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

check for updates

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

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

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.

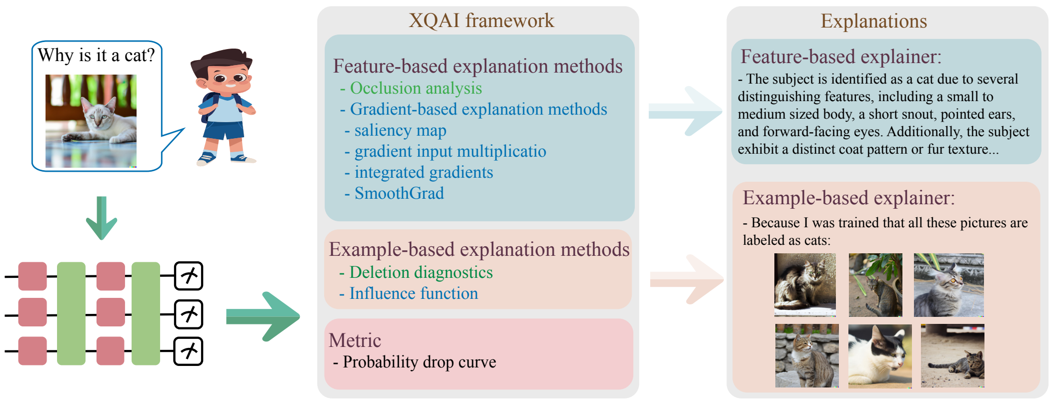

Why is it a cat?
XQAI framework
Feature-based explanation methods
- Occlusion analysis
- Gradient-based explanation methods
- saliency map
- gradient input multiplicatio
- integrated gradients
- SmoothGrad
Example-based explanation methods
- Deletion diagnostics
- Influence function
Metric
- Probability drop curve
Explanations
Feature-based explainer:
- The subject is identified as a cat due to several distinguishing features, including a small to medium sized body, a short snout, pointed ears, and forward-facing eyes. Additionally, the subject exhibit a distinct coat pattern or fur texture...
Example-based explainer:
- Because I was trained that all these pictures are labeled as cats:

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 depending 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.

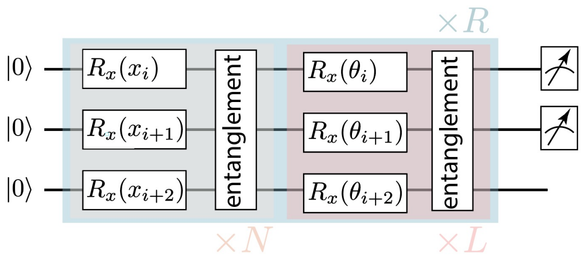


Figure 2. The circuit diagram of the QNN with data re-uploading.

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].

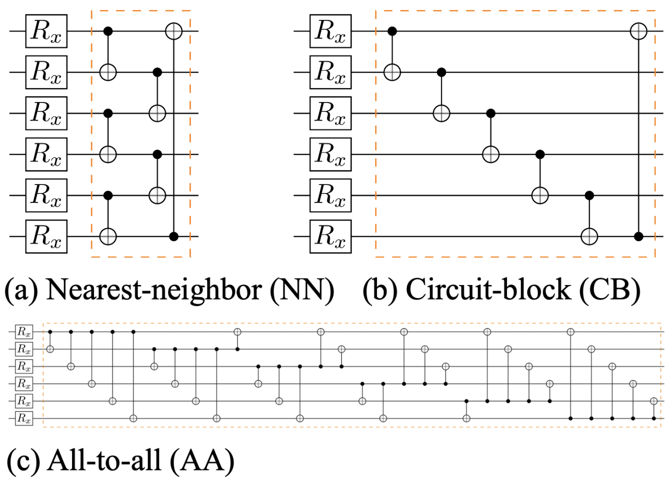


Figure 3. Three distinct types of two-qubit entangling operator layouts are presented: (**a**) nearest-neighbor (NN), (**b**) circuit-block (CB), and (**c**) all-to-all (AA). The two-qubit entangling operators are depicted within regions demarcated by an orange dashed box. Each layout corresponds to 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:

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

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 $P\big(y \mid \boldsymbol{x}_{p,q}; \theta\big) = \langle 0|U(\boldsymbol{x}_{p,q}, \boldsymbol{\theta})^{\dagger} O_y U(\boldsymbol{x}_{p,q}, \boldsymbol{\theta})|0\rangle$, 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:

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

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, $\frac{1}{n}\sum_i \nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^*(\epsilon,i),(x_i,y_i))$, and $v_i = \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^*(\epsilon,i),(x_i,y_i))$. 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

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

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

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

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 $\mathcal{L}(\boldsymbol{\theta},(|\psi\rangle, |\phi\rangle)) = 1 - |\langle\phi| U(\boldsymbol{\theta}) |\psi\rangle|^2$, where $|\langle\phi| U(\boldsymbol{\theta}) |\psi\rangle|^2$ 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 $(\tilde{x}_i, \tilde{y}_i) = (\boldsymbol{x}_i, y_i)$. 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:

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

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

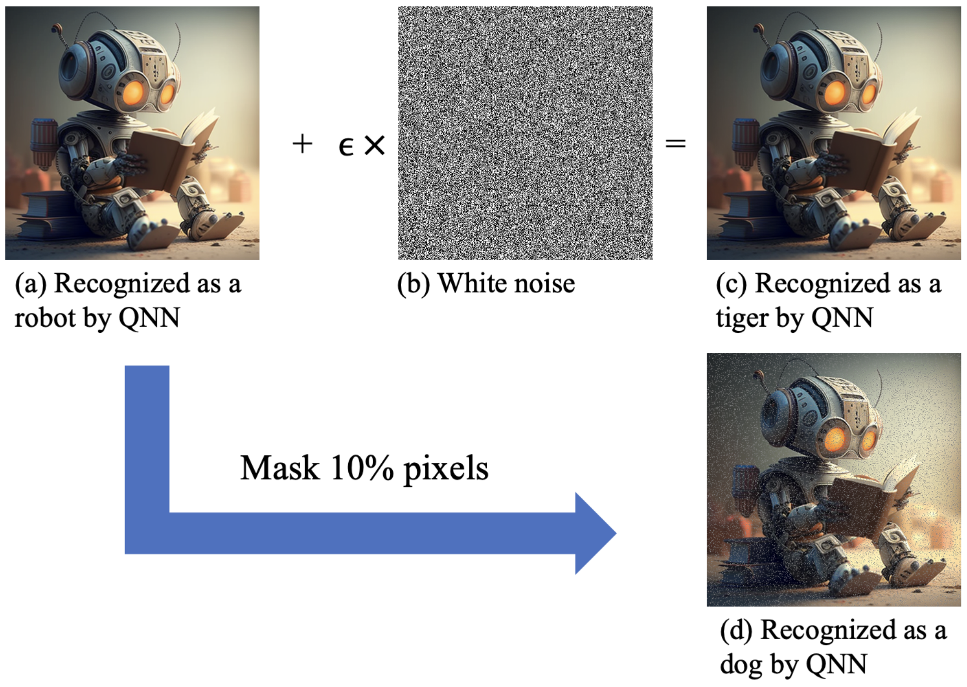

(a) Recognized as a robot by QNN
+ ϵ ×
(b) White noise
=
(c) Recognized as a tiger by QNN
Mask 10% pixels
(d) Recognized as a dog by QNN

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.

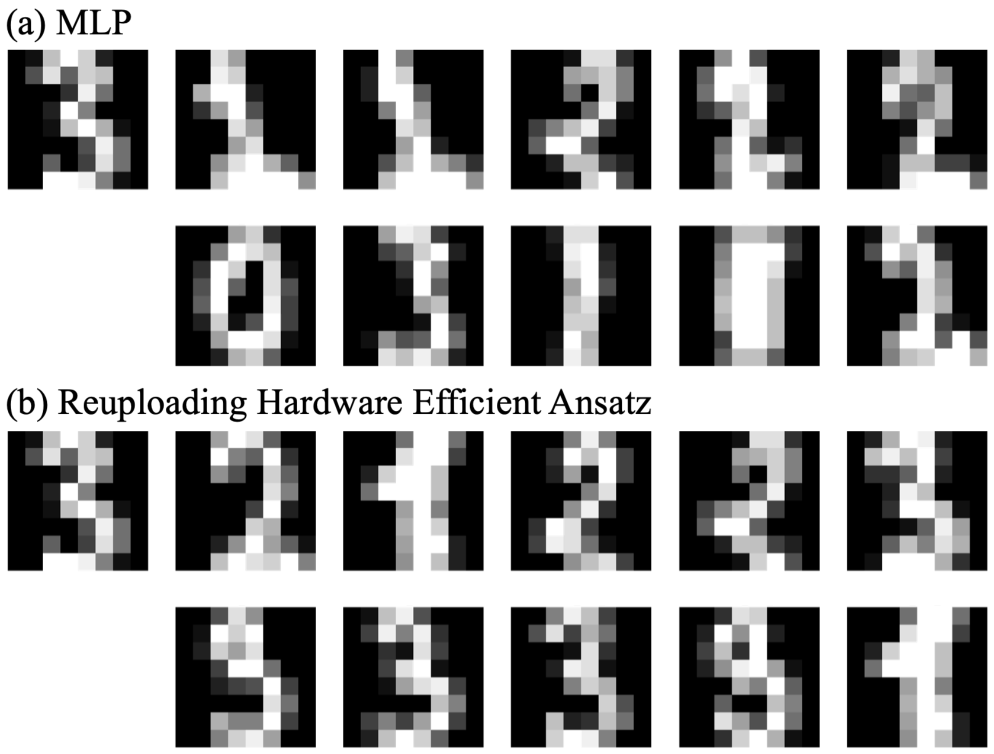


Figure 5. Explanations of a digit figure depicting the number 3. The top row shows the test data and the top-5 examples found by the influence function. The second row displays the top-5 examples found by the relative influence function.

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.

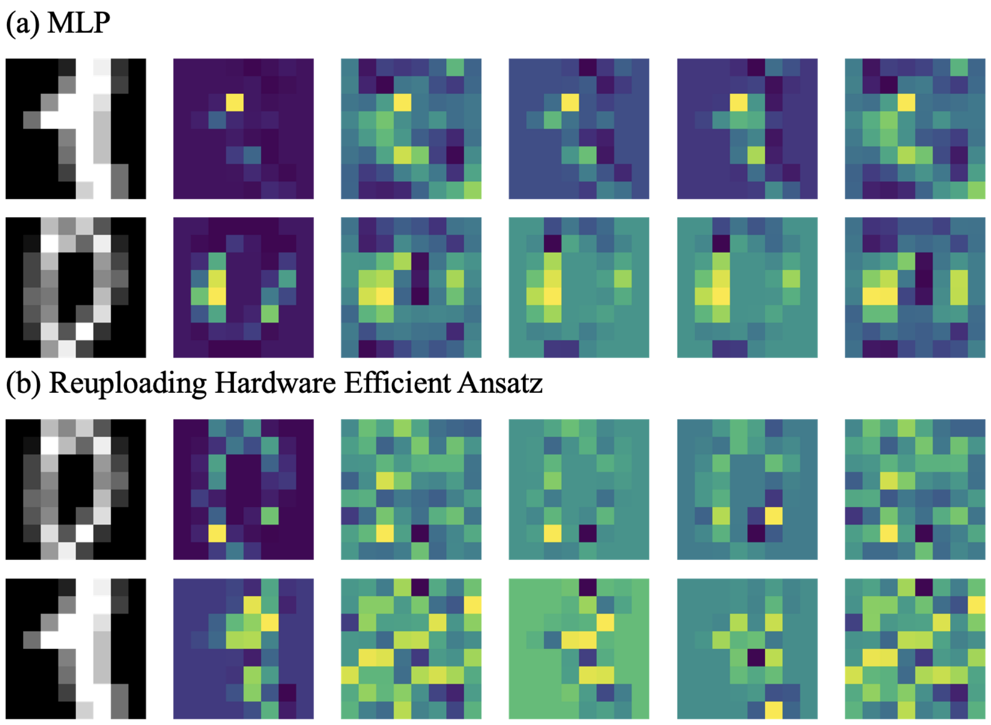
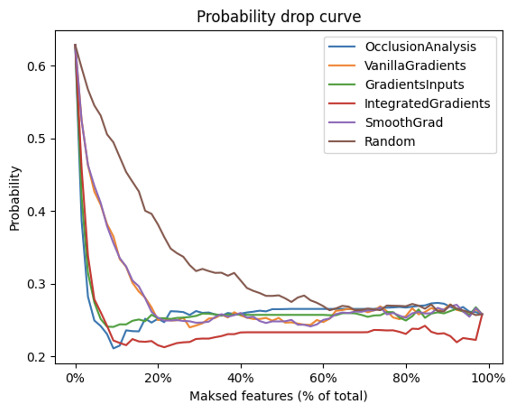
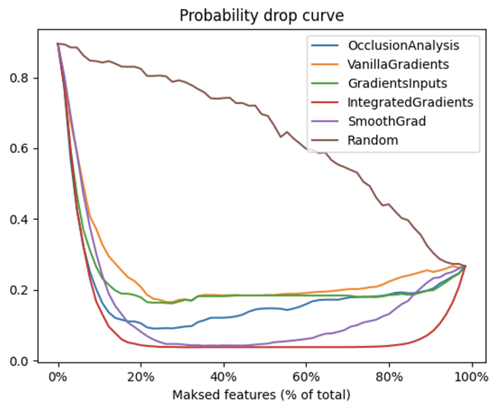


Figure 6. Explanations of two digit figures on Re-uploading Hardware Efficient Ansatz and MLP models. From left to right: original image, occlusion analysis, saliency map, gradient input multiplication, integrated gradients, and SmoothGrad heatmaps. In each heatmap, darker colors represent lower influence,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

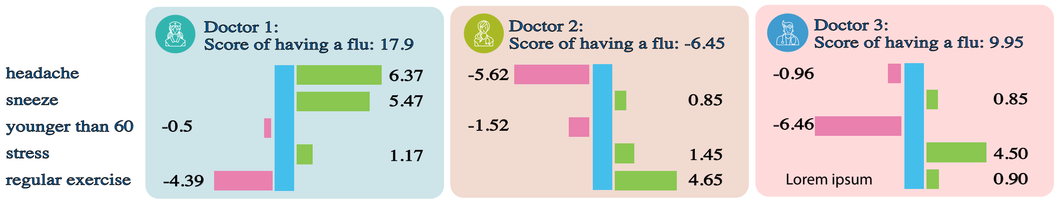


Figure A1. An example of the Clever Hans effect. This figure illustrates the decision-making process employed by three physicians in diagnosing a patient. The patient's symptoms are represented by a feature vector (True, True, False, True, False), corresponding to the presence or absence of a headache, 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.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.