a feature summary. Other methods return data points that help us understand the model, and some existing approaches build a simple model around the complex one. A simple model, which are usually called surrogates, can then be used to derive explanations. An overview of the XAI model's workflow is shown in Fig. 1, where one of the levels offers the specifics of local explanations, explanation kinds to be examined, and model agnostic categories. The final stage of the XAI model answers the queries of why the model makes a particular prediction and how the prediction is calculated.

In medical or healthcare domain, ML learns from a large amount of data and predicts more accurate results than the average clinicians. With this motivation, the aim is to reduce the accuracy gap while increasing the trust. Sometimes doctors have natural biases, so to overcome this problem, AI produces the objective diagnosis with preconceived socio-economic notions, which ultimately produces trustworthy relations between the parties, such as AI–doctor relations and AI–patient relations.

1.2. Challenges in medical data

There are a number of challenges in the healthcare related to the medical data [12–14], such as:

- Many medical image datasets are unavailable due to privacy concerns.
- Sometimes incorrect labels come up with wrong predictions.
- In some cases, features are strongly correlated, and most training data are uninformative and redundant.
- It is not easy to convert image data into pixel numeric data.

XAI has many challenges and associated issues, as discussed in the context of standard workflow shown in Fig. 2 [15,16]. Designing domain-agnostic systems with XAI is challenging as explanations require the context of rooming. Some explanations can be helpful for the targeted perspective but trivial for others. For instance, presenting interactive visualizations to explain layers of a Neural Network (NN) is beneficial for data scientists but less critical to radiologists who use the NN for data analysis. Integrating XAI in a standard workflow is challenging but it benefits from multiple perspectives [17]. The scope of XAI is broader than the traditional ML model [18]. The explainability of the underlying data and preprocessing methods, such as feature selection utilized for input data preparation, are important issues need to be addressed. Furthermore, if the underlying training data contains records from comparable demographics, the ML models used to forecast a disease based on a patient's medical records may not be acceptable. Another significant obstacle to XAI is its subjective applicability. Healthcare facilities, for instance, have varying degrees of access to medical data in the real world. It is crucial to comprehend, how a single model developed for a certain disease's diagnosis performs in multiple contexts. We suggest a simple and efficient incremental learning method to provide the first steps toward problem-solving and enable a level of explainability based on the recognized collection of linked difficulties.

The feature importance process is shown in Fig. 2 and relates to methods that rate input characteristics according to how well they can predict a target variable. Statistical correlation scores, coefficients derived as part of linear models, decision trees, and permutation importance scores are a few common examples of feature importance scores. At the same time, many other types

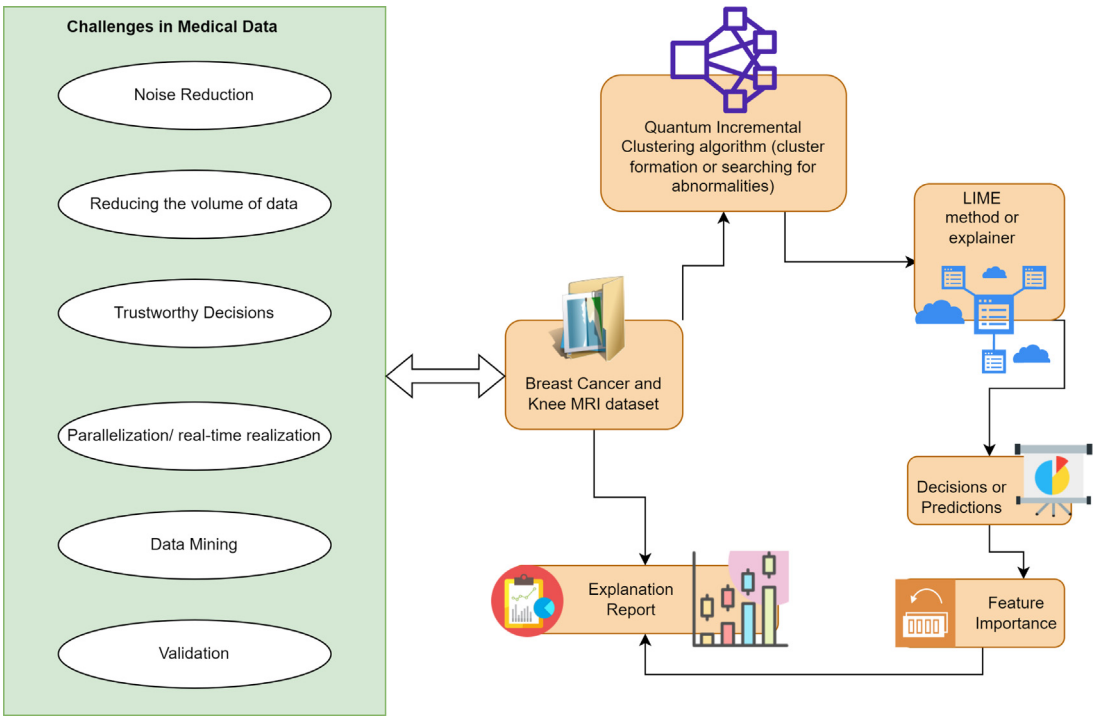


Fig. 2. Overview of the integration of XAI and machine learning models. The data and model come up with high-level explainability of data and then finally show the predictions.

and sources are also discussed in [19]. In predictive modeling, feature importance scores are crucial because they offer insight into the data and the model. Additionally, statistical correlation scores are applied to boost the effectiveness and efficiency of a predictive model. The most popular method for determining if two features are statistically connected is the t-test [20]. The t-test identifies the most important features from the set of features, then the t distribution is used to determine the correct t value, which is in turn used to determine the significance of a correlation coefficient (See Eq. (1)). By assuming no association between them (Sigma= 0), the t value can be computed as follows:

$$t = m\sqrt{\frac{n-2}{1-m^2}} \tag{1}$$

where m is the correlation coefficient for the sample data and n is the total number of data points. Probability levels (p) are used to express the relationship's importance (i.e., the significance at $p = 0.05$). There are n - 2 degrees of freedom when entering the t-distribution. The feature is chosen if the t value exceeds the crucial value at the 0.05 significant level, indicating that it is significant.

Fig. 2 illustrates the proposed steps to deal with the challenges in medical data. The quantum clustering algorithm clusters the data points with matching characteristics, and then the Local Interpretable Model-Agnostic Explanations (LIME) explainer shows the description of the decision or the important feature for the predictions. The detailed report generated from the ML models shows the explanation and maintains the clinical experts' trust.

1.3. Contributions

Our contributions can be summarized as follows:

(1) We propose hybrid approach, i.e. explainability and improved quantum clustering algorithm, for generating the best medical report prediction explanations.
(2) We encode the classical data into quantum states and form the clusters using the improved *qk*-means algorithm, with the help of the $UU^{\dagger}$ quantum circuit.
(3) We show that our proposed model works well in terms of the overall clustering accuracy of the breast cancer and Knee MRI dataset.
(4) We discuss the effectiveness of the improved *qk*-means algorithm with the experimental results obtained from the IBM quantum simulator, i.e. qasm simulator, and show that the hybrid approach outperforms the classical one.
(5) We integrate LIME with the improved *qk*-means algorithm for trustworthy decisions.

1.4. Organization

The rest of this paper is organized as follows: Section 2 highlights the medical literature, how explainability is used to provide expert predictions, and a brief about QML in medical data. Section 3 describes the proposed methodology of the hybrid classical quantum *k*-means algorithm and the implementation of the improved *qk*-means algorithm with LIME. Section 4's experimental result illustrates the evaluation environment, used datasets, and quantitative visualization. The results of explainability and hybrid approach are discussed in Section 5. Finally, Section 6 concludes the paper.

2. Related works

2.1. Explainability in the medical domain

In healthcare, human lives are at stake; therefore, it becomes vital for clinicians to understand the operation of the models and the prediction they make. XAI models help in the following ways:

- Supports in model improvement
- Merge the layer of confidence in the prediction made by the AI models

Furthermore, explanations behind inaccurate predictions can help trace the factors that cause them and improve the model for future use. The medical data field has different applications, such as detection, diagnosis, and treatment plan [21]. XAI assists in diagnosing diseases such as neurology, cancer, and cardiology [22,23]. In the radiology domain, the author explains how to identify the radiographs with a decision to maintain transparency [24]. In the pathology domain, the paper introduces the techniques to recognize the patterns for pathologists to speed up the diagnostic process [25].

Creating the step-wise framework of the ML pipeline process, which includes pre-processing methods, data manipulation, and explainability results, reduces the bias prediction and promotes fairness of the ML model. The pertinent input data from the human records helps to optimize the ML pipeline to obtain effective results to participate in the decision-making process.

2.2. Classical to quantum machine learning in medical data

Technological concerns have dominated the research of ML techniques for medical applications. However, it is crucial to improve the understanding of ML algorithms and to offer mathematical reasoning for their attributes in order to get helpful insight into the performance and behavior of ML approaches in a medical context. The diagnostic accuracy of a patient is traditionally based on a doctor's experience, although this knowledge is gained over many years of observing many patients' symptoms and receiving a verified diagnosis. Even then, accuracy cannot be guaranteed. A large amount of data can now be acquired and stored easily because of the development of computing technologies, as seen in the specialized databases of electronic patient records [26]. By using classification approaches to identify breast cancer patients (differentiate between benign and malignant tumors) to predict prognosis, ML techniques play an important role in the diagnosis and prognosis of breast cancer [15]. The most suitable treatment plan can be prescribed by professionals with the help of accurate classification. Table 2 shows some of the already existing works on the breast cancer dataset. Logistic regression, linear regression, and discriminant analysis are examples of traditional statistical methods for data analysis. To show the benefits of ML and its potential, a number of research works contrasting various classic statistical approaches with classical ML classification methods have been published.

Quantum machine learning (QML) is a new and promising paradigm in the domain of science and technology on today's edge as quantum data science is becoming mainstream [27]. QML fills the gap between theoretical advances in quantum computing and deployed ML science if it focuses on offering synthesis that describes more relevant ML algorithms in a quantum framework reducing the complexity of the disciplines involved. Quantum computing is revolutionalizing the way that we approach ML. This can be achieved by leveraging quantum effects such as entanglement and interference to solve currently unsolvable problems in ML. For example, the quantum kernel method has been shown to be effective as compared to other kernels in the presence of high noise [28]. The hybrid classical-quantum neural network is proposed for healthcare-based intrusion detection systems using quantum circuits [29]. The quantum kernelized binary classifiers are used to simplify the quantum kernel support vector machine and applied to clinical data. The quantum distance classifier technique explains in the literature where authors show the distance calculation using the Hadamard gate and SWAP test [30]. The following formula is used to calculate the distance between two vectors (Eq. (2)). The task is to calculate the $\langle u|v\rangle$ on a quantum computer.

$$|u\rangle - |v\rangle = \det\left||u\rangle|v\rangle\right| - \langle u|v\rangle \tag{2}$$

2.3. Explainability and clustering in medical data

The literature [39] explains multilevel clustering to build explainable data visual mapping. The explained method shows the clustering performed by the external agnostic algorithm on selected parameters to form clusters. With the help of such a method, exploratory information is provided to take knowledge-based actions. The described method successfully explains the accurate labels of the patient report [40]. The literature [41] presents the framework for XAI, that applies the ensemble clustering method to brain injury data. This accommodates the Traumatic Brain data using the ensemble method and presents an effective and an explainable report. Table 1 lists current XAI studies along with specifics on their techniques and targeted datasets.

3. Methodology

3.1. Classical and quantum k-means algorithm

The clustering is an unsupervised learning problem, where input data has no labels and the goal is to form meaningful subgroups with similar features from the data. The classical k-means approach finds the nearest data points of the selected clusters and forms a quality cluster to analyze the unseen data. The classical approach uses the random cluster's centroid, and the radius of the cluster is not defined in the initial phase. The detection of similar featured data points using the Euclidean distance calculation is the standard process used in most literature [42]. The classical approach quickly becomes inefficient when it comes to high-dimensional datasets. The allocation of data points is complex when data carrying the n number of dimensions. The classical-quantum approach uses the quantum clustering method to deal with the n-dimensional data, and provides strong theoretical base to deal with n dimensional data. Here, qk-means algorithm is also called as the hybrid classical-quantum method. Quantum clustering, the distance calculation between the data points and centroids, is exhibited by the probabilistic nature of quantum states. The quantum approach quantifies different probabilities' amplitudes to provide outputs from the classical inputs.

3.2. Dataset: breast cancer and knee MRI datset

In this paper, two different datasets i.e. Breast cancer and Knee MRI are used to discuss the performance of the quantum clustering algorithm.

Systems are increasingly combining relational tabular data with machine learning to intelligently supplement workflows and processes that are typically handled by people. Tabular data is the most frequent data type in business and the second most frequent format in academics, according to a recent poll by the data science platform KAGGLE [43]. For the cluster analysis of tabular datasets, a parallel k-means algorithm implementation that is massively parallel and scalable was created [44]. A logical conjunction of events specifies a generalized representation of patterns, or symbolic object [45]. These events connect values and features to express the data's abstract pattern. According to recent studies, feature extraction continues to put more emphasis on converting the data into a quantifiable data type than it does on identifying fresh patterns to describe the data [46]. In ML, researchers have begun to integrate clustering techniques and classifier models in order to minimize the quantitative training set dimension. A clustering approach is utilized [47,48] to group comparable tabular terms in order to facilitate quicker and more precise training on the job of quantitative data categorization. The fuzzy C-means clustering technique and a K-means version, are

Table 1
XAI methods on medical data.

Sr. no.	Article title	Year	Datasets	Methodology
1.	"ExMed: An AI Tool for Experimenting XAI Techniques on Medical Data Analytics" [31]	2021	Simulacrum Dataset	Explainability for prediction model using the feature attribution model
2.	"Diagnosis support model of cardiomegaly based on CNN using ResNet and explainable feature map" [32]	2021	X-ray dataset	Convolution neural network based ResNet for detection the cardiomegaly diagnosis.
3.	"Explainable Artificial Intelligence for Human Decision Support System in the Medical Domain" [33]	2021	Image data from Video Capsule Endoscopy	Local Interpretable Model-Agnostic Explanations (LIME) and Shapley Additive exPlanations are used to produce the explanations. The output description is evaluated by the professionals.
4.	"Applications of Explainable Artificial Intelligence in Diagnosis and Surgery" [15]	2022	Tomography Image dataset and MRI dataset	Summaries the latest explainability methods applied on Medical data.
5.	"Explainable AI for Glaucoma prediction analysis to understand risk factors in treatment planning" [16]	2022	Coherence tomography eyes image data and clinical data of glaucoma patients	To process data author uses spike neural network (SNN) and adaptive neuro-fuzzy inference systems.

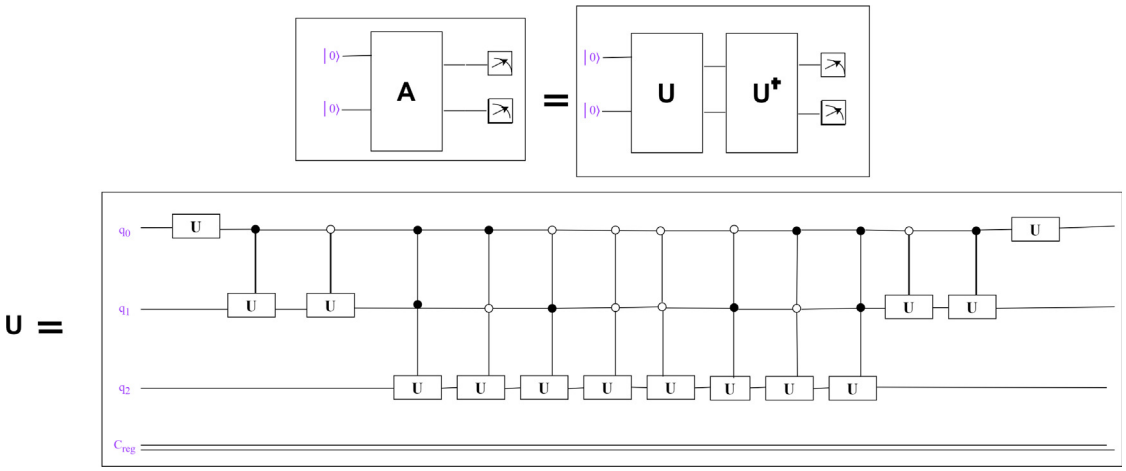


Fig. 3. The quantum circuit for $UU^{\dagger}$ gate for distance calculation in improved *qk*-means clustering algorithm. The circuit is a three-qubit system to show the various features of the input dataset.

Table 2
Performance measure of a classical learning model and classical-quantum learning model on the Breast cancer data (Dataset 1) and Knee MRI data (Dataset 2).

Models and sources	Dataset	Precision	Recall	F1 measure
Classical Classification Algorithms				
KNN [34,35]	Dataset 1	0.904	0.917	0.911
KNN [34,35]	Dataset 2	0.931	0.926	0.915
Random Forest [36]	Dataset 1	0.905	0.917	0.923
Random Forest [36]	Dataset 2	0.908	0.925	0.919
SVM [37]	Dataset 1	0.898	0.885	0.891
SVM [37]	Dataset 2	0.888	0.865	0.832
Logistic Regression [38]	Dataset 1	0.887	0.864	0.891
Logistic Regression [38]	Dataset 2	0.878	0.855	0.889

developed for the SVM classifier training set to explore hidden patterns.

The breast cancer dataset has 600 attributes or patient records and 7 features, whereas the second dataset contains 1370 knee MRI data from Stanford University Medical Center. However, we used 510 Knee MRI data with 5 features for the experiment. Our proposed improved quantum clustering algorithm clusters the MRI data into three cluster labels: anterior cruciate ligament (ACL), meniscal tears, and abnormalities. (Fig. 5).

Encoding Procedure: The basic procedure to encode the classical data points into the quantum states as discussed in [28,49]. The proposed method uses the amplitude encoding technique to encode the selected input data, so the classical data is encoded in quantum state amplitudes.

$$|X\rangle = \sum_{i=2}^{2^n} x_i |i\rangle \tag{3}$$

In Eq. (3), the elements of the amplitude vector are converted into computational basis states. Now the quantum model is integrated with the encoded quantum data and is ready to perform the clustering. Existing literature [50,51] shows that a significant speed-up using the Grover's search algorithm can be obtained, but the quality of clusters is unsatisfactory compared to the classical clustering algorithm.

3.3. Implementation of improved qk-means algorithm

In both datasets, the cluster centroid is allocated, and then the qk-means algorithm is used to find the cluster among them.

Table 3
Results collected from the different clustering metrics are presented for Dataset 1: Breast Cancer dataset, Dataset 2: Knee MRI dataset. In the table, the abbreviations are denoted as, ACC: Accuracy, Com: Completeness, Sils: Silhouette Score.

Models and sources	Dataset	Acc	Com	SilS
Classical Clustering Algorithms				
Classical K-means Algorithm [52]	Dataset 1	0.881	0.872	0.181
Classical K-means Algorithm [52]	Dataset 2	0.873	0.861	0.292
C-means [53]	Dataset 1	0.875	0.861	0.118
C-means [53]	Dataset 2	0.892	0.885	0.221
Classical-Quantum Clustering Algorithms				
qk-means [54]	Dataset 1	0.880	0.897	0.211
qk-means [54]	Dataset 2	0.891	0.881	0.194
Improved *qk*-means (Proposed)	Dataset 1	0.926	0.891	0.333
Improved *qk*-means (Proposed)	Dataset 2	0.937	0.932	0.301

Algorithm 1: Improved *qk*-means algorithm.

Input Classical data
Output Clusters of data
1 Assuming input data points $X_1, X_2, ...X_n$, and k number of clusters, we have $|X_1\rangle, |X_2\rangle, |X_3\rangle, ..., |X_n\rangle$ as the input quantum states, and $|C_1\rangle, |C_2\rangle, |C_3\rangle, ..., |C_k\rangle$ as the centroids of the clusters $C_1, C_2, C_3, ..., C_k$.
2 Initial stage $|0\rangle|X_i\rangle|C_i\rangle$
3 Calculate distance from one data point (X_i) to centroid (C_i) using the formula, $\langle X_i|C_i\rangle = \langle 00|U_1U^\dagger|00\rangle = \sqrt{P_0}$
4 Assign cluster to the data point. Observe the nearest distance among $d^2(X_i, C_i)$ and perform superposition on data points. $|X_i\rangle \otimes |C_i\rangle|d^2(X_i, C_i)\rangle$
5 Perform Step 2 and Step 3 again until all the input quantum states are converged.

Suppose $|X\rangle = |X_1\rangle, \ldots |X_n\rangle$ are the datapoints encoded in the quantum states. k denotes the number of clusters and $|C\rangle = |C_1\rangle, \ldots, |C_k\rangle$ are the cluster centroids encoded in the quantum states. To minimize the cluster size within-cluster variance, $|W\rangle = |W_1\rangle, \ldots, |W_k\rangle$ is used.

Our improved hybrid classical-quantum (improved *qk*-means) approach uses the $UU^\dagger$ method (See Fig. 3 and algorithm steps in Algo. 1) to compute the distance between data points (input vector) and cluster centroids (centroids = 2), and then the data point is assigned to the nearest centroid. We note that the initial state is chosen to be $|0\rangle$ because, at the end of the algorithm, the probability of $|0\rangle$ is obtained, whose square root equals to the inner product of the centroid and test data. If the initial state is kept to $|1\rangle$, the probability of $|1\rangle$ needs to be measured, whose square root does not equal to the inner product. For example, $\langle 0|U^\dagger U|0\rangle = \langle 0|A|0\rangle = \sqrt{P_0}$, to achieve this, we need to first apply A on $|0\rangle$ state, then an arbitrary state can occur, $a|0\rangle + b|1\rangle$, in this case, if we measure P_0, its square root would be equal to a, which is $\langle 0|A|0\rangle$. However, if we prepare the initial state to be $|1\rangle$, and measure P_1, the square root of P_1, will not always be b, as b can contain some phase information, which P_1 cannot tell about. Here, we assume that a is always a real number because if it contains any phase information, it can be taken as common from both the terms and considered as the global phase. Similarly, this case can be scaled to any n-qubit system, where the initial state should be taken as $|0\rangle^{\otimes n}$. The improved *qk*-means algorithm updates the cluster centroid in the second step with the help of Eqs. (4) and (5).

$$\underset{D}{\operatorname{argmin}}(|X_i\rangle, |W_k\rangle) = \sum_{i=1}^{k} \sum_{x \epsilon W_i} d_2(|X_n\rangle, |W_k\rangle) \tag{4}$$

$$\underset{D}{\operatorname{argmin}}(|X_i\rangle, |C_k\rangle) = argmin \sum_{k=1}^{k} d_2(|X_n\rangle, |C_k\rangle) \tag{5}$$

3.4. LIME method for explainability

The clusters are created by using the improved qk-means clustering technique. Furthermore, LIME is used for the prediction explanation [55]. It is used to describe cluster formation and the grouping of breast and knee datasets. When different types of data are fed into improved *qk*-means model, the LIME explainer extracts the explanations and provides a local explanation for the predictions. It then performs tests and demonstrates what happens to the predictions.

Our proposed model learns some complex patterns as a combination of the selected features. It is difficult to summarize the whole decision boundary into one explanation. The basic idea of explanations with improved qk-means is to zoom into the local area of the individual predictions with the help of LIME (See Algo. 2). In the local region, simple explanations make sense, so the rest of the model gets a valid explanation for why the prediction is made. The designed interpretable models, i.e. integration of LIME and improved qk-means to get explanations, fits a linear interpretable model in the local area. It is a local approximation of a complex model. The complex models are complete black boxes, and the internals are hidden for LIME, so it is based on a model's inputs and outputs. In our experiment, the input is tabular data of images. Usually, domain experts in a specific field have some prior knowledge about the problem. In the paper, we provide explanations to improve the acceptance of a predictive algorithm. For LIME, the requirement is that the explanations are locally faithful, but at the same time, explanations might not make sense globally. Hence, throughout this paper, the experiment focuses on the local area around the output predictions.

$$X = \underset{g \epsilon G}{\operatorname{argmin}} L(f, g, \pi_x) + \omega(g) \tag{6}$$

$$L(f, g, \pi_x) = \sum_{Z} \pi_x(z)(f(z) - g(z'))^2 \tag{7}$$

Now consider Eq. (7) for the mathematical optimization problem used in LIME. For instance, knowing the properties of breast cancer and MRI data in the tabular format which is the input data point X. In the optimization formula in [56], the complex model is denoted by f, and the linear model is denoted by g, which comes from the set of interpretable models, denoted by G (G is the family of a linear model). The loss function in optimization means looking for an approximation of the complex model f by the simple model g in the neighborhood of data point X (good approximation from a neighbor). The third argument π defines local neighborhoods of the selected data points and is some proximity measure. The final term is to regularize the complexity of the simple surrogate model. For the improved qk-means algorithm, a desirable condition could be to have many zero-weighted input features, which makes the explanation simpler when ignoring most of the features. Overall, ω is a complexity measure, and an optimization is a minimization problem, i.e., minimizing the complexity. The loss function says that the g argument in the *argmin* minimizes those two loss terms, so it should be an approximate complex model in that local area and stays as simple as possible. By achieving the good clustering accuracy, The complex model f and all input feature values minimizes the loss term. $f(z)$ is the label which comes from the imrpoved *qk*-means model and z' is the prediction model. The loss function, used for optimizing a model, is the sum of square distances between $f(z)$ and z'.

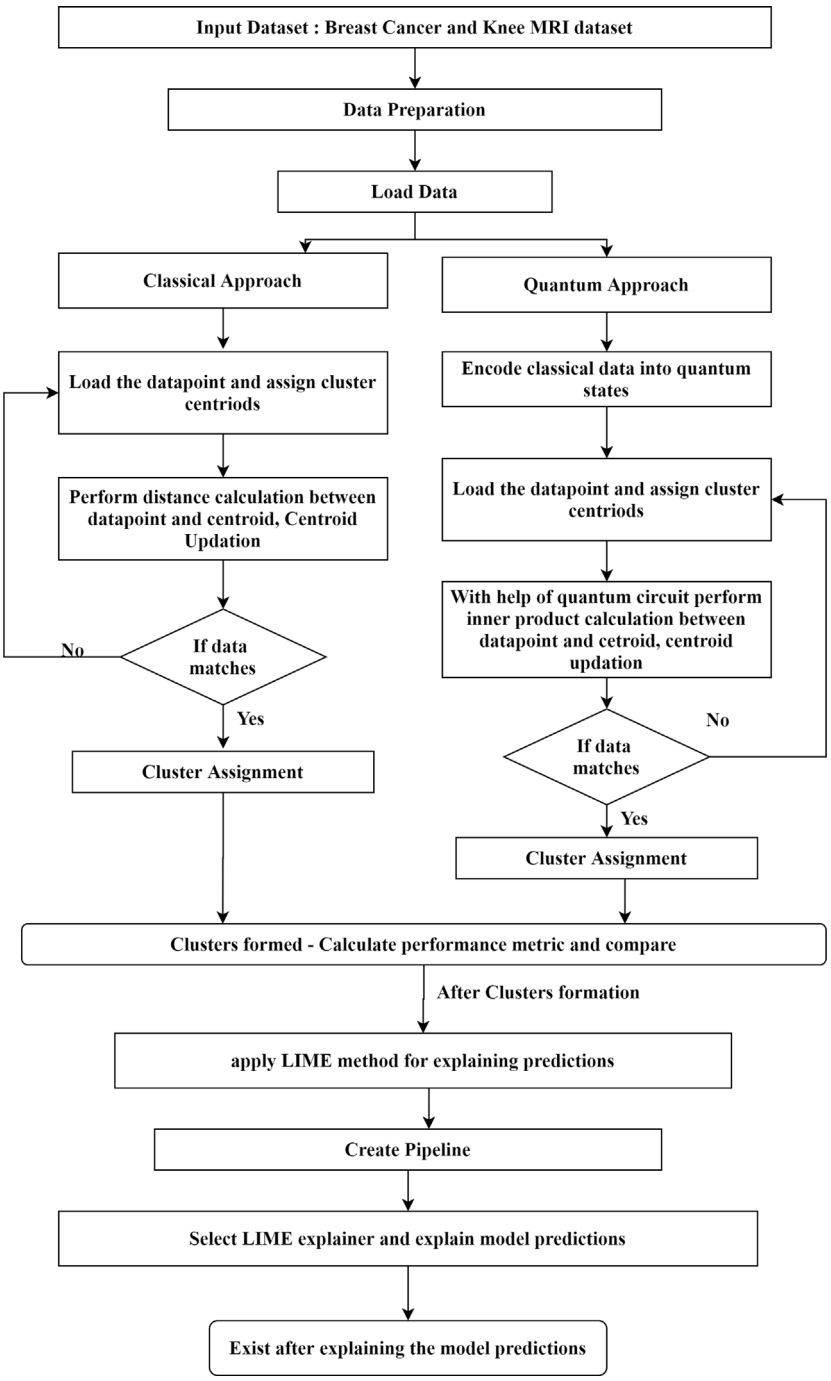


Fig. 4. The process flow of the hybrid approach using the quantum clustering of the labeling of the datasets and the LIME method to explain the prediction given by the improved *qk*-means clustering algorithm.

Fig. 4 illustrates the flow of the improved *qk*-means approach and classical approach for explaining the predictions. Therefore, the results from both approaches explain using the local explainer and provide the description of the prediction.

4. Experimental result

4.1. Evaluation environment and metric

The IBM quantum environment provides high-performance simulators for simulating various algorithms and quantum circuits under actual noise models. Experiments are performed on IBM quantum simulators, i.e. qasm simulator, which is based on superconducting transmon qubits to export measurement count. We have used the three-qubit quantum circuits to perform distance calculations and the centroid updation to maintain the cluster quality. Fig. 3 illustrates the quantum circuit used to show the improved *qk*-means algorithm evaluation.

The evaluation metric can be found as follows:

$$\text{Accuracy} = \sum_{X=x_i,..,x_n}^{c} = \frac{TP_X + TN_X}{TP_X + TN_X + FP_X + FN_X} \tag{8}$$

$$\text{Precision} = \sum_{X=x_i,..,x_n}^{c} = \frac{TP_X}{TP_X + FN_X} \tag{9}$$

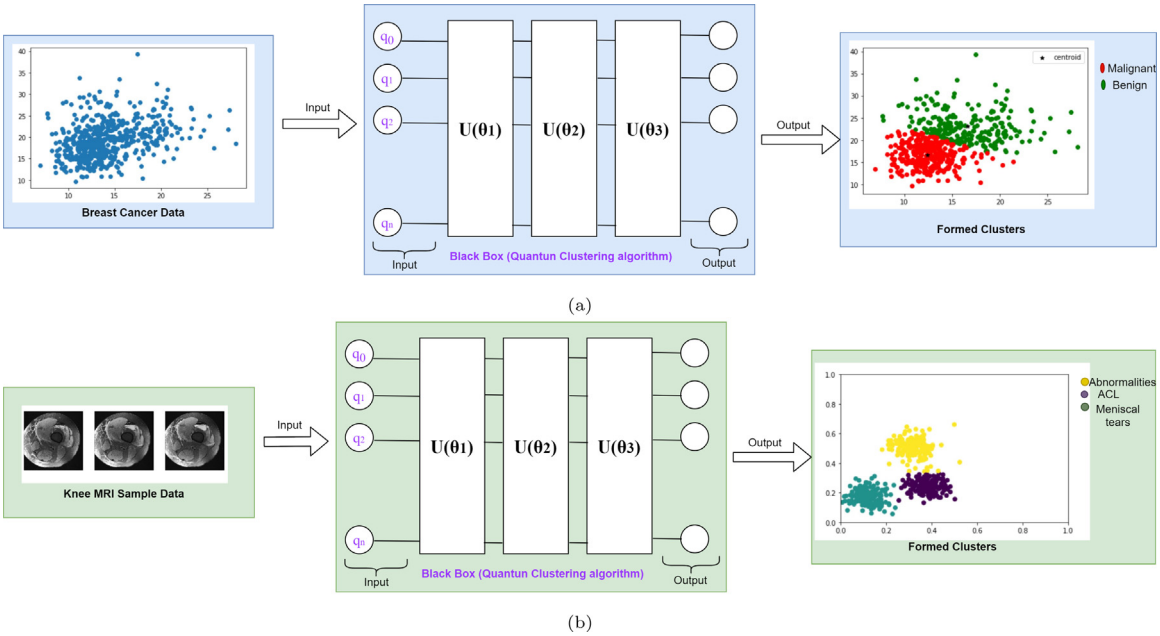


Fig. 5. (a) The clustering results given by the qk-means algorithm using the features of the Breast Cancer dataset. The scatter plot shows the two clusters representing benign and Malignancy. (b) The illustration shows that the three types of Knee series result from the quantum clustering algorithm. The predicted results form three clusters for ACL, meniscal tears, and abnormalities.

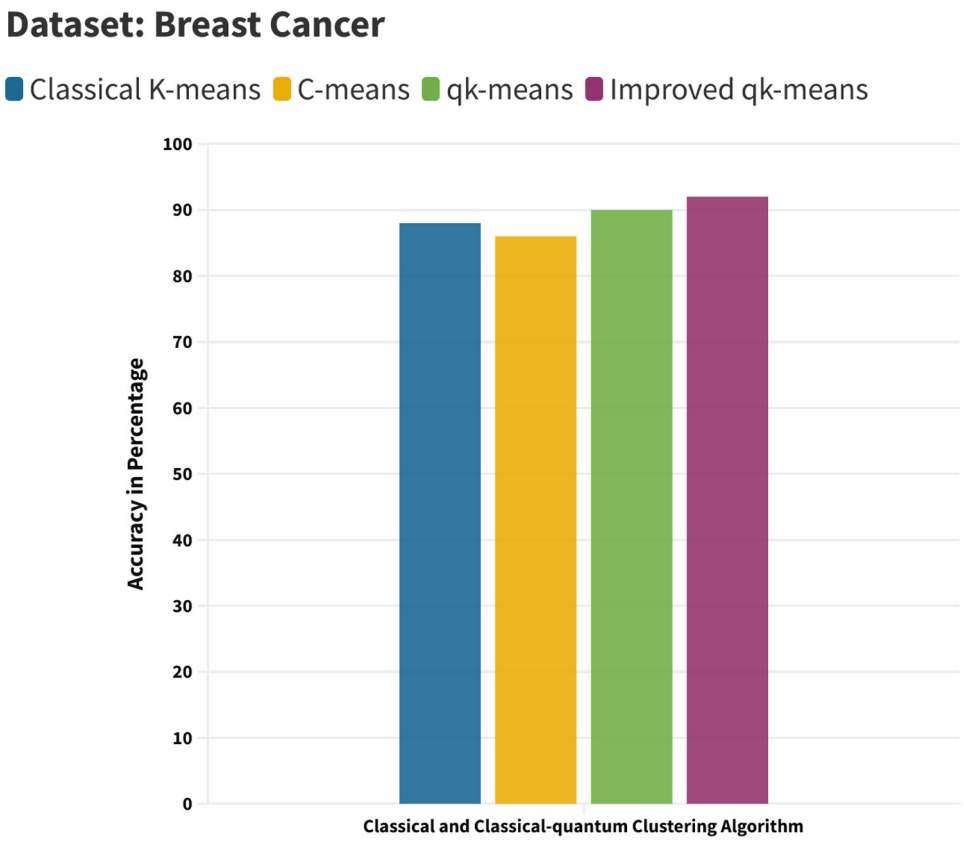


Fig. 6. Using the breast cancer dataset, the bar graph compares the accuracy of classical clustering algorithms like K-means and C-means and classical-quantum clustering algorithms like qk-means and our improved qk-means algorithm.

$$\text{Recall} = \sum_{X=x_i,\ldots,x_n}^{C} = \frac{TP_X + TN_X}{TP_X + FP_X} \tag{10}$$

$$TP_{X_i} = \sum_{i=1}^{n} X_n \tag{11}$$

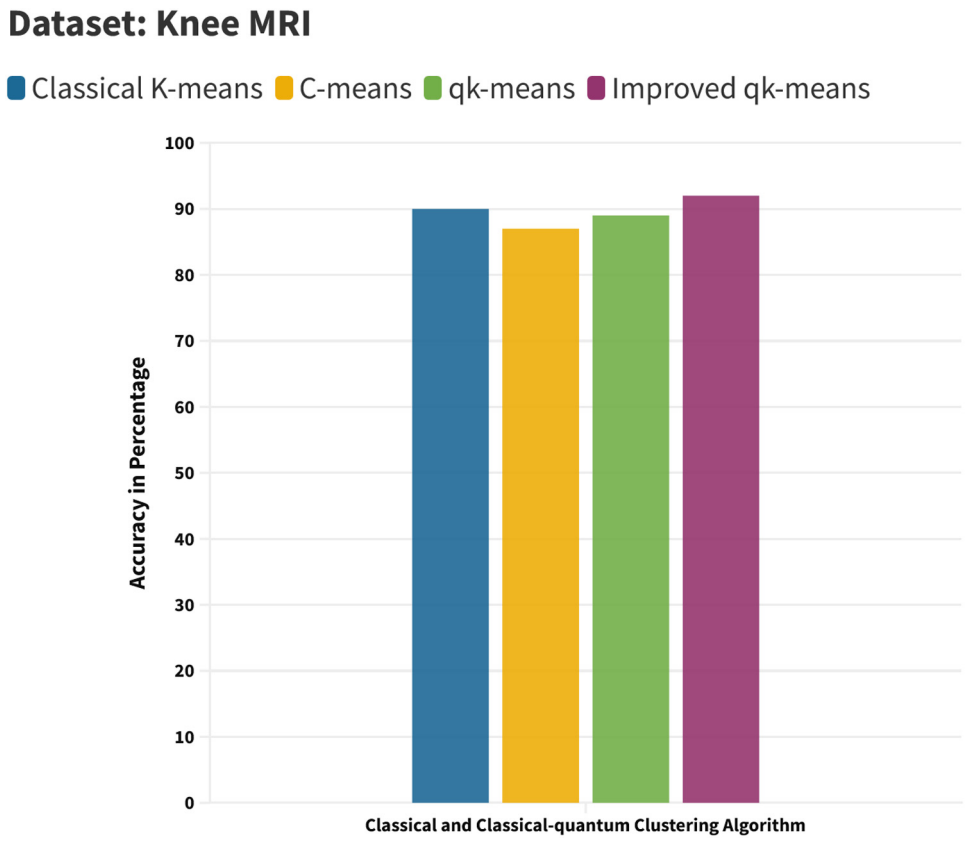


Fig. 7. The accuracy of the classical-quantum clustering methods, such as qk-means and our improved qk-means algorithm, and traditional classical clustering algorithms, such as K-means and C-means, using the Knee MRI dataset is presented in the bar graph.

Algorithm 2: Hybrid approach of LIME and Improved *qk*-means algorithm to explain the features

Input Classical data
Output Prediction and explanations of the data
1 Test data
2 Estimate distance between test data points and centroids
3 Make clusters and predictions on new data using the improved *qk*-means algorithm
4 Choose n number of features that best describes the quantum clustering model predictions from the test data.
5 Feature weights from the simple model make explanations for the model's local behavior.

$$TN_{X_i} = \sum_{j=1}^{c}\sum_{k=1}^{c} X_j k, j \neq i, k \neq i \tag{12}$$

$$FP_{X_i} = \sum_{j=1}^{c} X_j i, j \neq i \tag{13}$$

$$FN_{X_i} = \sum_{i=1}^{c} X_i j, j \neq i \tag{14}$$

$$Accuracy(C, C') = max_{perm \epsilon p} \frac{1}{n} \sum_{i=0}^{n-1} 1(perm(C_i') = C_i) \tag{15}$$

$$S = \sum_{i=0} \frac{b_i - a_i}{max(a_i * b_i)} \tag{16}$$

4.2. Quantitative and visualization results

We form the learning task by taking the image data, especially paper focused on a digitized image of a fine needle aspirate (FNA) of a breast mass (UCI repository). First, we train the model and then we use the learning model to produce two clusters as the following:

- Cluster of a malignancy
- Cluster of a benign

In our dataset, there is multicollinearity as the features (such as radius mean, area mean and parameter mean) depend on each other and affect the model's performance (See Fig. 5(a)). So we chose area mean as one of the features for calculating clusters. Furthermore, there are two independent features such as concavity and compactness, for which regression method can be used for the feature selection.

Both input datasets are clustered and classified using a variety of algorithms. Tables 2 and 3 show the existing ML algorithms, *qk*-means algorithm and improved *qk*-means algorithm. First, the classical clustering algorithms, which include the K-means [52], and C-means [53] are used. The classification methods, i.e. KNN [34,35], Random Forest [36], SVM [37], and Logistic regression [38], are then used to demonstrate comparison and locate gaps to discover any hidden patterns in the data. The evaluation metrics like precision, recall, and F-measures are used to measure the performance of classification methods. After discussing the results of classical clustering and classification methods, we employed the *qk*-means algorithm, and improved *qk*-means algorithm. We may infer from Table 3 that the clustering accuracy obtained from our improved qk-means algorithm is

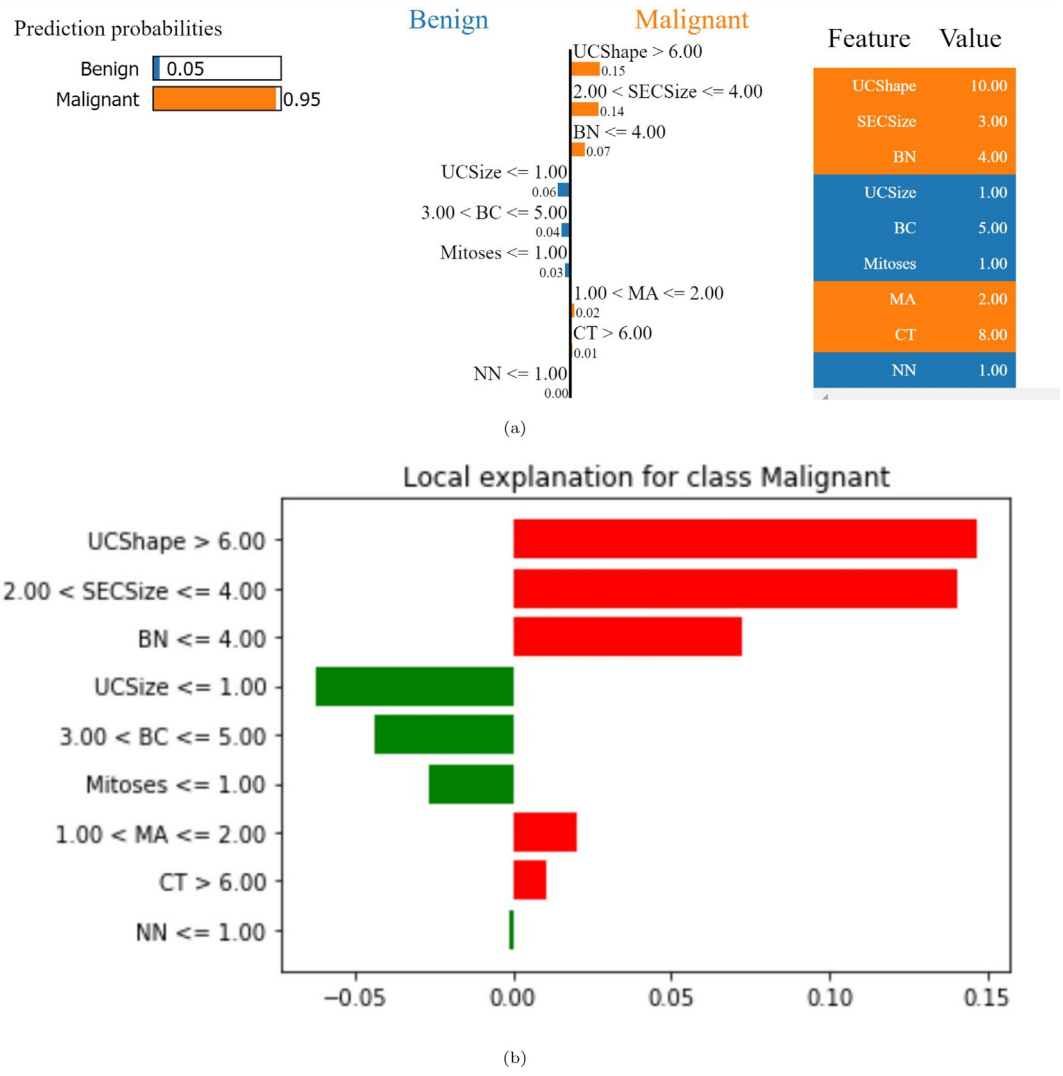
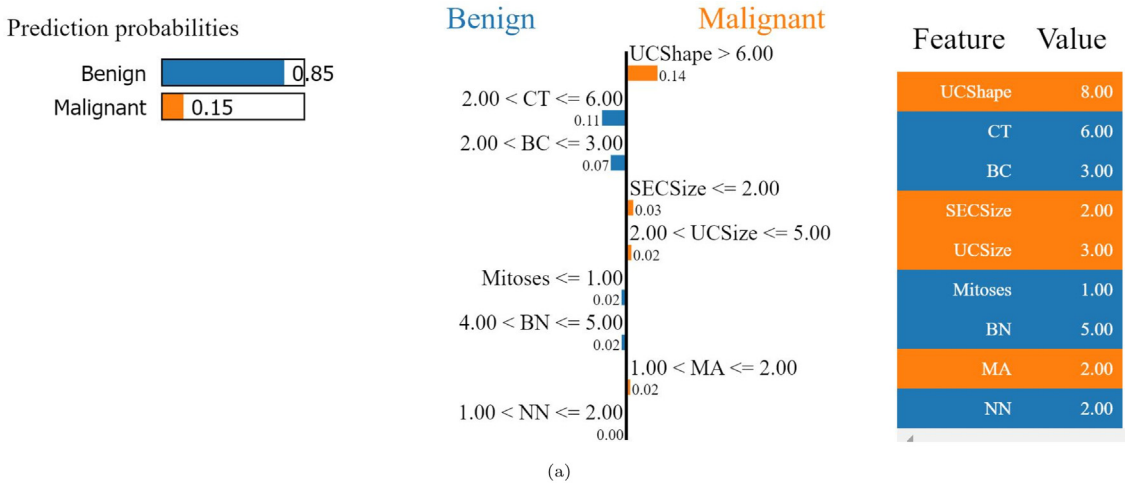
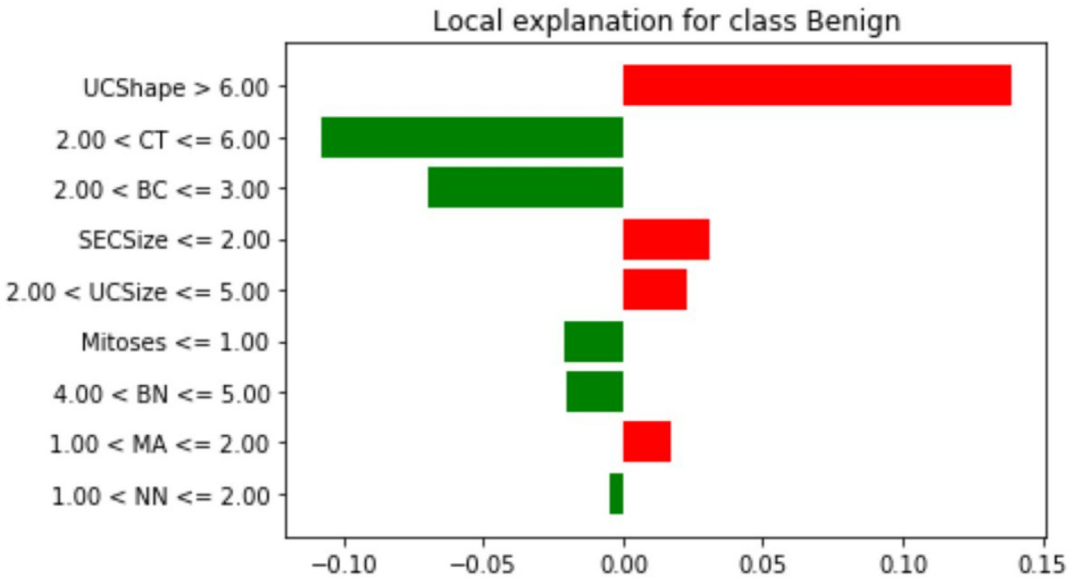


Fig. 8. (a) The illustration shows the relation between the corresponding feature and predictions of Malignant. The orange color shows the positively correlated features. (b) The illustration shows the local explanation for Malignant. The green-colored feature shows the negatively correlated features.

better for clustering the input data. We modified the classical k-means algorithm's assignment stage to choose a random centroid from those near the closest centroid to implement our improved qk-means algorithm, and we added errors to updated clusters. The completeness and accuracy of the evaluation are two criteria used for the classical and classical-quantum clustering algorithms (See Eqs. (15) and (16)). A clustering is fully complete when each cluster contains information pointing to other clusters that are similar to it. The completeness metric shows how closely the clustering technique adheres to the ideal (completeness score). A permutation of the cluster label values does not alter the score value. Knowledge of the actual labels of the data points is optional to understand the silhouette coefficient. Only the initial, unlabeled sample and the clustering outcome are used to measure the clustering's quality. To begin, the silhouette score is calculated for each observation. Let p be the average distance between the data points from the closest cluster, and d represents the mean distance between data points inside a cluster (different from the one datapoint belongs to). The silhouette measure is then presented in Eq. (16). The outcomes of clustering will be better if the silhouette score (S) is higher. By calculating the number of clusters that maximizes the silhouette coefficient, the silhouette score is used to determine the ideal number of clusters k. Our improved qk-means clustering algorithm updates the centroids after the cluster assignment. It helps to form the quality cluster while new data points are added to the new clusters. For each model, functions need to be created that follow a similar structure, executing all algorithms and setting random states to zero. The KNN and random forest model set the criterion equal to entropy which is used to calculate the information gain within the data points of the model [34,36]. Both the train and test data are passed through the algorithms to check the classification accuracy as shown in Table 2. The random forest and KNN performed with the classification accuracies of 90.1% and 88.7% respectively. Then the quality of the predictions are made and calculated using the true and false positives and negatives. The possible outcomes of the binary classification are TP: True Positive, TN: True Negative, FP: False Positive, FN: False Negative. The input data is clustered into c classes. Table 2 discusses the model's classification accuracy metric, i.e., precision and recall from Eq. (8) to Eq. (14). The comparison between the existing qk-means method and our improved qk-means algorithm is shown in Table 3 and Fig. 6. Utilizing the completeness, silhouette score, and accuracy, the revised qk-means algorithm's prediction and cluster quality are assessed. With clustering accuracies of 92.6% and 93.7% respectively, our improved qk-means algorithm is used to explain the predictions from the breast cancer and knee MRI data (See Figs. 6 and 7). The trust determining in prediction is



(a)



(b)

Fig. 9. (a) The illustration shows the relation between the corresponding feature and predictions of Benign. The orange color shows the positively correlated features, and the blue color features show the negatively correlated features against Malignant. (b) The illustration shows the local explanation for Benign. The green-colored feature shows the negatively correlated features against the prediction results of Malignant.

essential to making decisions using the improved *qk*-means. After assessing the effective outcomes from our modified qk-means, we provide a local explanation for the created clusters.

5. Discussion

The medical data has more dimensions and relationships between predictors and outcome variables. The complex relationship makes it challenging to build a model that provides information about all the detailed information or relations. The feature importance plot is one of the solutions to find essential features while predicting the outcome. However, it does not give the indication or direction of the mapping between the features. In the experiment, seven features of a breast cancer dataset and five features of a Knee MRI dataset are used. First, the decision boundary of a model needs to be measured and checked whether the data point is Benign or Malignant. So the prediction made is highly non-linear (there is no easy way to explain the relationship in how we perform predictions). LIME showcases the features to create or develop predictions when we have many features available. We use LIME with the improved qk-means algorithm to develop the predictions from the breast cancer dataset (Algo. 2 and Fig. 4). LIME takes the observations and calculates the distance metrics or similarities between the test dataset. The improved qk-means algorithm makes clusters for predictors. Finally, LIME tries a combination of predictors, i.e. *n* number of predictors, to figure out the minimum number of predictors that gives the maximum likelihood of the clusters predicted by the improved qk-means algorithm. It picks *n* features with similarity scares to derive coefficients, which serves as an explanation for the local scale's observation. LIME gets a pipeline that begins with data pre-processing and runs the ML model (here, we used improved *qk*-means algorithm) as input for interpreting the model result. The LIME method offers various categories of the model for various data types such as tabular, image, text, etc. In our paper, tabular data is used, so the tabular explainer is selected to explain the predictions ("LimeTabularExplainer"). Once the improved *qk*-means model has been trained, the LIME explainer gives the relation between the selected feature and prediction. Figs. 8 and 9 illustrate the prediction of malignant and benign, respectively. The top left corner of Figs. 8(a) and 8(b) illustrate the prediction

and details of positively and negatively correlated features. The prediction of benign provides positive and negative correlations to the prediction of malignant. In both Figs. 8 and 9, the green color shows a negative relation against the result of malignant. The green color depicts the negative relation against the result of malignant but benign results favor.

6. Conclusion

In this paper, we implement improved qk-means clustering algorithm, and test them on the Breast Cancer and Knee MRI datasets. We then calculate, compare, and evaluate the various results based on accuracy, completeness, and silhouette score to demonstrate the clustering accuracy and reliability of the improved qk-means algorithm. Following a precise comparison of our models, we show that our proposed improved qk-means clustering algorithm outperforms all exisitng algorithms with a greater accuracy of 93.2% and completeness of 94.5%. We merge the LIME approach and the improved qk-means clustering algorithm to explain the predictions. XAI models pose complex challenges and offer various opportunities for the medical sector. Recent research and proposed methods show promising directions and provide solutions using quantum explainability to model medical data.

XAI accelerates decision speed by achieving transparency in solving complex problems. The classical machine is reaching its limit, so we need explainable quantum models to learn from hidden patterns in medical data. Quantum computing has demonstrated exciting potential to accelerate ML and data science-related applications and so on in the future.

CRediT authorship contribution statement

Shradha Deshmukh: Literature analysis, Interpretation of results and the preparation of the manuscript. **Bikash K. Behera:** Literature analysis, Interpretation of results and the preparation of the manuscript. **Preeti Mulay:** Literature analysis, Interpretation of results and the preparation of the manuscript. **Emad A. Ahmed:** Literature analysis, Interpretation of results and the preparation of the manuscript. **Saif Al-Kuwari:** Literature analysis, Interpretation of results and the preparation of the manuscript. **Prayag Tiwari:** Literature analysis, Interpretation of results and the preparation of the manuscript. **Ahmed Farouk:** Literature analysis, Interpretation of results and the preparation of the manuscript.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The source code of the described method is available on GitHub (https://github.com/shradha-deshmukh/Explainable-Quantum-Clustering.git).

References

[1] A.J. London, Artificial intelligence and black-box medical decisions: accuracy versus explainability, Hastings Cent. Rep. 49 (1) (2019) 15–21.

[2] D. Slack, A. Hilgard, S. Singh, H. Lakkaraju, Reliable post hoc explanations: Modeling uncertainty in explainability, Adv. Neural Inf. Process. Syst. 34 (2021) 9391–9404.

[3] I.P. de Sousa, M.M.B.R. Vellasco, E.C. da Silva, Explainable artificial intelligence for bias detection in covid ct-scan classifiers, Sensors 21 (16) (2021) 5657, http://dx.doi.org/10.3390/s21165657.

[4] S. Vollert, M. Atzmueller, A. Theissler, Interpretable machine learning: A brief survey from the predictive maintenance perspective, in: 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, IEEE, 2021, pp. 01–08.

[5] E. Tjoa, C. Guan, A survey on explainable artificial intelligence (xai): Toward medical xai, IEEE Trans. Neural Netw. Learn. Syst. 32 (11) (2020) 4793–4813.

[6] N. Bussmann, P. Giudici, D. Marinelli, J. Papenbrock, Explainable AI in fintech risk management, Front. Artif. Intell. 3 (2020) 26.

[7] J. Lötsch, D. Kringel, A. Ultsch, Explainable artificial intelligence (XAI) in biomedicine: Making AI decisions trustworthy for physicians and patients, BioMedInformatics 2 (1) (2022) 1–17.

[8] H.W. Loh, C.P. Ooi, S. Seoni, P.D. Barua, F. Molinari, U.R. Acharya, Application of explainable artificial intelligence for healthcare: A systematic review of the last decade, Comput. Methods Programs Biomed. (2022) 107161.

[9] Y.-L. Chou, C. Moreira, P. Bruza, C. Ouyang, J. Jorge, Counterfactuals and causability in explainable artificial intelligence: Theory, algorithms, and applications, Inf. Fusion 81 (2022) 59–83, http://dx.doi.org/10.1016/j.inffus.2021.11.003.

[10] S.R. Islam, W. Eberle, S.K. Ghafoor, M. Ahmed, Explainable artificial intelligence approaches: A survey, 2021, arXiv preprint arXiv:2101.09429.

[11] M. Erwig, P. Kumar, Explainable dynamic programming, J. Funct. Programming 31 (2021) http://dx.doi.org/10.1017/S0956796821000083.

[12] A. Kalantari, A. Kamsin, S. Shamshirband, A. Gani, H. Alinejad-Rokny, A.T. Chronopoulos, Computational intelligence approaches for classification of medical data: State-of-the-art, future challenges and research directions, Neurocomputing 276 (2018) 2–22, http://dx.doi.org/10.1016/j.neucom.2017.01.126.

[13] I. Scholl, T. Aach, T.M. Deserno, T. Kuhlen, Challenges of medical image processing, Comput. Sci. Res. Dev. 26 (1) (2011) 5–13, http://dx.doi.org/10.1007/s00450-010-0146-9.

[14] A. Ampavathi, Research challenges and future directions towards medical data processing, Comput. Methods Biomech. Biomed. Eng. Imaging Visual. (2022) 1–20.

[15] Y. Zhang, Y. Weng, J. Lund, Applications of explainable artificial intelligence in diagnosis and surgery, Diagnostics 12 (2) (2022) 237, http://dx.doi.org/10.3390/diagnostics12020237.

[16] M.S. Kamal, N. Dey, L. Chowdhury, S.I. Hasan, K. Santosh, Explainable AI for glaucoma prediction analysis to understand risk factors in treatment planning, IEEE Trans. Instrum. Meas. 71 (2022) 1–9, http://dx.doi.org/10.1109/TIM.2022.3171613, URL https://doi.org/10.1109/TIM.2022.3171613.

[17] A.B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, et al., Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI, Inf. Fusion 58 (2020) 82–115.

[18] S. Mohseni, N. Zarei, E.D. Ragan, A multidisciplinary survey and framework for design and evaluation of explainable AI systems, ACM Trans. Interact. Intell. Syst. (TiiS) 11 (3–4) (2021) 1–45.

[19] H. Bansal, B. Chinagundi, P.S. Rana, N. Kumar, An ensemble machine learning technique for detection of abnormalities in knee movement sustainability, Sustainability 14 (20) (2022) 13464, http://dx.doi.org/10.3390/su142013464.

[20] Q. Liu, L. Wang, T-test and ANOVA for data with ceiling and/or floor effects, Behav. Res. Methods 53 (1) (2021) 264–277, http://dx.doi.org/10.3758/s13428-020-01407-2.

[21] D. Kim, J. Chung, J. Choi, M.D. Succi, J. Conklin, M.G.F. Longo, J.B. Ackman, B.P. Little, M. Petranovic, M.K. Kalra, et al., Accurate auto-labeling of chest X-ray images based on quantitative similarity to an explainable AI model, Nature Commun. 13 (1) (2022) 1–15, http://dx.doi.org/10.1038/s41467-022-29437-8.

[22] S. Panakkal, B. Falkenstein, A.B. Tosun, B. Campbell, M. Becich, J. Fine, D.L. Taylor, S.C. Chennubhotla, F. Pullara, TumorMapr™ analytical software platform: Unbiased spatial analytics and explainable AI (xAI) platform for generating data, extracting information, and creating knowledge from multi to hyperplexed fluorescence and/or mass spectrometry image datasets, Cancer Res. 82 (12_Supplement) (2022) 454, http://dx.doi.org/10.1158/1538-7445.AM2022-454.

[23] M. Hossain, S.S. Haque, H. Ahmed, H.A. Mahdi, A. Aich, Early stage detection and classification of colon cancer using deep learning and explainable AI on histopathological images (Ph.D. thesis), Brac University, 2022.

[24] N. Gozzi, E. Giacomello, M. Sollini, M. Kirienko, A. Ammirabile, P. Lanzi, D. Loiacono, A. Chiti, Image embeddings extracted from CNNs outperform other transfer learning approaches in chest radiographs' classification, Diagnostics (2022) http://dx.doi.org/10.21203/rs.3.rs-1361817/v1.

[25] W.H. Abir, M. Uddin, F.R. Khanam, T. Tazin, M.M. Khan, M. Masud, S. Aljahdali, et al., Explainable AI in diagnosing and anticipating leukemia using transfer learning method, Comput. Intell. Neurosci. 2022 (2022) http://dx.doi.org/10.1155/2022/5140148.

[26] P. Silitonga, Clustering of patient disease data by using K-means clustering, Int. J. Comput. Sci. Inform. Secur. (IJCSIS) 15 (7) (2017) 219–221.

[27] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, S. Lloyd, Quantum machine learning, Nature 549 (7671) (2017) 195–202, http://dx.doi.org/10.1038/nature23474.
[28] P. Tiwari, S. Dehdashti, A.K. Obeid, P. Marttinen, P. Bruza, Kernel method based on non-linear coherent states in quantum feature space, J. Phys. A 55 (35) (2022) 355301.
[29] N. Laxminarayana, N. Mishra, P. Tiwari, S. Garg, B.K. Behera, A. Farouk, Quantum-assisted activation for supervised learning in healthcare-based intrusion detection systems, IEEE Trans. Artif. Intell. (2022) 1–8, http://dx.doi.org/10.1109/TAI.2022.3187676, URL https://ieeexplore.ieee.org/document/9813378.
[30] S. Moradi, C. Brandner, C. Spielvogel, D. Krajnc, S. Hillmich, R. Wille, W. Drexler, L. Papp, Clinical data classification with noisy intermediate scale quantum computers, Sci. Rep. 12 (1) (2022) 1–9, http://dx.doi.org/10.1038/s41598-022-05971-9.
[31] M. Kapcia, H. Eshkiki, J. Duell, X. Fan, S. Zhou, B. Mora, ExMed: An AI tool for experimenting explainable AI techniques on medical data analytics, in: 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence, ICTAI, IEEE, 2021, pp. 841–845, http://dx.doi.org/10.1109/ICTAI52525.2021.00134.
[32] H. Yoo, S. Han, K. Chung, Diagnosis support model of cardiomegaly based on CNN using ResNet and explainable feature map, IEEE Access 9 (2021) 55802–55813, http://dx.doi.org/10.1109/ACCESS.2021.3068597.
[33] S. Knapič, A. Malhi, R. Saluja, K. Främling, Explainable artificial intelligence for human decision support system in the medical domain, Mach. Learn. Knowl. Extract. 3 (3) (2021) 740–770, http://dx.doi.org/10.3390/make3030037.
[34] J.-B. Lamy, B. Sekar, G. Guezennec, J. Bouaud, B. Séroussi, Explainable artificial intelligence for breast cancer: A visual case-based reasoning approach, Artif. Intell. Med. 94 (2019) 42–53, http://dx.doi.org/10.1016/j.artmed.2019.01.001.
[35] T.A. Assegie, An optimized K-nearest neighbor based breast cancer detection, J. Robot. Control (JRC) 2 (3) (2021) 115–118, http://dx.doi.org/10.18196/jrc.2363.
[36] B. Dai, R.-C. Chen, S.-Z. Zhu, W.-W. Zhang, Using random forest algorithm for breast cancer diagnosis, in: 2018 International Symposium on Computer, Consumer and Control (IS3C), IEEE, 2018, pp. 449–452, http://dx.doi.org/10.1109/IS3C.2018.00119.
[37] S. Vashisth, I. Dhall, G. Aggarwal, Design and analysis of quantum powered support vector machines for malignant breast cancer diagnosis, J. Intell. Syst. 30 (1) (2021) 998–1013.
[38] R.C. Ripan, I.H. Sarker, S.M. Hossain, M. Anwar, R. Nowrozy, M.M. Hoque, M. Furhad, et al., A data-driven heart disease prediction model through K-means clustering-based anomaly detection, SN Comput. Sci. 2 (2) (2021) 1–12, http://dx.doi.org/10.1007/s42979-021-00518-7.
[39] J.M. Clementino, B.S. Faiçal, C.C. Bones, C. Traina, M.A. Gutierrez, A.J. Traina, Multilevel clustering explainer: An explainable approach to electronic health records, in: 2021 IEEE 34th International Symposium on Computer-Based Medical Systems, CBMS, IEEE, 2021, pp. 253–258, http://dx.doi.org/10.1109/CBMS52027.2021.00073.
[40] M.S. Kamal, N. Dey, L. Chowdhury, S.I. Hasan, K. Santosh, Explainable AI for glaucoma prediction analysis to understand risk factors in treatment planning, IEEE Trans. Instrum. Meas. 71 (2022) 1–9, http://dx.doi.org/10.1109/TIM.2022.3171613.
[41] D. Yeboah, L. Steinmeister, D.B. Hier, B. Hadi, D.C. Wunsch, G.R. Olbricht, T. Obafemi-Ajayi, An explainable and statistically validated ensemble clustering model applied to the identification of traumatic brain injury subgroups, IEEE Access 8 (2020) 180690–180705, http://dx.doi.org/10.1109/ACCESS.2020.3027453.
[42] H. Han, W. Li, J. Wang, G. Qin, X. Qin, Enhance explainability of manifold learning, Neurocomputing (2022) http://dx.doi.org/10.1016/j.neucom.2022.05.119.
[43] Kaggle: Your Machine Learning and Data Science Community, URL https://www.kaggle.com.
[44] B. Zheng, S.W. Yoon, S.S. Lam, Breast cancer diagnosis based on feature extraction using a hybrid of K-means and support vector machine algorithms, Expert Syst. Appl. 41 (4) (2014) 1476–1482, http://dx.doi.org/10.1016/j.eswa.2013.08.044.
[45] K.C. Gowda, E. Diday, Symbolic clustering using a new dissimilarity measure, Pattern Recognit. 24 (6) (1991) 567–578, http://dx.doi.org/10.1016/0031-3203(91)90022-W.
[46] K.M. Boehm, P. Khosravi, R. Vanguri, J. Gao, S.P. Shah, Harnessing multimodal data integration to advance precision oncology, Nat. Rev. Cancer 22 (2) (2022) 114–126, http://dx.doi.org/10.1038/s41568-021-00408-3.
[47] T.Y. Wen, S.A.M. Aris, Hybrid approach of EEG stress level classification using K-means clustering and support vector machine, IEEE Access 10 (2022) 18370–18379, http://dx.doi.org/10.1109/ACCESS.2022.3148380.
[48] R. Tripathy, R.K. Nayak, Eukaryotic plasma cholesterol prediction from human GPCRs using K-means with support vector machine, in: Advanced Deep Learning for Engineers and Scientists, Springer, 2021, pp. 243–257, http://dx.doi.org/10.1007/978-3-030-66519-7_10.
[49] M. Schuld, N. Killoran, Quantum machine learning in feature Hilbert spaces, Phys. Rev. Lett. 122 (4) (2019) 040504, http://dx.doi.org/10.1103/PhysRevLett.122.040504.
[50] A.Y. Wei, P. Naik, A.W. Harrow, J. Thaler, Quantum algorithms for jet clustering, Phys. Rev. D 101 (9) (2020) 094015.
[51] A.W. Harrow, Small quantum computers and large classical data sets, 2020, arXiv preprint arXiv:2004.00026.
[52] Z. Huang, Extensions to the k-means algorithm for clustering large data sets with categorical values, Data Min. Knowl. Discov. 2 (3) (1998) 283–304, http://dx.doi.org/10.1023/A:1009769707641.
[53] B.R. Reddy, Y.V. Kumar, M. Prabhakar, Clustering large amounts of healthcare datasets using fuzzy c-means algorithm, in: 2019 5th International Conference on Advanced Computing & Communication Systems, ICACCS, IEEE, 2019, pp. 93–97, http://dx.doi.org/10.1109/ICACCS.2019.8728503.
[54] S.U. Khan, A.J. Awan, G. Vall-Llosera, K-means clustering on noisy intermediate scale quantum computers, 2019, arXiv preprint arXiv:1909.12183.
[55] X.-H. Li, C.C. Cao, Y. Shi, W. Bai, H. Gao, L. Qiu, C. Wang, Y. Gao, S. Zhang, X. Xue, et al., A survey of data-driven and knowledge-aware explainable ai, IEEE Trans. Knowl. Data Eng. 34 (1) (2020) 29–49, http://dx.doi.org/10.1109/TKDE.2020.2983930.
[56] A. Holzinger, A. Saranti, C. Molnar, P. Biecek, W. Samek, Explainable AI methods-a brief overview, in: International Workshop on Extending Explainable AI beyond Deep Models and Classifiers, Springer, 2022, pp. 13–38, http://dx.doi.org/10.1007/978-3-031-04083-2_2.

Capacity and quantum geometry of parametrized quantum circuits

Tobias Haug,[1,*] Kishor Bharti,[2] and M. S. Kim[1]

[1] *QOLS, Blackett Laboratory, Imperial College London SW7 2AZ, UK*
[2] *Centre for Quantum Technologies, National University of Singapore 117543, Singapore*

To harness the potential of noisy intermediate-scale quantum devices, it is paramount to find the best type of circuits to run hybrid quantum-classical algorithms. Key candidates are parametrized quantum circuits that can be effectively implemented on current devices. Here, we evaluate the capacity and trainability of these circuits using the geometric structure of the parameter space via the effective quantum dimension, which reveals the expressive power of circuits in general as well as of particular initialization strategies. We assess the expressive power of various popular circuit types and find striking differences depending on the type of entangling gates used. Particular circuits are characterized by scaling laws in their expressiveness. We identify a transition in the quantum geometry of the parameter space, which leads to a decay of the quantum natural gradient for deep circuits. For shallow circuits, the quantum natural gradient can be orders of magnitude larger in value compared to the regular gradient; however, both of them can suffer from vanishing gradients. By tuning a fixed set of circuit parameters to randomized ones, we find a region where the circuit is expressive, but does not suffer from barren plateaus, hinting at a good way to initialize circuits. We show an algorithm that prunes redundant parameters of a circuit without affecting its effective dimension. Our results enhance the understanding of parametrized quantum circuits and can be immediately applied to improve variational quantum algorithms.

I. INTRODUCTION

Quantum computers promise to tackle challenging problems for classical computers such as drug design, combinatorial optimisation and simulation of many-body physics. While fully-fledged large-scale quantum computers with error correction are not expected to be available for many years, noisy intermediate-scale quantum (NISQ) devices have been investigated as a way to approach computationally hard problems with quantum processors available now and in the near future [1, 2]. Variational quantum algorithms (VQA) [3–6] have been a major hope in achieving a quantum speedup with NISQ devices. The core idea is to update a parametrized quantum circuit (PQC) in a hybrid quantum-classical fashion. Measurements performed on the PQC are fed into a classical computer to propose a new set of variational parameters. A key challenge has been the occurrence of barren plateaus, i.e. the gradients used for optimisation vanish exponentially with increasing number of qubits [7], as well as for various types of cost functions [8], entanglement [9] and noise [10]. Further, the classical optimization part of variational algorithms was shown to be NP-hard [11]. Quantum algorithms that avoid the feed-back loop to circumvent the barren plateau problems have been proposed [12–18]. Besides this approach, initialization strategies [19–21] and layer-wise learning [22] for VQA could help to solve the aforementioned problems. However, tools to evaluate the power of these strategies are lacking. Hardware efficient ansätze have been proposed to tailor a PQC to the restrictions of the hardware [23]. A widely used choice is quantum circuits arranged in layers of single-qubit rotations followed by two-qubit entangling gates. However, a key question is the space of possible states this ansatz type can express [24–27].

Here, we introduce the effective quantum dimension G_C and parameter dimension D_C as a quantitative measure of the capacity of a PQC. Parameter dimension D_C measures the total number of independent parameters a quantum state defined by the PQC can express. In contrast, the effective quantum dimension G_C [28, 29] is a local measure to quantify the space of states that can be accessed by locally perturbing the parameters of the PQC. Both measures can be derived from the quantum geometric structure of the PQC via the quantum Fisher information metric (QFI) $\mathcal{F}$ [30, 31]. From the QFI, one can obtain the quantum natural gradient (QNG) for a more efficient optimisation via gradients [30–32]. These methods allow us to evaluate the expressive power, trainability and number of redundant parameters of different PQCs, and find better initialization strategies.

As demonstration of our tools, we provide an in-depth investigation of popular hardware-efficient circuits, composed of layered single-qubit rotations and two-qubit entangling gates in various arrangements. We find striking differences depending on the choice of circuit structure that affect both the expressive power of the PQC in general as well as the quality of specific initialization strategies. We calculate the number of redundant parameters of various PQC types, as well as how fast they converge towards random quantum states as a function of the number of layers. The choice of entangling gate has a pronounced effect on the expressive power of particular initialization strategies.

We reveal a transition in the spectrum of the QFI in deep circuits, which leads to a decay of the QNG. For shallow circuits, the QNG can be orders of magnitude

* thaug@ic.ac.uk

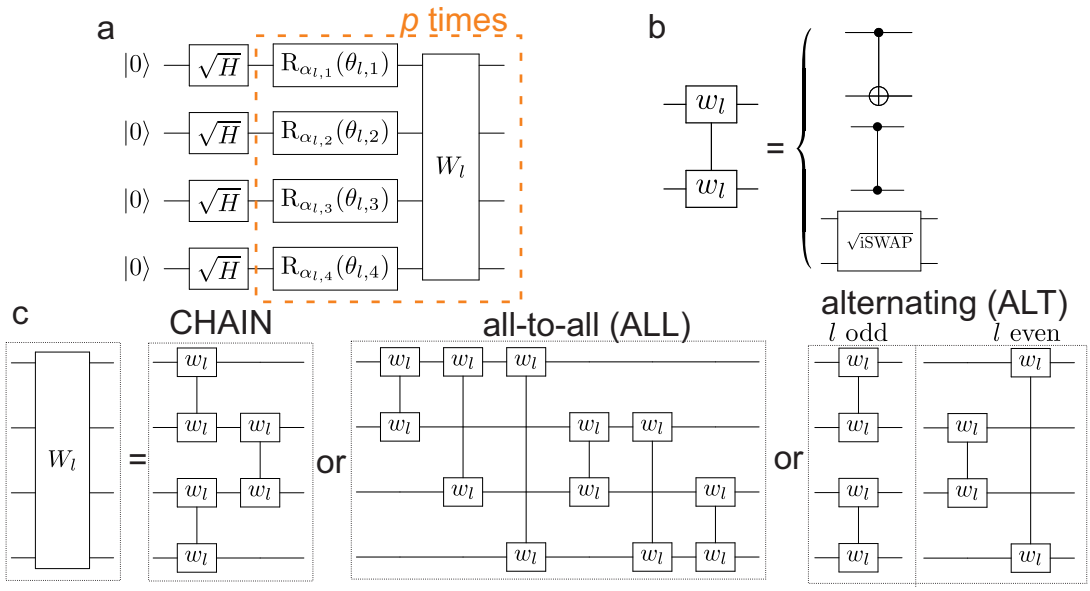


FIG. 1. **a)** Sketch of hardware-efficient parametrized quantum circuit (PQC) $U(\theta)|0\rangle^{\otimes N} = \prod_{l=p}^{1}\left[W_l V_l(\theta_l)\right]\sqrt{H_\mathrm{d}}^{\otimes N}|0\rangle^{\otimes N}$ with parameters θ and initial state $|0\rangle$ of all N qubits being in state zero. The PQC consists of an initial layer of $\sqrt{H_\mathrm{d}}$ gates applied to each qubit, where H_d is the Hadamard gate, followed by p repeated layers of parametrized single qubit rotations $V_l(\theta_l)$ and entangling gates W_l. $V_l(\theta_l)$ consists of single-qubit rotations $R_\alpha(\theta_{l,n}) = \exp(-i\sigma_n^\alpha\theta_{l,n}/2)$ at layer l and qubit n around axis $\alpha \in \{x, y, z\}$. **b)** Two-qubit entangling gates w_l considered are CNOT gates (control-σ_x), CPHASE gates (control-σ_z, $\mathrm{diag}(1, 1, 1, -1)$) or $\sqrt{\mathrm{iSWAP}}$ gates. **c)** Entangling layer W_l is composed of the two-qubit entangling gates w_l, which are arranged in either a nearest-neighbor one-dimensional chain topology (denoted as CHAIN), all-to-all connection (ALL) or in a alternating fashion (ALT) for even and odd layers l.

larger than the regular gradient. However, both suffer from the barren plateau problem. By tuning the PQCs parameters from zero to a random set of parameters, we find a region where both large gradients and large effective quantum dimension G_C coexist, which could serve as a good set of initial parameters for the training of variational algorithms. Finally, as an application of our method we propose and apply an algorithm that prunes redundant parameters from PQCs, while keeping the parameter dimension constant. This algorithm helps us to find expressive PQCs with a reduced number of parameters to simplify training for variational algorithms as well as a reduced circuit depth to ease the impact of noise.

The paper is organized as follows. First, we define PQCs in Sec.II and the parameter dimension D_C in Sec.III. Then, we introduce the effective quantum dimension G_C in Sec.IV. Our results and the algorithm are presented in Sec.V, which are discussed in Sec.VI. We give an overview of the definitions of symbols in Tab.I.

II. PARAMETRIZED QUANTUM CIRCUITS

A PQC generates a quantum state of N qubits

$$|\psi(\theta)\rangle = U(\theta)|0\rangle^{\otimes N}, \tag{1}$$

with the unitary $U(\theta)$, the M-dimensional parameter vector θ and product state $|0\rangle^{\otimes N}$ as shown in Fig.1. The structure of the PQC influences its power to express quantum states [24, 25, 33]. One way to measure expressiveness is by determining the distance between the distribution of states generated by the circuit and the Haar random distribution of states [24, 25]. This tells us how well the PQC can express arbitrary states across the Hilbert space. The appearance of barren plateaus or vanishing gradients is connected to the aforementioned measure [7, 34]. The variance of the gradient $\mathrm{var}(\partial_i E) = \langle(\partial_i E)^2\rangle - \langle\partial_i E\rangle^2$ ($\langle.\rangle$ denoting statistical average over many random instances) in respect to the expectation value of a Hamiltonian H ($E = \langle 0|U^\dagger(\theta)HU(\theta)|0\rangle$) can vanish exponentially with the number of qubits for PQCs with a random choice of parameters. The variance decreases also with number of layers p of the PQC until a specific p_r, where it remains constant upon further increase of $p > p_\mathrm{r}$. For local cost functions, it has been shown that in most cases low variance of the gradient of such PQCs correlates with high expressibility [34].

III. PARAMETER DIMENSION

We now introduce the parameter dimension D_C of a PQC as another measure of capacity. As example, we take a PQC that can represent arbitrary N qubit quantum states which is parametrized by in total $M = 2^{N+1}$

Name	Symbol	Definition
Parameterized quantum circuit (PQC)	$\|\psi(\theta)\rangle$	$U(\theta)\|0\rangle^{\otimes N}$
Quantum Fisher information metric (QFI)	$\mathcal{F}_{ij}(\theta)$	$\mathrm{Re}(\langle\partial_i\psi\|\partial_j\psi\rangle - \langle\partial_i\psi\|\psi\rangle\langle\psi\|\partial_j\psi\rangle)$
Expectation value of Hamiltonian H	E	$E = \langle 0\|U^\dagger(\theta)HU(\theta)\|0\rangle$
Quantum natural gradient	QNG	$\mathcal{F}^{-1}(\theta)\partial_k E$
Variance	$\mathrm{var}(E)$	$\langle(E)^2\rangle - \langle E\rangle^2$
Effective dimension	$G_\mathrm{C}(\theta)$	$\mathrm{rank}(\mathcal{F}(\boldsymbol{\theta}))$
Parameter dimension	D_C	$G_\mathrm{C}(\theta_\mathrm{random})$
Number of parameters of a PQC	M	
Redundancy	R	$(M - D_\mathrm{C})/M$

TABLE I. Definitions of symbols.

parameters $\boldsymbol{a}$, $\boldsymbol{b}$

$$|\psi(\boldsymbol{a},\boldsymbol{b})\rangle = \sum_{j=1}^{2^N}(a_j + ib_j)|j\rangle\,, \tag{2}$$

where $|j\rangle$ is the j-th computational basis state and $a_j, b_j \in \mathcal{R}$. One can map the above state to $D_\mathrm{C} = 2^{N+1} - 2$ independent parameters, that lie on the surface of $2^{N+1} - 1$ dimensional sphere. Of the in total M parameters, the final 2 parameters are dependent and do not change the quantum state, as they correspond to the norm and global phase of the quantum state. Conversely, for a generic real-valued quantum state with $b_j = 0$, we find $D_\mathrm{C} = 2^N - 1$ independent parameters, with 1 dependent parameter due to the norm of the real-valued quantum state. Analogous to the generic quantum state, we now define the parameter dimension D_C for a PQC C as the number of independent parameters that the PQC can express in the space of quantum states. In general, D_C for N qubits is upper bounded by the generic state Eq. (2) with $D_\mathrm{C} \leq 2^{N+1} - 2$. We define the redundancy

$$R = \frac{M - D_\mathrm{C}}{M}\,, \tag{3}$$

which is the fraction of dependent parameters of the PQC that do not contribute to changing the quantum state. In the next section, we show how D_C can be determined for hardware efficient PQCs.

IV. EFFECTIVE QUANTUM DIMENSION

Now, we explain how the QFI $\mathcal{F}(\theta)$ quantifies the expressive power of a PQC (see Appendix B for an introduction to the QFI and QNG, and Appendix G on how to calculate it). One can relate $\mathcal{F}(\theta)$ to the distance in the space of pure quantum states, which is given by the Fubini-Study distance

$$\mathrm{Dist}_Q\Big(|\psi(\theta)\rangle,\ |\psi(\theta+\mathrm{d}\theta)\rangle\Big)^2 = \sum_{i,j}\mathcal{F}_{ij}(\theta)\mathrm{d}\theta_i\mathrm{d}\theta_j\,, \tag{4}$$

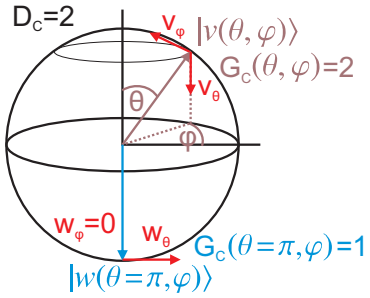


FIG. 2. Example to demonstrate the effective quantum dimension G_C and parameter dimension D_C for a single qubit parametrized as $|\psi(\theta,\varphi)\rangle = \cos(\theta/2)|0\rangle + \exp(i\varphi)\sin(\theta/2)|1\rangle$. The quantum state is described by $M = 2$ parameters θ and φ. $D_\mathrm{C} = 2$ is the number of *independent* parameters of the quantum state. $G_\mathrm{C}(\theta,\varphi)$ denotes the number of independent directions the quantum state can change to by locally perturbing its parameters θ, φ. For a random state $|v\rangle$ ($\theta \notin \{0,\pi\}$) two possible directions exist, along v_θ and v_φ. The particular state $|w(\theta=\pi,\varphi)\rangle$ can only be perturbed in direction w_θ as adjusting φ does not change the state (e.g. $|w(\pi,\varphi+\epsilon)\rangle = |w(\pi,\varphi)\rangle$), thus $G_\mathrm{C}(\theta=\pi,\varphi) = 1$.

where $\mathrm{Dist}_Q(x,y) = |\langle x|y\rangle|^2$ and the QFI [30, 31]

$$\mathcal{F}_{ij}(\theta) = \mathrm{Re}(\langle\partial_i\psi|\partial_j\psi\rangle - \langle\partial_i\psi|\psi\rangle\langle\psi|\partial_j\psi\rangle)\,, \tag{5}$$

which corresponds to the real part of the quantum geometric tensor. $\mathcal{F}(\theta)$ quantifies the change of the quantum state when adjusting its parameter θ infinitesimally to $\theta + \mathrm{d}\theta$. The eigenvalue decomposition

$$\mathcal{F}(\theta) = VSV^T\,, \tag{6}$$

gives us V, which is a real-valued unitary with the i-th eigenvector $\alpha^{(i)}$ placed at the i-th column of V, and S, which is a diagonal matrix with the M non-negative eigenvalues $\lambda^{(i)}$ of $\mathcal{F}(\theta)$ along the diagonal. The eigenvalues and eigenvectors obey the equation $\mathcal{F}(\theta)\alpha^{(i)} = \lambda^{(i)}\alpha^{(i)}$. Inserting Eq. (6) into Eq. (4) gives us

$$\mathrm{Dist}_Q\Big(|\psi(\theta)\rangle,\ |\psi(\theta+d\theta)\rangle\Big) = \mathrm{d}\theta^T\mathcal{F}d\theta = d\theta^T VSV^T\mathrm{d}\theta\,. \tag{7}$$

Now, we assume that the small variations in θ are in the direction of the i-th eigenvector of $\mathcal{F}(\theta)$ with $\mathrm{d}\theta = \mathrm{d}\mu\alpha^{(i)}$, where $\mathrm{d}\mu$ is an infinitesimal scalar. We find

$$\mathrm{Dist}_Q\Big(|\psi(\theta)\rangle,\ |\psi(\theta + d\mu\alpha^{(i)})\rangle\Big)^2 = d\mu{\alpha^{(i)}}^T V S V^T \alpha^{(i)} \mathrm{d}\mu = \lambda^{(i)} \mathrm{d}\mu \mathrm{d}\mu\,,$$

where we have used $V^T\alpha^{(i)} = e^{(i)}$, where $e^{(i)}$ is the i-th basis vector. When updating $\theta' = \theta + \mathrm{d}\mu\alpha^{(i)}$, the quantum state changes at a rate that is proportional to $\lambda^{(i)}$. Eigenvalues $\lambda^{(i)} = 0$ are called singularities as there is no change in the quantum state at all, i.e. $|\langle\psi(\theta)|\psi(\theta + \mathrm{d}\mu\alpha^{(i)})\rangle| = 1$. The case $\lambda^{(i)}$ being very small, i.e. $1 \gg \lambda^{(i)} > 0$, is called near singularity and is associated with plateaus in classical machine learning where training slows down [35].

We now define the effective quantum dimension $G_\mathrm{C}(\theta)$ for a PQC C as the rank of the QFI $\mathcal{F}(\theta)$. It is given as the total number of non-zero eigenvalues $\lambda^{(i)}(\theta)$ of $\mathcal{F}(\theta)$ initialized with parameters θ [28, 29]

$$G_\mathrm{C}(\theta) = \sum_{i=1}^{M} \mathcal{I}(\lambda^{(i)}(\theta))\,, \tag{8}$$

where $\mathcal{I}(x) = 0$ for $x = 0$ and $\mathcal{I}(x) = 1$ for $x \neq 0$. $G_\mathrm{C}(\theta)$ is a local measure of expressiveness that counts the number of independent directions in the state space that can be accessed by an infinitesimal update of θ.

A straightforward example is a generic single qubit quantum state shown in Fig.2

$$|\psi(\theta,\varphi)\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + \exp(i\varphi)\sin\left(\frac{\theta}{2}\right)|1\rangle \tag{9}$$

$$\mathcal{F}(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & \sin^2(\theta) \end{bmatrix}. \tag{10}$$

The eigenvalues and eigenvectors of the QFI $\mathcal{F}$ are straightforward to calculate with $\lambda_1 = 1$, $\alpha_1 = \{1, 0\}$ and $\lambda_2 = \sin^2(\theta)$, $\alpha_2 = \{0, 1\}$. The effective quantum dimension is $G_\mathrm{C}(\theta,\varphi) = D_\mathrm{C} = 2$, except for the special case $\theta = n\pi$, n integer, wjere the eigenvalue is $\lambda_2 = 0$ and thus $G_\mathrm{C}(n\pi,\varphi) = 1$. Here, any change in the direction of eigenvector α_2 (corresponding to changing φ) will not yield any change in the underlying quantum state. However note that except for these singular parameters we find $G_\mathrm{C} = 2$, which is equivalent to the maximal number of independent parameters D_C of the system.

As further example we consider the single qubit circuit with Pauli z matrix σ_z and Hadamard gate H_d

$$U(\theta)|0\rangle = \prod_{i=1}^{M}\left[\exp(-i\frac{\theta_i}{2}\sigma_z)\right] H_\mathrm{d}|0\rangle\,. \tag{11}$$

Here, we find $\mathcal{F} = \frac{1}{4}J_{M,M}$, where $J_{M,M}$ is a $M \times M$ matrix filled with ones. Diagonalizing $\mathcal{F}$ gives us $M-1$ eigenvalues with $\lambda = 0$, and one eigenvalue $\lambda_1 = \frac{M}{4}$ with eigenvector $\alpha_1 = \frac{1}{\sqrt{M}}J_{M,1}$. This circuit has a low parameter dimension $D_\mathrm{C} = 1$ and a large redundancy of $R = (M-1)/M$, i.e. there are $M-1$ parameter directions which do not yield any change of the quantum state.

For the type of PQC as shown in Fig.1, which are arranged in a layer-wise structure with the parametrized gates being Pauli operators, the effective quantum dimension G_C is equal or less than the parameter dimension D_C, which in turn is equal or less than the number of parameters M

$$G_\mathrm{C}(\theta) \leq D_\mathrm{C} \leq M\,. \tag{12}$$

Given the aforementioned PQC types with a random set of parameters $\theta_\mathrm{random} \in \mathrm{random}(0, 2\pi)$, we find numeric evidence that $G_\mathrm{C}(\theta_\mathrm{random})$ is approximately equivalent to D_C

$$G_\mathrm{C}(\theta_\mathrm{random}) \simeq D_\mathrm{C}\,. \tag{13}$$

Thus, we can calculate D_C by determining $G_\mathrm{C}(\theta_\mathrm{random})$ for random sets of PQC parameters. The core intuition is that starting from a sufficiently random initial parameter set, a change of the PQC parameters in the right direction is able to bring one closer to any quantum state that can be expressed by the PQC. For specific choices of parameters such as $\theta = 0$ we find $G_\mathrm{C} < D_\mathrm{C}$. Moving sufficiently away from these special points, we recover that $G_\mathrm{C} \simeq D_\mathrm{C}$.

We stress that Eq. (13) is not valid for arbitrary quantum circuits, e.g. circuits where the parameters do not enjoy a 2π periodicity. As simple example take the evolution of a single qubit with a single parameter t $U(t)|0\rangle = \exp(-i\sqrt{2}\sigma_z t)\exp(-i\sigma_x t)|0\rangle$. The evolution over all possible t (note the absence of 2π periodicity) will cover all possible quantum states and thus $D_\mathrm{C} = 2$, whereas the effective quantum dimension (with only a single parameter t) is $G_\mathrm{C} = 1 < D_\mathrm{C}$.

We now consider different types of hardware efficient PQC $|\psi(\theta)\rangle = U(\theta)|0\rangle^{\otimes N}$, which are circuits that can be efficiently run on NISQ quantum processors. We choose an initial state $|0\rangle^{\otimes N}$, followed by a single layer of the square root of the Hadamard gate ($\sqrt{H_\mathrm{d}}$) on every qubit. Then, we repeat p layers composed of parametrized single qubit rotations and a set of two-qubit entangling gates (see Fig.1a). The single qubit rotations are either chosen randomly to be around the $\{x, y, z\}$ axis, or fixed to a specific axis. The two-qubit entangling gates are either CNOT, CPHASE or $\sqrt{\mathrm{iSWAP}}$ gates (see Fig.1b), that are common native gates in current quantum processors [36]. The entangling gates in each layer are arranged in either a nearest-neighbor chain topology (CHAIN), all-to-all connections (ALL) or in an alternating nearest-neighbor fashion (ALT) (see Fig.1c). The numerical calculations are performed using Yao [37].

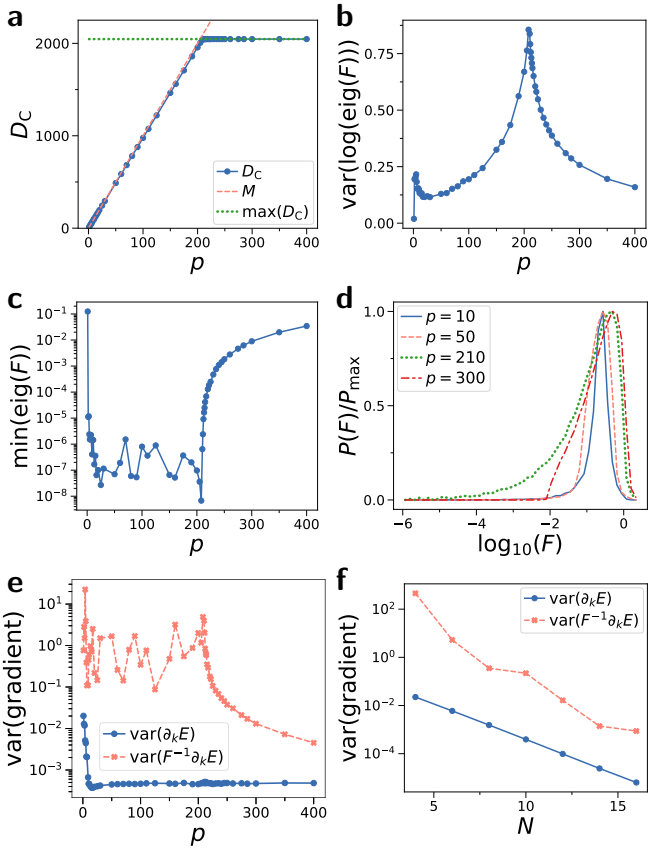


FIG. 3. Properties of PQC consisting of p layers of randomly chosen x, y, z rotations, followed by CNOT gates in a chain topology (see Fig.1) for $N = 10$ qubits. **a)** The parameter dimension D_C of the PQC (determined via Eq.13) scales linearly with p, until it levels at a characteristic value $p_c \approx 210$. **b)** Variance of the logarithm of the non-zero eigenvalues of $\mathcal{F}$. The variance peaks around $p \approx p_c$. **c)** Minimal non-zero eigenvalue of $\mathcal{F}$ against p. It increases for $p > p_c$. **d)** Histogram of logarithm of eigenvalues of Fisher information matrix $\mathcal{F}$. The width of the distribution increases with p, with a pronounced tail at small $\mathcal{F}$ developing around $p \approx p_c$, which disappears for $p > p_c$. **e)** Variance of the gradient $\mathrm{var}(\partial_k E)$ and QNG $\mathrm{var}(\mathcal{F}^{-1}\partial_k E)$ in respect to the Hamiltonian $H = \sigma_1^z \sigma_2^z$. The gradient decays until $p \approx 20$, after which it remains constant. The QNG remains larger than the regular gradient, but decreases for $p > p_c$. **f)** Variance of gradients and QNG for varying qubit number N for depth $p = 2N$, showing approximate exponential decrease with N.

V. RESULTS

As a demonstration of our methods, we provide an in-depth characterization of a PQC consisting of randomly chosen x, y, z rotations and CNOT gates in a chain topology as function of number of layers p in Fig.3. The parameter dimension D_C (i.e. number of independent parameters of the quantum state that can be expressed by the PQC) increases linearly with p in Fig.3a, until it reaches the maximal possible value for $D_C = 2^{N+1} - 2$ at a characteristic number of layers p_c. This point is reflected in the spectrum of the QFI $\mathcal{F}$, averaged over random instances of the PQC (see Fig.3b-d). Most notably, the variance of the logarithm of the non-zero eigenvalues reaches a maximum for p_c (Fig.3b). Further, the minimum taken over all eigenvalues becomes minimal (Fig.3c). We can see this more clearly in the distribution of eigenvalues (Fig.3d). With increasing p, the distribution becomes broader, with a pronounced tail of small eigenvalues of $\mathcal{F}$ appearing close to the transition at p_c. Above the transition $p > p_c$, the small eigenvalues suddenly disappear from the distribution. We investigate the variance of the gradient and QNG in Fig.3e for the two-qubit Hamiltonian $H = \sigma_1^z \sigma_2^z$. The variance of the regular gradient decays with p, reaching a minimum around $p \approx 20$ [7], upon which it remains constant. The variance of the QNG remains larger than the regular gradient, however the QNG decays for $p > p_c$. In Fig.3f, we numerically find that variance of both regular gradient and QNG vanish exponentially with increasing number of qubits N, demonstrating the barren plateau problem. In the Appendix D, we show that the same result is found also for more complicated Hamiltonians such as the transverse Ising model.

In Fig.4, we compare different types of PQCs with different entangling gates and arrangements. We note that all circuits show the same qualitative behavior regarding the transition in the QFI (see Fig.3 and Appendix C) as well as suffer from exponential decrease of the variance of the gradient with increasing number of qubits. However, key differences in the different PQCs appear. We show the variance of the gradient for the Hamiltonian $H = \sigma_1^z \sigma_2^z$ in Fig.4a,c,e for different arrangements of the entangling gates (CHAIN, ALL, ALT) as well as different types of entangling gates (CNOT, CPHASE, $\sqrt{\mathrm{iSWAP}}$). The variance decays with increasing p, until it reaches a constant level, the value of which is the same for all gates and arrangements. However, CPHASE requires the most layers p to converge, followed by $\sqrt{\mathrm{iSWAP}}$ and CNOT. Fig.4b,d,f shows the redundancy R, which is the fraction of redundant parameters of the PQC. It quickly reaches a constant level with increasing p. $\sqrt{\mathrm{iSWAP}}$ has consistently low R, while for CNOT it varies depending on the arrangement of entangling gates. For CPHASE, we have consistently larger R. This can be easily understood when considering that z rotations commute with the entangling CPHASE layer. When two z rotations appear consecutively on the same qubit, they yield a redundant parameter. R for CNOT depends highly on the entangling gates arrangement.

We note that for these PQCs the number of layers p_c at which the transition of the QFI occurs can be estimated from the value of redundancy R. We find $p_c \approx (1 - R_C)(2^{N+1} - 2)/b$, where R_C is the value of R for sufficiently large p and b is the number of parameterized rotations per layer. The eigenvalue spectrum of these PQCs and further types of PQCs are discussed in Appendix C.

In Fig.5, we fix the single-qubit rotations around the y-

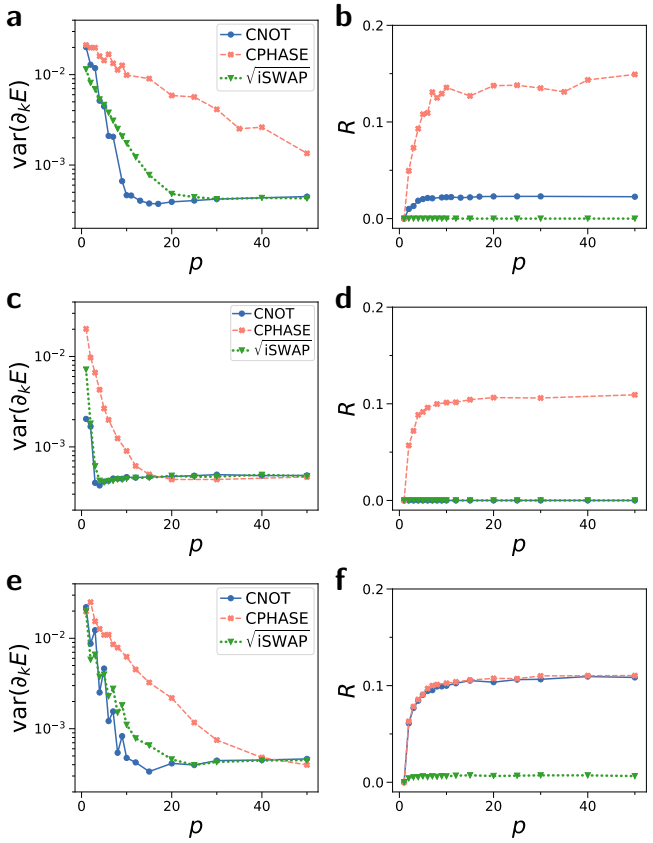


FIG. 4. Variance of gradient and redundancy of different hardware efficient PQCs plotted against number of layers p for $N = 10$ qubits. Each layer consists of single qubit-rotations as randomly chosen rotations around $\{x, y, z\}$ axis. We plot different arrangements of entangling gates (as shown in Fig.1c) with **a,b)** nearest-neighbor one-dimensional chain, **c,d)** all-to-all, **e,f)** alternating nearest-neighbor connections. **a,c,e)** Variance of the gradient $\mathrm{var}(\partial_k E)$ in respect to the Hamiltonian $H = \sigma_1^z\sigma_2^z$. **b,d,f)** Redundancy R (Eq. (3)), which is the fraction of redundant parameters of the PQC.

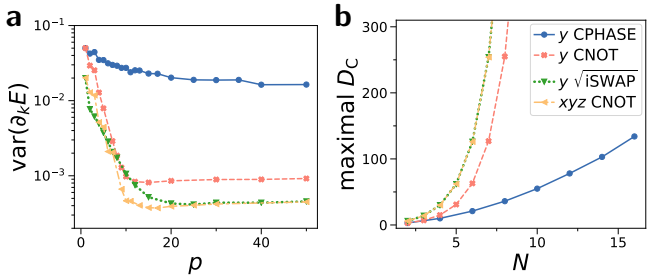


FIG. 5. Capacity of PQCs with y rotations and different entangling gates. The entangling layer is arranged as a nearest-neighbor one-dimensional chain. Three of the PQCs have y rotations, and as reference we show a PQC with randomized x, y or z rotations and CNOT gates. **a)** Variance of the gradient $\mathrm{var}(\partial_k E)$ in respect to the Hamiltonian $H = \sigma_1^z\sigma_2^z$. **b)** Maximal parameter dimension D_C of the PQCs as function of number of qubits N. For y CPHASE we find an approximate powerlaw $D_C \propto N^2$.

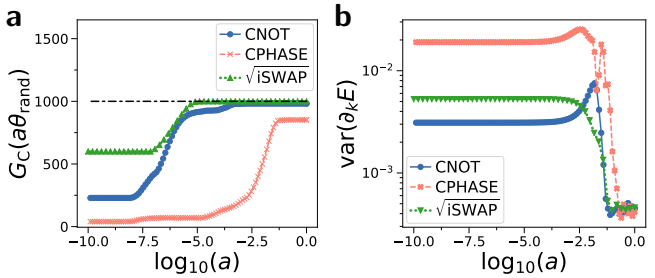


FIG. 6. Tuning the PQC parameters $\theta = a\theta_{\mathrm{random}}$, where $a = (0, 1]$ and $\theta_{\mathrm{random}} \in [0, 2\pi)$ for circuits composed of random x, y and z rotations and entangling gates arranged in a chain configurations. **a)** The effective quantum dimension G_C as function of $\log_{10}(a)$. Black dashed-dotted line is number of parameters M. **b)** Variance of the gradient $\mathrm{var}(\partial_k E)$ in respect to Hamiltonian $H = \sigma_1^z\sigma_2^z$. All plots show number of layers $p = 100$ and $N = 10$ qubits.

axis and investigate different entangling gates arranged in a nearest-neighbor one-dimensional chain. Depending on the choice of entangling gates, we find that the variance of the gradient for $H = \sigma_1^z\sigma_2^z$ decays to a different constant level with increasing p (see Fig.5a). y $\sqrt{\mathrm{iSWAP}}$ matches the variance found in Fig.3e, whereas y CNOT and y CPHASE have higher variance. In Fig.5b we show the maximal D_C for many layers p. D_C scales exponentially for y CNOT ($D_C \propto 2^N$) and y $\sqrt{\mathrm{iSWAP}}$ ($D_C \propto 2^{N+1}$), whereas for y CPHASE we find numerically an approximate quadratic scaling $D_C \propto N^2$.

In Fig.6 we show how G_C and the variance of the gradient for $H = \sigma_1^z\sigma_2^z$ changes when tuning the parameters of a PQC defined as $U(a\theta_{\mathrm{random}})|0\rangle$, $\theta_{\mathrm{random}} \in [0, 2\pi)$, $a \in [0, 1]$. When adjusting $a = 0$ to $a = 1$, this corresponds to changing the PQC from parameters all zero to a PQC with random parameters. As example, we show a PQC consisting of layered randomly chosen single qubit rotations around x,y,z axis and entangling gates arranged in a chain. In Fig.6a, we show G_C for different types of entangling gates. G_C increases with a, reaching the parameter dimension D_C for $a = 1$. CNOT and $\sqrt{\mathrm{iSWAP}}$ increase faster with a compared to the PQC with CPHASE gates. In Fig.6b, the variance of the gradient decreases sharply once a particular a is reached. Note that there is a specific range of parameters $\log_{10}(a) \approx -2.5$ where the PQCs have nearly maximal G_C and the variance of gradients remains large.

In Fig.7 we show the scaling of $G_C(\theta = 0, N)$ with number of qubits N for a PQC with entangling gates in a chain arrangement initialized with $\theta = 0$, corresponding to the point $a = 0$ in Fig.6. Numerically, we find linear scaling of $G_C(\theta = 0, N)$ for CPHASE entangling gates, quadratic scaling for CNOT gates and higher order polynomial or even exponential scaling for $\sqrt{\mathrm{iSWAP}}$ gates.

As an application, we propose Algorithm 1 to remove

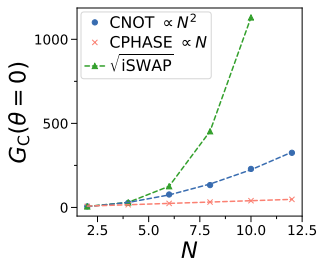


FIG. 7. Effective quantum dimension $G_C(\theta = 0)$ plotted against number of qubits N for a circuit consisting of randomly chosen parametrized rotations around x, y or z axis with parameters $\theta = 0$, and two-qubit entangling gates arranged in a nearest-neighbor chain. We compare CNOT, CPHASE and $\sqrt{\text{iSWAP}}$ entangling gates. From numerical results, we find G_C scales quadratically for CNOT gates, linearly for CPHASE gates and higher order polynomial or even exponential scaling for $\sqrt{\text{iSWAP}}$. Number of layers p is chosen such that $G_C(\theta = 0)$ is maximized.

redundant parameters from a PQC C. The algorithm calculates the eigenvectors of the QFI with eigenvalue zero. Parameters which have a non-zero amplitude in the eigenvectors can potentially be removed from the PQC without changing its expressive power. The algorithm removes one redundant gate and removes the corresponding entry in the QFI, then re-calculates the eigenvectors of the QFI. These steps are repeated until no redundant gates are left. The resulting pruned PQC C_{pruned} has as many parameters as the parameter dimension D_C of the original PQC. We demonstrate our algorithm on the CPHASE-CHAIN PQC in Appendix F and find a substantial reduction of parameters without affecting D_C.

Algorithm 1: Prune PQC of redundant parameters

Input : PQC C, QFI $\mathcal{F}^C(\theta_{\text{random}})$, number of parameters N^C_{param}, $D_C < N^C_{\text{param}}$, empty set $\mathcal{K} = \{\}$

Output: Pruned PQC C_{pruned} with $N^{C_{\text{pruned}}}_{\text{parameters}} \approx D_C$

do
- Get eigenvalues $\lambda^{(i)}$ of $\mathcal{F}^C(\theta_{\text{random}})$ sorted in ascending order, eigenvectors $\alpha^{(i)}$ and rank r
- Calculate $\boldsymbol{\beta}_j = \sum_{i=1}^{N_{\text{param}}-r} |\alpha^{(i)}_j|^2$ where $\alpha^{(i)}$ denotes the i-th eigenvector with corresponding eigenvalue $\lambda^{(i)} = 0$
- Pick largest index k such that $\boldsymbol{\beta}_k \neq 0$
- Update $\mathcal{F}^C(\theta_{\text{random}})$ by removing row k and column k
- Add k to set $\mathcal{K}$

while $\mathcal{F}^C(\theta_{random})$ *has non-zero eigenvalues*;

Removing parameters corresponding to set $\mathcal{K}$ from C gives pruned PQC C_{pruned}

VI. DISCUSSION

We investigated the capacity and trainability of hardware efficient PQCs using the quantum geometric structure of the parameter space. We introduced the notion of parameter dimension D_C and effective quantum dimension G_C which are global and local measures respectively of the space of quantum states that can be accessed by the PQC. Both can be derived from the QFI. We applied these concepts on PQCs composed of layers of single-qubit rotations and different types of entangling gates arranged in various geometries. For comparable circuit depth p, we find strong numerical evidence that PQCs constructed from CNOT or $\sqrt{\text{iSWAP}}$ gates have lower variance of the gradient, and thus higher expressibility compared to PQCs with CPHASE gates. While two-qubit gates such as CNOT and CPHASE gate can be expressed as each other by applying specific single-qubit rotations, the PQCs we use only have a limited amount of single qubit rotations and thus the choice and arrangement of two-qubit gates strongly affects the expressibility of the PQC. Without loosing generality, we study the properties of the variance of the gradient using a two-qubit Hamiltonian $H = \sigma^z_1\sigma^z_2$, where we take the variance over an ensemble of randomized PQCs. The variance of the gradient of a generic Hamiltonian $H = \sum_i b_i H_i$ that consists of a polynomial number of Pauli operators H_i shows the same exponential decay as the two-qubit Hamiltonian [7, 8], which we demonstrate for a many-body Hamiltonian in the Appendix D.

For a specific type of PQC composed of y rotations and CPHASE gates, D_C scales only quadratically with number of qubits, which may imply that this PQC can be efficiently simulated on classical computers. We find that the redundancy of parameters varies strongly depending on the configuration of the PQC as well as the type of gates.

The effective quantum dimension G_C reveals the expressive power of a PQC by local variations around a specific parameter set. We find that depending on the entangling gates, G_C shows widely different scaling with number of qubits, with the largest value found for $\sqrt{\text{iSWAP}}$ gates. While we only studied the case $\theta = 0$, PQCs with correlated parameters could feature similar behavior [19]. Tuning the parameters of a PQC from zero to a random set of parameters yields a crossover from large gradients and small G_C to vanishing gradients and large G_C. For the PQCs investigated, we can find a range of parameters that combines large gradients with a nearly maximal G_C, which could be an optimal starting point for gradient based optimisation. Trade-offs between the expressibility of a circuit and the magnitude of its gradients are a key challenge in finding good initialization strategies [34].

When increasing the number of layers p to a value p_c, a transition occurs in the QFI when D_C reaches its maximal possible value. The transition is characterized by a disappearance of small eigenvalues of the QFI and a peak in the variance of the logarithm of eigenvalues. This

peak may be related to a transition in the optimization landscape of control theory. When the system becomes overparameterized with more parameters than degrees of freedom, the optimization landscape changes from being spin-glass like with many near-degenerate minima to one with many degenerate global minima [38, 39]. This peak in the QFI could be used to identify the transition. The overparameterized regime may be useful for mitigating the effect of noise [40, 41]. For deep circuits $p > p_c$, the transition leads to a decay of the QNG as small eigenvalues are suppressed. For shallow circuits $p < p_c$, the QNG can be orders of magnitude larger in value compared to the regular gradient, however our numerical results suggest that both regular gradient and QNG decrease exponentially with number of qubits. Thus, the QNG most likely cannot help to solve the barren plateau problem. This contrasts the natural gradient in classical machine learning, which is known to be able to overcome the plateau phenomena that leads to a slow down of optimization [35].

Imaginary-time evolution and variational quantum simulation use a matrix related to the QFI to update the parameters of the PQC [31, 42]. The effective quantum dimension G_C could give major insights on the convergence properties of these algorithms. Recent proposals for adaptively generated ansätze could benefit from the QFI by taking the geometry of the PQC into account when designing PQCs [21].

We demonstrated an algorithm to systematically reduce the number of parameters and depth of PQCs while keeping the parameter dimension constant. This algorithm can be immediately applied to PQCs used in VQAs to reduce the number of parameters M without sacrificing expressive power. Commonly used PQCs often contain more parameters than necessary. Removing them reduces the computational effort for calculating the gradient as well as the QFI necessary for the QNG, which has been shown to be highly beneficial for training [32, 43]. When compared to ordinary gradient descent, the sampling overhead of training using the QFI and QNG is constant asymptotically for both an increasing number of iterations and number of qubits, as has been proven recently [43]. Furthermore, training with the QNG has a reduced total cost since it approaches the optimum faster [43]. Thus, NISQ algorithms that use the QFI to update their parameters accomplish faster training than ordinary gradient descent. Our algorithm reduces the cost of calculating QFI in each iteration of the training by truncating the size of the QFI. As the QFI is a matrix, removing a single parameter already reduces the number of elements to measure by M. Further, with our approach one can lower the number of parameterized gates needed to run the VQA, which is especially important for NISQ era algorithms.

The QFI has widespread use in quantum metrology [44] and quantum computing [31, 45, 46]. To facilitate its application, various methods to calculate the QFI on quantum computers have been developed and are continuously improved [46, 47], which we review in Appendix G. The most commonly applied methods are the shift-rule [47, 48], the Hadamard test [45, 49, 50] and direct measurement methods [51]. For these approaches, the number of circuits to measure scales as the square $\left(M^2\right)$ of the number of parameters (M). Various approximations for the QFI have been proposed [31, 52–54]. Improved methods for numerical simulation of the QFI are being developed as well [55]. We provide code that can simulate the QFI for 26 qubits on a desktop computer [56]. We note that calculations relying on a reduced number of qubits or layers can help to design better PQCs. Most commonly used PQCs are constructed according to specific rules in a layer-wise fashion. By evaluating the effective dimension within smaller PQCs, one can identify rules and patterns for constructing PQCs with few redundant parameters. Then, one can extrapolate these rules to PQCs with many qubits and layers.

During the training of a PQC, the eigenvalue spectrum of the QFI can gain specific features, as has been shown for restricted Boltzmann machines [57]. We show that the PQCs have a characteristic eigenvalue spectra depending on the type of gates and their arrangement (see Appendix E). The eigenvalues hold important information about the trainability and generalization of a model. For example, a model that generalizes well is known to have a low effective dimension in classical machine learning [29]. It would be interesting to study in what way these statements translate to quantum machine learning. The eigenvalues of the Hessian could be applied as well [58]. Further, connections to complementary measures of capacity based on classical Fisher information [59] and memory capacity [60] respectively could be explored.

While we studied hardware efficient PQCs, some of our results can be carried over to other types of PQCs. The transition in the QFI spectrum we observed could be used to characterize when a PQC is overparameterized. Further, $G_C(\theta)$ can be used to determine the amount of quantum states that can be reached by varying the parameters of PQCs. It would be straightforward to extend our concepts to evaluate the capacity and trainability of noisy PQCs [61], convolutional PQCs [62], optimal control [63], quantum metrology [64] and programmable analog quantum simulators [65].

Python and Julia code for the numerical calculations performed in this work are available at [56].

Acknowledgements— This work is supported by a Samsung GRC project and the UK Hub in Quantum Computing and Simulation, part of the UK National Quantum Technologies Programme with funding from UKRI EPSRC grant EP/T001062/1. We are grateful to the National Research Foundation and the Ministry of Education, Singapore for financial support.

[1] J. Preskill, Quantum **2**, 79 (2018).
[2] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek, and A. Aspuru-Guzik, arXiv:2101.08448 (2021).
[3] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. Obrien, Nature communications **5**, 4213 (2014).
[4] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, New Journal of Physics **18**, 023023 (2016).
[5] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, *et al.*, arXiv preprint arXiv:2012.09265 (2020).
[6] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. Sawaya, *et al.*, Chemical reviews **119**, 10856 (2019).
[7] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Nature communications **9**, 4812 (2018).
[8] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, Nature Communications **12**, 1 (2021).
[9] C. O. Marrero, M. Kieferová, and N. Wiebe, arXiv:2010.15968 (2020).
[10] S. Wang, E. Fontana, M. Cerezo, K. Sharma, A. Sone, L. Cincio, and P. J. Coles, arXiv:2007.14384 (2020).
[11] L. Bittel and M. Kliesch, arXiv:2101.07267 (2021).
[12] H.-Y. Huang, K. Bharti, and P. Rebentrost, arXiv:1909.07344 (2019).
[13] K. Bharti, arXiv:2009.11001 (2020).
[14] K. Bharti and T. Haug, arXiv:2011.06911 (2020).
[15] K. Bharti and T. Haug, arXiv:2010.05638 (2020).
[16] T. Haug and K. Bharti, arXiv:2011.14737 (2020).
[17] J. W. Z. Lau, K. Bharti, T. Haug, and L. C. Kwek, arXiv:2101.07677 (2021).
[18] K. H. Lim, T. Haug, L. C. Kwek, and K. Bharti, arXiv preprint arXiv:2104.01931 (2021).
[19] T. Volkoff and P. J. Coles, Quantum Science and Technology **6**, 025008 (2021).
[20] E. Grant, L. Wossnig, M. Ostaszewski, and M. Benedetti, Quantum **3**, 214 (2019).
[21] H. R. Grimsley, S. E. Economou, E. Barnes, and N. J. Mayhall, Nature communications **10**, 1 (2019).
[22] A. Skolik, J. R. McClean, M. Mohseni, P. van der Smagt, and M. Leib, Quantum Machine Intelligence **3**, 1 (2021).
[23] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, Nature **549**, 242 (2017).
[24] K. Nakaji and N. Yamamoto, arXiv:2005.12537 (2020).
[25] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, Advanced Quantum Technologies **2**, 1900070 (2019).
[26] S. E. Rasmussen, N. J. S. Loft, T. Bækkegaard, M. Kues, and N. T. Zinner, Advanced Quantum Technologies **3**, 2000063 (2020).
[27] L. Funcke, T. Hartung, K. Jansen, S. Kühn, and P. Stornati, arXiv preprint arXiv:2011.03532 (2020).
[28] D. J. MacKay, in *Advances in neural information processing systems* (1992) pp. 839–846.
[29] W. J. Maddox, G. Benton, and A. G. Wilson, arXiv:2003.02139 (2020).
[30] N. Yamamoto, arXiv:1909.05074 (2019).
[31] J. Stokes, J. Izaac, N. Killoran, and G. Carleo, Quantum **4**, 269 (2020).
[32] D. Wierichs, C. Gogolin, and M. Kastoryano, Physical Review Research **2**, 043246 (2020).
[33] Y. Du, M.-H. Hsieh, T. Liu, and D. Tao, Phys. Rev. Res. **2**, 033125 (2020).
[34] Z. Holmes, K. Sharma, M. Cerezo, and P. J. Coles, arXiv preprint arXiv:2101.02138 (2021).
[35] S.-i. Amari, *Information geometry and its applications*, Vol. 194 (Springer, 2016).
[36] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, Applied Physics Reviews **6**, 021318 (2019).
[37] X.-Z. Luo, J.-G. Liu, P. Zhang, and L. Wang, Quantum **4**, 341 (2020).
[38] M. Bukov, A. G. R. Day, D. Sels, P. Weinberg, A. Polkovnikov, and P. Mehta, Phys. Rev. X **8**, 031086 (2018).
[39] H. A. Rabitz, M. M. Hsieh, and C. M. Rosenthal, Science **303**, 1998 (2004).
[40] E. Fontana, N. Fitzpatrick, D. M. Ramo, R. Duncan, and I. Rungger, Physical Review A **104**, 022403 (2021).
[41] E. Fontana, M. Cerezo, A. Arrasmith, I. Rungger, and P. J. Coles, arXiv:2011.08763 (2020).
[42] S. McArdle, T. Jones, S. Endo, Y. Li, S. C. Benjamin, and X. Yuan, npj Quantum Information **5**, 1 (2019).
[43] B. van Straaten and B. Koczor, PRX Quantum **2**, 030324 (2021).
[44] J. Liu, H. Yuan, X.-M. Lu, and X. Wang, Journal of Physics A: Mathematical and Theoretical **53**, 023001 (2019).
[45] Y.-X. Yao, N. Gomes, F. Zhang, C.-Z. Wang, K.-M. Ho, T. Iadecola, and P. P. Orth, PRX Quantum **2**, 030307 (2021).
[46] J. J. Meyer, arXiv:2103.15191 (2021).
[47] D. Wierichs, J. Izaac, C. Wang, and C. Y.-Y. Lin, arXiv:2107.12390 (2021).
[48] A. Mari, T. R. Bromley, and N. Killoran, Physical Review A **103**, 012405 (2021).
[49] Y. Li and S. C. Benjamin, Physical Review X **7**, 021050 (2017).
[50] X. Yuan, S. Endo, Q. Zhao, Y. Li, and S. C. Benjamin, Quantum **3**, 191 (2019).
[51] K. Mitarai and K. Fujii, Physical Review Research **1**, 013006 (2019).
[52] J. Gacon, C. Zoufal, G. Carleo, and S. Woerner, arXiv:2103.09232 (2021).
[53] M. Cerezo, A. Sone, J. L. Beckey, and P. J. Coles, Quantum Science and Technology **6** (2021).
[54] A. Rath, C. Branciard, A. Minguzzi, and B. Vermersch, arXiv:2105.13164 (2021).
[55] T. Jones, arXiv preprint arXiv:2011.02991 (2020).
[56] T. Haug, "Quantum geometry of parametrized quantum circuits," https://github.com/txhaug/quantum-geometry.
[57] C.-Y. Park and M. J. Kastoryano, Physical Review Research **2**, 023232 (2020).
[58] P. Huembeli and A. Dauphin, Quantum Science and Technology **6** (2021).
[59] A. Abbas, D. Sutter, C. Zoufal, A. Lucchi, A. Figalli, and S. Woerner, Nature Computational Science **1**, 403 (2021).

[60] L. G. Wright and P. L. McMahon, arXiv:1908.01364 (2019).
[61] B. Koczor and S. C. Benjamin, arXiv preprint arXiv:1912.08660 (2019).
[62] I. Cong, S. Choi, and M. D. Lukin, Nature Physics **15**, 1273 (2019).
[63] A. B. Magann, C. Arenz, M. D. Grace, T.-S. Ho, R. L. Kosut, J. R. McClean, H. A. Rabitz, and M. Sarovar, P R X Quantum **2**, 010101 (2020).
[64] J. J. Meyer, J. Borregaard, and J. Eisert, npj Quantum Information **7**, 1 (2021).
[65] V. Bastidas, T. Haug, C. Gravel, L.-C. Kwek, W. Munro, and K. Nemoto, arXiv:2009.00823 (2020).
[66] S.-I. Amari, Neural computation **10**, 251 (1998).
[67] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, Phys. Rev. A **98**, 032309 (2018).
[68] O. Kyriienko and V. E. Elfving, arXiv:2108.01218 (2021).
[69] A. F. Izmaylov, R. A. Lang, and T.-C. Yen, arXiv preprint arXiv:2107.08131 (2021).
[70] A. K. Ekert, C. M. Alves, D. K. Oi, M. Horodecki, P. Horodecki, and L. C. Kwek, Physical review letters **88**, 217901 (2002).

Appendix A: Variational quantum eigensolver

The core idea of Variational quantum eigensolver (VQE) is to find the ground state of a Hamiltonian H by minimizing the parameters θ of a PQC in regards to an objective function that represents the energy of a given Hamiltonian $E(\theta) = \langle 0|U^\dagger(\theta)HU(\theta)|0\rangle$ [3]. The minimisation is performed with a classical optimisation algorithm, whereas the energy is measured on a quantum device. According to the Ritz variational principle, the objective function is lower bounded by the ground state energy of H, i.e. $E(\theta) \geq E_\mathrm{g}$, where E_g is the true ground state of H.

Appendix B: Quantum Fisher information metric

For VQE, the objective function is updated in hybrid classical-quantum algorithm in an iterative manner. At step n of the procedure, the objective function is evaluated on the quantum computer for a given θ_n. Based on the result, a classical computer selects the next choice θ_{n+1} such that it (hopefully) decreases the objective function. A common scheme to update parameters is ordinary gradient descent

$$\theta_{n+1} = \theta_n - \eta \frac{\partial E(\theta)}{\partial \theta}\,, \tag{B1}$$

where η is a small coefficient and $\partial E(\theta)/\partial\theta$ is the gradient of the objective function.

The above update rule assumes that the parameter space for θ is a flat Euclidian space. However, in general this is not the case, as the underlying PQC and cost function do not have such simple forms. Recent studies have proposed the quantum natural gradient (QNG), inspired from the natural gradient in classical machine learning [66], to minimize the objective function [30, 31]. The main idea is to use information about how fast the quantum state changes when adjusting the parameter θ in a particular direction. Optimisation with the natural gradient updates the parameters according to

$$\theta_{k+1} = \theta_k - \eta_k \mathcal{F}^{-1}(\theta)\frac{\partial E(\theta)}{\partial \theta}, \tag{B2}$$

where $\mathcal{F}(\theta)$ is the Fubini-Study metric tensor or quantum Fisher information metric (QFI)

$$\mathcal{F}_{ij} = \mathrm{Re}(\langle\partial_i\psi|\partial_j\psi\rangle - \langle\partial_i\psi|\psi\rangle\langle\psi|\partial_j\psi\rangle)\,, \tag{B3}$$

where $|\partial_i\psi\rangle = \frac{\partial}{\partial\theta_i}|\psi(\theta)\rangle$ denotes the partial derivative of $|\psi(\theta)\rangle$. One can relate $\mathcal{F}(\theta)$ to the distance in the space of pure quantum states, which is the Fubini-Study distance given by

$$\mathrm{Dist}_Q\Big(|\psi(\theta)\rangle,\ |\psi(\theta+\mathrm{d}\theta)\rangle\Big)^2 = \sum_{i,j}\mathcal{F}_{ij}(\theta)\mathrm{d}\theta_i\mathrm{d}\theta_j\,, \tag{B4}$$

where $\mathrm{Dist}_Q(x,y) = |\langle x|y\rangle|^2$.

Appendix C: Further data on the PQCs

In Fig.8, we show further types of PQCs as defined in the caption. We highlight that the PQC rand(xyw) CPHASE has lower redundancy compared to rand(xyz) CPHASE. The reason is that the z rotations, which can commute with the CPHASE layer, are replaced with non-commuting $(x+y)/\sqrt{2}$ rotations. This leads to a faster decrease in the variance of the gradient as well. We also define a common type of PQC zxz CNOT, which has first been introduced in [23]. We note that while it has three rotations per qubit and layer, compared to rand(xyz) CNOT the decay of the variance of the gradient as function of p remains the same in both types of PQC. Finally, we show further examples of the transition in the QFI, visible both in the peak of the variance of the logarithm of the eigenvalues, and in the decay of the QNG.

Appendix D: Variance of gradient of Hamiltonians

The variance of the gradient shows the same exponential decay due to barren plateaus for any Hamiltonian that consists of a sum of a polynomial number of Pauli operators [7]. To demonstrate this, we compare the simple two-qubit Hamiltonian $H = \sigma_1^z\sigma_2^z$ and the transverse Ising model

$$H_\mathrm{ising} = \sum_{n=1}^{N}\sigma_n^z\sigma_{n+1}^z + h\sum_{n=1}^{N}\sigma_n^x \tag{D1}$$

with $h = 1$. The variance of the gradient in respect to the Hamiltonian for different PQCs is shown in Fig.9.

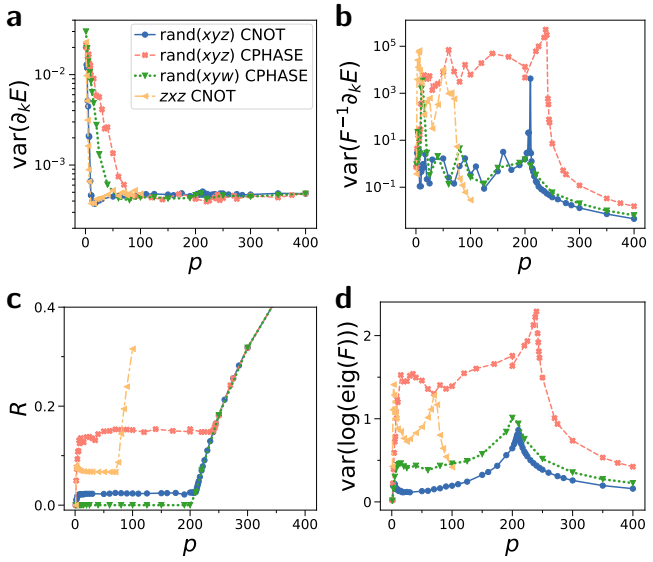


FIG. 8. Properties of further PQCs plotted against layers p for $N = 10$ qubits. PQCs have nearest-neighbor chain entangling layers. We define the type of PQCs in the legend: rand(xyz) denotes randomized single-qubit rotations around $\{x, y, z\}$ axis. rand(xyw) denotes randomized single-qubit rotations around $\{x, y, (x+y)/\sqrt{2})\}$ axis. zxz denotes that for every layer there are three single-qubit rotations, around z, x and z axis. **a)** Variance of the gradient $\mathrm{var}(\partial_k E)$ in respect to the Hamiltonian $H = \sigma_1^z\sigma_2^z$. **b)** Variance of QNG $\mathrm{var}(\mathcal{F}^{-1}\partial_k E)$. **c)** Redundancy R of parameters of the PQCs. **d)** Variance of the logarithm of the eigenvalues of the QFI.

We find that the variance of the gradient divided by the number of terms in the Hamiltonian has nearly the same value for both the two-qubit Hamiltonian and the transverse Ising Hamiltonian. As we take the variance over an ensemble of randomized PQCs, it does not matter which Pauli operator we use to calculate the variance.

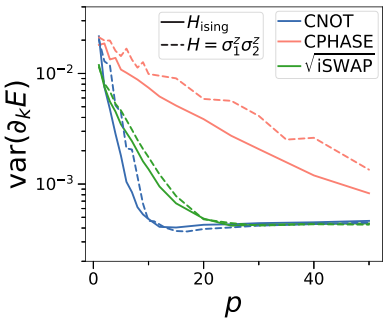


FIG. 9. Variance of gradient for $H = \sigma_1^z\sigma_2^z$ (solid curves) and transverse Ising Hamiltonian (dashed curves) for different types of PQC with entangling gates in a CHAIN configuration for varying number of layers p. For the transverse Ising Hamiltonian, we divided the variance of the gradient by the number of terms in the Hamiltonian. We use $N = 10$ qubits.

Appendix E: Histograms of eigenvalues

In Fig.10 we show the distribution of eigenvalues for the PQCs of Fig.4 in the main text. We find that a characteristic spectrum for the different PQC types. Note that CPHASE appears to have more pronounced tails in all cases.

Appendix F: Pruning PQCs of redundant parameters

We apply the Algorithm 1 of the main text to prune a PQC of redundant parameters, i.e. parameters which can be removed without changing the parameter dimension D_C and thus the expressiveness of the circuit. In Fig.11a, we show the initial PQC, which is the CPHASE CHAIN PQC. In Fig.11b, we apply Algorithm 1 of the main text to remove redundant parameters, and reduce the number of unitaries within the circuit substantially. The parameter dimension $D_C = D_{C_{\text{pruned}}} = 126$ remains constant before and after pruning.

Appendix G: Measuring the quantum Fisher information metric

The QFI has found widespread use in quantum metrology [44] and quantum computing [31, 46, 49]. As such, various methods to calculate the QFI have been proposed and are continuously improved and developed. We now proceed to review methods for calculating the $M \times M$ dimensional positive-semidefinite QFI $\mathcal{F}(\boldsymbol{\theta})$ or Fubini-Study metric

$$\mathcal{F}_{ij}(\theta) = \mathrm{Re}(\langle\partial_i\psi|\partial_j\psi\rangle - \langle\partial_i\psi|\psi\rangle\langle\psi|\partial_j\psi\rangle)\,. \tag{G1}$$

Shift-rule— The QFI for pure states can be reformulated as the second order derivative of the fidelity of a quantum state [47]

$$\mathcal{F}_{ij}(\boldsymbol{\theta}) = -\frac{1}{2}\partial_i\partial_j\,|\langle\psi(\boldsymbol{\theta})|\psi(\boldsymbol{\theta}_0)\rangle|^2\,\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}\,. \tag{G2}$$

A straightforward way to calculate the QFI is thus calculating the Hessian of the fidelity. First, we explain how to calculate fidelities using the inversion test. The fidelity $K = |\langle\psi_1|\psi_2\rangle|^2$ of two quantum states $|\psi_1\rangle = U(\boldsymbol{\theta}_1)|0\rangle$ and $|\psi_2\rangle = U(\boldsymbol{\theta}_2)|0\rangle$ is computed by preparing the first state followed by the inverse of the second state $|\psi_{12}\rangle = U^\dagger(\boldsymbol{\theta}_2)U(\boldsymbol{\theta}_1)|0\rangle$. Then, the fidelity is measured as the probability of sampling the all-zero state $|\langle\psi_1|\psi_2\rangle|^2 = |\langle 0|\psi_{12}\rangle|^2$. Now that we know how to calculate the fidelity, we can now proceed to calculate its gradients as well. For quantum computers, the shift rule is a practical way to calculate gradients [67]. It directly applies to all PQCs where the parameterized rotations are of the form $\exp(-i\theta G)$, where the generator G is a Pauli string. Recently, the shift-rule has been also extended

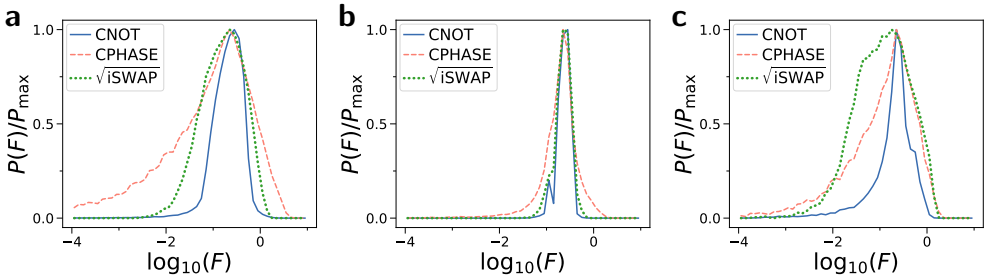


FIG. 10. Distribution of eigenvalues of QFI for PQCs shown in Fig.4 in main text. **a)** nearest-neighbor chain arrangement of entangling gates **b)** all-to-all connectivity **c)** alternating nearest-neighbor. All graphs for $N = 10$ qubits and number of layers $p = 50$.

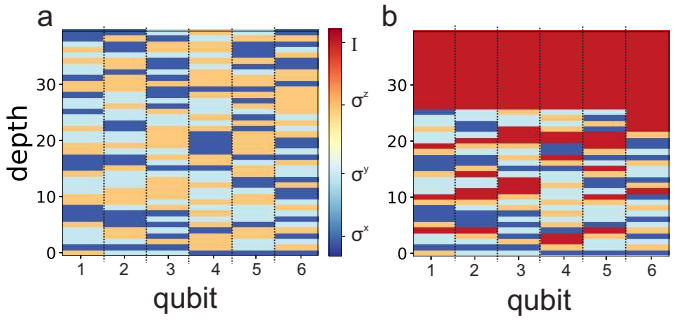


FIG. 11. Pruning of layered PQC composed of random parameterized rotations with Pauli operators $\{\sigma^x, \sigma^y, \sigma^z\}$ and CPHASE gates arranged in a nearest-neighbor chain. PQC has $p = 40$ layers and $N = 6$ qubits. The colour scheme indicates the type of parameterized rotation at a specific qubit and layer. The identity operation I is to show that no operation (identity) is applied at that particular qubit and depth, which results in pruning the redundant rotations. **a)** We show the initial PQC with $N_{\text{param}} = 240$ parameters and parameter dimension $D_C = 126$. **b)** Algorithm 1 removes redundant parameterized rotations and replaces them with identity operator I. We show the pruned PQC with $N_{\text{param}} = 126$ and same parameter dimension $D_C = 126$, reducing the PQC by 14 circuit layers and 114 parameters.

to circuits with general generators G [47, 68, 69]. The shift-rule for the QFI takes the following form [47, 48]

$$\begin{aligned}\mathcal{F}_{ij}(\boldsymbol{\theta}) = -\frac{1}{8}[&|\langle\psi(\boldsymbol{\theta}|\psi(\boldsymbol{\theta}+(\boldsymbol{e}_i+\boldsymbol{e}_j)\pi/2\rangle|^2 + \\ &- |\langle\psi(\boldsymbol{\theta}|\psi(\boldsymbol{\theta}+(\boldsymbol{e}_i-\boldsymbol{e}_j)\pi/2\rangle|^2 \\ &- |\langle\psi(\boldsymbol{\theta}|\psi(\boldsymbol{\theta}+(-\boldsymbol{e}_i+\boldsymbol{e}_j)\pi/2\rangle|^2 \\ &+ |\langle\psi(\boldsymbol{\theta}|\psi(\boldsymbol{\theta}-(\boldsymbol{e}_i+\boldsymbol{e}_j)\pi/2\rangle|^2]\,,\end{aligned}$$

where $\boldsymbol{e}_i$ is the basis vector for i-th index of parameter $\boldsymbol{\theta}$. The diagonal elements of the QFI simplify to

$$\mathcal{F}_{ii}(\boldsymbol{\theta}) = \frac{1}{4}[1 - |\langle\psi(\boldsymbol{\theta}|\psi(\boldsymbol{\theta}+\boldsymbol{e}_i\pi\rangle|^2\,. \tag{G3}$$

To determine the full QFI, we have to measure in total $2M(M-1)+M$ fidelities via the shift-rule.

Approximations of the QFI can be measured even more efficiently on quantum computers [31, 52]. For example, when unitaries within the PQC commute, one can use this to speed up the calculation. This is the case for the diagonal and block-diagonals entries of the QFI, which can be calculated in a time that scales linearly with M [31, 47].

Hadamard test— We now review an alternative approach to calculate the QFI. We assume a general ansatz for the PQC for N qubits and M parameters

$$|\psi(\theta)\rangle = \prod_{l=1}^{M} W_l R_l(\theta_l)|0\rangle^{\otimes N}\,, \tag{G4}$$

where W_i is an arbitrary unparameterized unitary and $R_l(\theta_l) = \exp(-\iota\frac{\theta_l}{2}\sigma_{n_l}^{\alpha_l})$ is a parameterized rotation with Pauli operator $\sigma_{n_l}^{\alpha_l}$ acting on qubit n_l and $\alpha_l \in \{x, y, z\}$. This ansatz includes hardware efficient PQCs as used within our manuscript.

As notation for our circuit, we define

$$U_{[l_1:l_2]} := W_{l_2}R_{l_2}\cdots W_{l_1}R_{l_1}\,. \tag{G5}$$

The derivative of a PQC $|\psi(\theta)\rangle = |\psi\rangle$ in respect to the l-th index of the parameter is given by

$$\begin{aligned}\partial_l|\psi\rangle &= U_{(l:M]}W_l\partial_l R_l(\theta_l)U_{[1:l)}|0\rangle\ \,, \\ &= U_{(l:M]}W_l R_l(\theta_l)(-\iota\frac{\sigma_{n_l}^{\alpha_l}}{2})U_{[1:l)}|0\rangle\ \,, \\ &= U_{[l:M]}(-\iota\frac{\sigma_{n_l}^{\alpha_l}}{2})U_{[1:l)}|0\rangle^{\otimes N}\,.\end{aligned}$$

We now discuss how to calculate the QFI with this ansatz. The QFI Eq. (G1) consists of two terms. The second term of the QFI is a product of two overlaps. Each overlap takes a simple form

$$\langle\psi|\partial_l\psi\rangle = -\frac{\iota}{2}\langle 0|^{\otimes N}U_{[1:l)}^\dagger\sigma_{n_l}^{\alpha_l}U_{[1:l)}|0\rangle^{\otimes N}\,, \tag{G6}$$

which can be evaluated as a measurement of the Pauli operator $\sigma_{n_l}^{\alpha_l}$ on the quantum state $U_{[1:l)}|0\rangle$. This can be easily measured by sampling in a Pauli rotated computational basis.

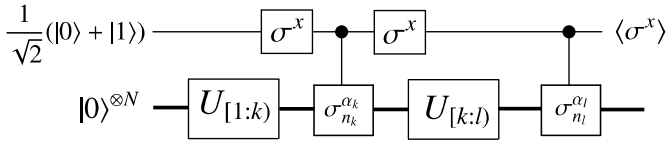


FIG. 12. Hadamard test to evaluate overlap $\mathrm{Re}[\langle\partial_k\psi|\partial_l\psi\rangle]$ for QFI.

The first term of the QFI Eq. (G1) consists of two derivatives. For $l \geq k$ and our ansatz, it is given by

$$\mathrm{Re}[\langle\partial_k\psi|\partial_l\psi\rangle] = \frac{1}{4}\mathrm{Re}[\langle 0|^{\otimes N} U^\dagger_{[1:k)}\sigma^{\alpha_k}_{n_k}U^\dagger_{[k:l)}\sigma^{\alpha_l}_{n_l}U_{[1:l)}|0\rangle^{\otimes N}]\,. \tag{G7}$$

For $l = k$, this overlap is trivial to evaluate $\mathrm{Re}[\langle\partial_l\psi|\partial_l\psi\rangle] = \frac{1}{4}$. For $l \neq k$, this overlap is not an observable and takes a complex number in general. Here, the Hadamard test can be employed [47, 50, 70]. The Hadamard test calculates overlaps of two quantum states $\langle\psi_1|\psi_2\rangle$ and can measure both real and imaginary parts. To measure Eq. (G7), we prepare an ancilla qubit in the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. The ancilla is entangled with the state $\sigma^{\alpha_l}_{n_l}U_{[k:l)}\sigma^{\alpha_k}_{n_k}U_{[1:k)}|0\rangle^{\otimes N}$ by replacing the Pauli operators $\sigma^{\alpha_l}_{n_l}$ and $\sigma^{\alpha_k}_{n_k}$ with controlled unitaries. The corresponding measurement circuit is depicted in Fig.12. Finally, we measure the expectation value of σ^x of the ancilla and find

$$\begin{aligned}\langle\sigma^x\rangle =& \mathrm{Re}[\langle 0|^{\otimes N}U^\dagger_{[1:k)}\sigma^{\alpha_k}_{n_k}U^\dagger_{[k:l)}\sigma^{\alpha_l}_{n_l}U_{[1:l)}|0\rangle^{\otimes N}] = \\ & 4\mathrm{Re}[\langle\partial_k\psi|\partial_l\psi\rangle]\end{aligned}$$

To calculate the QFI with this method, one requires $M(M-1)/2$ measurements with the Hadamard test for the terms of type $\mathrm{Re}[\langle\partial_k\psi|\partial_l\psi\rangle]$. Further, one requires M measurements of Pauli strings to get terms of type $\langle\psi|\partial_l\psi\rangle$.

The Hadamard test for our ansatz requires controlled unitaries applied on the ancilla and the qubit where the Pauli operator is acting on. In case one wants to avoid implementing controlled unitaries and the ancilla, one can replace the controlled unitaries with direct measurements [51].