Maastricht University

MSP Bachelor Thesis Research

Interpretability of Quantum and Classical Machine Learning Algorithms

Maastricht Science Programme

2022-2025

Author:

Miguel Miralda Porrero

Main Supervisor:

Jacco de Vries

Secondary Supervisor:

George William Scriven

Word count:

6976 words

19th May 2025

Abstract

With the aim to ultimately improve the interpretability of machine learning (ML) models, a newly developed explainable quantum technique, Q-LIME π was compared with the better established Local Interpretable Model-Agnostic Explanations (LIME), shedding light on how this new technique could be improved. This comparison was made through both quantum and classical machine learning models, aiming to also shed light on some of their differences.

Several key points of improvement were found for Q-LIME π, which proved to be too underdeveloped for a direct comparison of performance with LIME. These results are nevertheless relevant, constituting an exciting opportunity for future research.

Through the use of LIME, a global explainability technique was also developed by averaging the weights of all local explanations made by it, and creating a dictionary of global importance. Through this dictionary, the ML models were retrained many times by gradually filtering low-importance words by different thresholds.

The Bayesian information criterion (BIC) was afterwards employed to select the best model, with an optimal relationship between the number of features and the accuracy achieved through them. It was found that a model with seventeen times fewer features than the original was able to maintain a similar accuracy as the original model, with a drop in it of only 2.8%, pointing to the fact that many of the features used by the original neural network may be noise.

A similar procedure was applied for a quantum convolutional neural network (QCNN), however, the results were notably less favorable, indicating the need for further refinement in its training.

Contents

1 Introduction

Machine learning (ML), widely used in the modern world, encompasses a wide range of fields, from natural language processing in Large Language Models (LLMs) [1] to classification and prediction tasks in identifying exotic particles in high-energy physics [2]. However, the inner workings of these models are often opaque and difficult to understand, leaving open questions about their reliability. As such, different criteria have been developed to evaluate the explainability of a model. Some commonly used ones are: The comprehensibility of a model, the fidelity an explanation has with respect to the model explained, the accuracy of the explanations, the scalability of the method and the generality of it, which determines whether using the method requires any special treatment or whether it is broadly applicable [3]. Understanding how these models work is essential to guarantee their reliability and usefulness.

This explainability problem is also commonly found in quantum computing and quantum machine learning due to its complex nature. In the current world where ML is booming, quantum computing promises speed-ups on many of these algorithms with respect to their classical counterparts [4].

These quantum algorithms have shown promising applications in many fields; In early-stage drug development, they promise a speed-up from a 15-month process to just 7 weeks [5]. It has also been shown that the finance sector could potentially also benefit from these speed-ups [6]. However, the current state-of-the-art is not enough to realize the full potential of these applications.

The main problem lies in the current hardware limitations, as noise heavily affects quantum devices and the number of qubits available on them is very limited. Pure quantum methods are therefore not optimal, which is why a hybrid approach is more commonly used in Quantum Machine Learning (QML). Focusing the quantum algorithm on specific tasks, while the rest of the workload is processed by classical algorithms [7].

It is therefore essential not only to combine both the classical and quantum field but also to ensure that this combination remains reliable and comprehensible. For these reasons, classical algorithms were compared to their quantum counterparts, studying their explainability through the use of different techniques such as Local Interpretable Model-Agnostic Explanations (LIME) and the newly developed technique Q-LIME π. This study therefore aimed to gain new insights into these algorithms, exploring whether Q-LIME π could potentially be a good extension of LIME by further improving some of its aspects, and optimizing both the NN and QCNN trough

feature selection, aided by a computed global explainability technique.

2 Theoretical Background

2.1 Classical Machine Learning and Classification

Classical Machine Learning is commonly divided into supervised, unsupervised, and reinforcement learning [8]. This research mainly encompasses supervised learning, which corresponds to labeled data. The model trained on an input X is paired with the correct output Y. From there, the model learns to identify and predict the output for every given X.

Some of the classical ML models used throughout this work include a logistic regression model and a neural network (NN). An XGBoost model was also tested, but as minimal improvement was found when compared with the logistic classifier, it was dropped.

2.2 Quantum Machine Learning and Classification

Many different quantum algorithms have arisen through the years, aiming to provide speedups to their classical counterparts. In this research, due to time constraints, the main focus will be on studying and comparing a hybrid algorithm known as Quantum Convolutional Neural Networks (QCNNs)[9] with its classical counterpart (NN).

QCNNs can process both classical and quantum data by mapping the information into a quantum circuit. These circuits can be trained by optimizing a set of parameters θ that parameterize the circuit [10]. Its optimization is computed through classical methods, while the loss function can be calculated through different ways, with the parameter shift rule remaining as one of the most popular ones [11] [12].

2.3 Quantum Computing Background

In quantum computing, a pure state ψ can be represented through the formula:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1}$$

With $|0\rangle$ and $|1\rangle$ constituting the basis vectors of the Hilbert space:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2}$$

In the above-mentioned formula, alpha and beta are complex numbers known as the probability amplitudes of the wavefunction. These amplitudes, when squared, represent the probability of a particle being in a certain state.

$$P(0) = |\alpha|^2, \quad P(1) = |\beta|^2 \tag{3}$$

With the sum of probabilities of all basis states being 1, they satisfy the normalization condition:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{4}$$

In regard to defining a Hilbert space $\mathcal{H}$, it is a complete inner-product space that provides the mathematical framework of quantum mechanics. It has an orthonormal basis, meaning the set of vectors within it are orthonormal with each other and have unit length.

A Hilbert space with a single qubit can be defined as $\mathcal{H} = \mathbb{C}^2$, with $\mathbb{C}^2$ being the complex vector space described by the basis states $|0\rangle$ and $|1\rangle$, which makes it 2-dimensional. Regarding a Hilbert space with n number of qubits: For $\mathcal{H}^{\otimes n} = \underbrace{\mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2}_{n \text{ times}}$ with $\dim\left(\mathcal{H}^{\otimes n}\right)$ the computational complexity increases with 2^n. This is sometimes also referred to as the n-qubit register.

In practical terms, this makes increasing the number of qubits on a classical computer quantum simulator very restricted, as the resource consumption grows exponentially. As such, it's generally not recommended to simulate more than 20 qubits.

2.3.1 Angle Embedding

Angle embedding is a technique used in quantum ML to encode a classical feature vector $x = (x_1, \ldots, x_N)$ into a quantum state, by applying single qubit Pauli rotations to every feature in x, as shown by the following equation:

$$\text{Angle Embedding } (x) = \prod_{i=1}^{N} R_i(x_i) \quad \text{with} \quad R_i(\theta) \in \{\text{RX}_i(\theta), \text{RY}_i(\theta), \text{RZ}_i(\theta)\} \tag{5}$$

2.4 Explainable Artificial Intelligence (XAI)

Developed with the aim to grant a better understanding of the inner workings of ML models, explainability techniques are mainly classified into two types: intrinsic and post-hoc methods.

2.4.1 Intrinsic methods

Intrinsic methods aim to explain the model as part of the model itself. The decision-making process is transparent and understandable, making the model's reasoning inherently explainable.

An example is a binary logistic regression model used for classification. In the following equation, every input feature is represented as x_i, with a corresponding feature importance β_i, and β_0 as the bias term. A logistic function σ then maps this prediction to 0 or 1:

$$P(y = 1 \mid \mathbf{x}) \;=\; \sigma\left(\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n\right), \quad \text{where } \sigma(z) = \frac{1}{1 + e^{-z}} \tag{6}$$

2.4.2 Post-hoc methods

Post-hoc methods aim to provide explanations for a ML model after it has been trained. These post-hoc methods are used often in modern ML architectures to explain black-box models like neural networks (NN). These models are often too complex for a simple explanation, and in these cases, model distillation is often used. Model distillation is a type of post-hoc explanation that translates the reasoning of a complex model into a simpler and more interpretable one [13]. The local post-hoc methods introduced in this paper fall into this category. However, it has also been shown that post-hoc explanations remain vulnerable to adversarial attacks and, as such, aren't foolproof [14].

A possible use case for these post-hoc techniques is shown in the following figure, where a ML model predicts the diagnosis of a patient based on its symptoms, explaining its reasoning to a doctor, so that it can make a final, more informed decision:

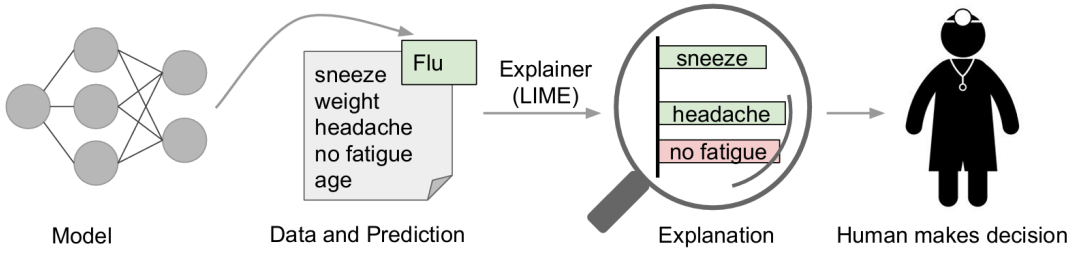

Figure 1: *Illustration of a possible application of XAI techniques, retrieved from [15].*

2.5 Local Explainability Techniques

In the context of XAI, a local explanation refers to an instance of the data. Thus, the explanation quantifies each feature's contribution to that specific prediction but doesn't explain the feature's relevance for other predictions.

2.5.1 Local Interpretable Model-Agnostic Explanations (LIME)

LIME is a post-hoc, local, model distillation technique that treats the model as a black-box, and therefore makes it model-agnostic [15]. It is classified as a model distillation technique, as it distills a larger model into a simpler, more interpretable one.

It has been shown that it provides reliable explanations for the importance of features in local predictions when used correctly [16].

LIME aims to provide explanations for local instances of a model f that has been previously trained by taking a specific instance $x \in \mathbb{R}^d$ to explain and generating multiple perturbed instances z that will be compared to the original instance to determine the local impact of each feature in the model. The original input x is evaluated by the original model f while the perturbed instance z is evaluated by a different model $g \in G$, where G is the set of candidate models for the local linear explanations.

$$\mathcal{L}(f, g, \pi_x) = \sum_{z,z' \in \mathcal{Z}} \pi_x(z) \, (f(x) - g(z))^2 \tag{7}$$

LIME also employs a kernel $\pi_x(z)$ that weights perturbations based on their Euclidean distance from the original instance x.

$$\pi_x(z) = \exp\left(-\frac{d(x,z)^2}{\sigma^2}\right) \tag{8}$$

This kernel is very useful when studying continuous data, as it is used to evaluate perturbed points z closer to the original instance x with a higher weight than otherwise.

However, for a discrete vector space, if the perturbations are made in the unit Hamming sphere (as is the case of the currently known use case of the algorithm presented in 2.5.2) the kernel becomes ineffective.

The whole formula for LIME is the following:

$$\xi(x) = \arg\min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{9}$$

In it, the term $\Omega(g)$ reduces the complexity of the explanation and prevents overfitting by using L_p regularization, therefore limiting the number of features used in the explanation.

Aiming to fix some of LIME's limitations, different techniques have been developed throughout the last years: For tasks with highly non-linear decision boundaries, like natural language processing (NLP), Anchor Explanations employs if-then rules [17]. To improve limited interpretability, LIMASE, a technique yet to be peer-reviewed, merges LIME and Shapley values [18].

2.5.2 Q-LIME π

Q-LIME π is another post-hoc, local, model distillation XAI method that closely mimics LIME's inner workings. It is a recently developed quantum algorithm that only employs binary vectorized data [19]. It has been tested and implemented by the author, but as it has yet to be peer-reviewed, its reliability is still in doubt. However, it has been shown that for discrete feature spaces, it is currently significantly faster than its classical counterpart, with a time complexity of O(1) when run on quantum hardware, while keeping a certain degree of similarity to the results obtained by LIME (as shown in 4.2), which has a best time complexity of O(n). It is important to note that the original Q-LIME π significantly differs from LIME in many areas, and as such, the discrepancies seen are to be expected (for e.g. no regularization is being done).

Through the use of a vectorizer, the presence or absence of features is encoded into a binary vector. Afterwards, in a similar manner as in LIME, perturbations are applied. The features are first encoded into a quantum state through an RY gate. Afterwards, the active feature meant to be perturbed $x_k = 1$ is mapped from 1 to 0 through a Pauli-X gate.

These perturbed features, unlike in LIME, simply select the most repeated features through the sample. This, however, is a suboptimal approach. If the same feature selection technique used in LIME is applied here, its speed-up promises are likely to be compromised, with a time complexity of O(n).

After perturbing the selected features, like in LIME, it measures the impact this changed feature has on the local prediction and calculates its importance.

The following formula describes how this importance is calculated:

$$\Delta f_k = f(\mathbf{x}) - f(\mathbf{x}_{\text{perturbed},k}) \tag{10}$$

Where $\mathbf{x}_{\text{perturbed},k} = [x_1, \ldots, 1 - x_k, \ldots, x_n]$ is the feature vector obtained by flipping x_k.

2.5.3 Differences between Q-LIME π and LIME

There are many differences between these two algorithms. The most obvious one being that Q-LIME π aims to explain data encoded into a quantum state, with the number of qubits used in the circuit being equivalent to the number of features explained. This limits the number of features to explain, due to the high computational cost of simulating more qubits on a classical computer [20]. Meanwhile, LIME only processes classical data.

The way they select their features also differs. While LIME creates a high number of perturbations and explains the features with more impact among those, Q-LIME π lacks anything related to the selection of features. It is instead done by CountVectorizer, a method that extracts features based on their frequency in the text sample.

Due to all of this, both techniques currently output very different results, and directly comparing them can be misleading. However, if Q-LIME π had been developed as much as its classical counterpart, similar outputs could be expected, as their inner workings would remain similar.

Some important differences will nonetheless remain; although LIME can easily be implemented in either a discrete or a continuous vector space, Q-LIME π has so far only been implemented in a discrete space. While its use in a continuous vector space is possible, it would require an appropriate quantum encoding, making it a less general technique.

2.6 Global Explainability Techniques

A global explanation aims to quantify the role of the features in the model's overall reasoning. It is considered global because it reflects the average importance of features across all instances, rather than focusing on a single prediction.

Common use cases for these global techniques include the study of biases present in LLMs, caused by both the data they are trained on and their model architecture. Racial biases have been found in many models, like GPT-3 [21], and more recently Grok, the LLM of X, with its

false claims of white genocide in South Africa. [22]

Figure 2: *Illustration of Grok, the LLM employed by X, which recently showed racial bias through false claims in X*

2.6.1 Shapley Values as a Global Explainer

Shapley Values is a technique originated in game theory, named after Lloid Shapley and for which he was awarded the Nobel Memorial Prize in Economic Sciences in 2012. It is often used as a global model agnostic XAI technique, as it is capable of assigning importance values to the features present in a dataset based on their contributions, giving first a local explanation, which can be used to compute a global one. Let:

- $f(x)$ be the prediction function of a model.
- $x = (x_1, x_2, \ldots, x_j)$ be an input instance.
- $N = \{1, 2, \ldots, n\}$ be the set of all feature indices.
- $S \subseteq N \setminus \{i\}$ be any subset of features excluding feature i.

The following formula shows how the importance of a feature i in an instance x is computed:

$$\phi_i(f, x) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(n - |S| - 1)!}{n!} \left[f_{S \cup \{i\}}(x) - f_S(x) \right] \tag{11}$$

By adding these over all instances M, a global explanation is computed:

$$\bar{\Phi}_i \;=\; \frac{1}{M} \sum_{j=1}^{M} \left| \phi_i(f,\, x^{(j)}) \right| \tag{12}$$

It is generally accepted as a good explainer that can work in a global manner, with the drawback that it's a combinatorial problem. Due to this, its computational complexity grows exponentially with the number of features computed, by $O(2^n)$. This complicates its usage in some ML models like LLMs, as they often employ a huge number of features in order to obtain their predictions [23].

2.6.2 Global LIME

Lime was previously proposed as the foundation for a potential global XAI technique [24]. This method would work by simply averaging the absolute values of the importance scores across all data instances, so that a set of features remains with a global value of importance assigned to each of them. However, this approach may suffer from high variance due to multiple local approximations.

A more sophisticated approach was taken through the development of Globally Assessing Local Explanations (GALE), where several modifications to this method were implemented, allegedly improving the global explanations [25].

2.7 Bayesian Information Criterion (BIC)

The Bayesian Information Criterion is a model selection criterion that aims to optimize the number of parameters used with respect to the accuracy of a model. [26]

It is defined by the formula:

$$\mathrm{BIC} = -2\ \ln\left(\widehat{L}\right) + k\ \ln(n) \tag{13}$$

Where $\widehat{L}$ is the maximized likelihood function, k is the number of free parameters in the model, and n is the validation set size.

A likelihood function gives an estimate of how well a model explains the observed data. For binary outcomes with observed accuracy a:

$$\widehat{a} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{14}$$

A maximized Bernoulli likelihood function is employed:

$$\widehat{L} = f(n \mid a) = a^n(1-a)^{1-n} \tag{15}$$

For the model-agnostic approach used in this research, k is defined as the sum of the number of features m with the intercept:

$$k = m + 1 \tag{16}$$

After applying the logarithm, the original formula becomes:

$$BIC = -2n(\ln a + (1-a)\ln(1-a)) + (m+1)\ln(n) \tag{17}$$

When searching for the most optimal settings by varying the number of free parameters k, the lower the BIC value, the more efficient the parameters-accuracy relationship is. As such, BIC is considered optimal at the minimum.

3 Methodology

Different datasets were subjected to appropriate ML models for classification tasks while evaluating their explainability and performance. These ML models included quantum algorithms such as QCNNs. The data used for this project consisted of a test dataset, for which the iris dataset was employed, and a dataset consisting of tabular text data, used for sentiment classification.

The tabular text data originates from the IMDb dataset, which is an open-source dataset of film reviews consisting of text data. This project originally aimed to also classify particle physics and time series data, with the aim of applying these new techniques to different types of problems. However, a more constrained and specialized topic were prioritized, and these were dropped.

3.1 Iris Dataset

With the goal of building familiarity with the algorithms used in this project, a well-known dataset - Iris, was employed at the beginning of the project for conducting a preliminary test of the explained ML and XAI techniques.

A neural network (NN) and a QCNN were used as models, to test both XAI techniques and search for differences in their predictions.

In the process of doing so, an issue with Q-Lime π appeared, due to how it employs binary vectorized data while the Iris consists of continuous data.

To fix it, the data was converted into a binary format by taking the mean value of every feature and setting the values lower to 0, while the rest were set to 1. While this method discards an important part of the information that may have been useful to a classifier, it is sufficient to conduct a preliminary test.

3.2 Sentiment Classifier

Film reviews were retrieved and classified based on positive or negative sentiment. These predictions were then explained through explainability techniques, aiming to get insights into both the XAI techniques and the underlying ML models.

3.2.1 Data Pre-processing

Text classification is a NLP task, however, ML models operate using numerical data rather than raw text, and as such, the input must first be converted into a suitable format.

The text sample is initially split into a set of tokens, in a process known as tokenization. Afterwards, each token is mapped into a unique vector representation that serves as an identifier. This process is known as vectorization, and is what ultimately enables the ML model to process the data. Therefore, every word is ultimately encoded into a vector, which acts as a feature of the model.

Due to this, in order to reduce the huge amount of words and therefore features that compose this vector space, additional pre-processing processes were carried out, including an initial filtering of the data aimed towards common English stop-words, as their importance when training the model is negligible.

Another technique used in order to reduce the dimensionality of this vector space, making the ML models more efficient, was lemmatization, a process in which different versions of a word are converted into its 'lemma'. An example being how all of these different versions of the word "run": "running", "ran" and "runs" are mapped to "run", sharing one feature index, and simplifying the explanation.

3.2.2 Training of the NN model

The NN was trained using the following number of layers:

1. First a dense layer of 128 units using ReLU as an activation function, accepting inputs in the shape of the number of samples as rows, and the number of features as columns.
2. A dropout of 30% to reduce overfitting.
3. Another dense layer of 64 units using ReLU as an activation function.
4. A last dropout of 20%.
5. A single sigmoid unit for binary classification into a positive or a negative sentiment.

Lastly, to compile the model, the NADAM optimizer was used, with a learning rate of just 0.0005, as after several trials it was found to be an optimal value for the training. The loss was measured through binary cross-entropy, and as a metric of improvement, accuracy was used.

Ultimately, the optimizer optimizes the loss function through several epochs, aiming to improve the accuracy.

Even after implementing the dropout layers, overfitting issues persisted. These were noticed through the observation of the validation loss parameter, which gives direct insight into how the model performs on new, unseen data. To avoid overfitting, an early stopping mechanism was used, for which if 2 consecutive epochs of training failed to improve validation loss, the training was stopped to avoid overfitting, and the best weights so far were used.

3.2.3 Training of the QCNN model

Raw binary labels $0/1$ were first converted to -1 and $+1$, aligning the target range with the QCNN's Pauli Z expectation values, which lie in $[-1, +1]$. The architecture of the model worked as follows: First, using angle embedding, the features were divided into blocks of up to n_{qubits} features, which were then loaded into qubit rotation angles (θ). Afterwards, these qubits were entangled, and CNOT and RY rotations gradually shrank the number of qubits in a manner similar to a convolutional neural network. The final measurement yielded a real-valued output in $[-1, +1]$, which was then mapped back to binary values through the following formula:

$$\hat{y} = \mathbb{I}\left(\frac{\langle Z \rangle + 1}{2} > 0.5\right) \tag{18}$$

Where $\mathbb{I}$ is the indicator function:

$$\mathbb{I} = \begin{cases} 1, & \text{if true} \\ 0, & \text{if false} \end{cases} \tag{19}$$

The model was trained using 2 parameters, w_{ent} of shape $(n_{\text{layers}}, n_{\text{qubits}}, 3)$ and w_{conv} of shape $(n_{\text{qubits}} - 1, 2)$. Both of which were initialized with a small amount of noise.

PennyLane's GradientDescentOptimizer was used as the optimizer, with a fixed step size η. One gradient-descent step on $(w_{\text{conv}}, w_{\text{ent}})$ was performed per batch, and the loss was monitored through the mean squared error.

The training itself took a significant amount of time due to the large feature set (100 features) and the dataset size. Consequently, every individual prediction also required more time than a normal classical counterpart would.

3.2.4 Bayesian Information Criterion (BIC) Optimization through LIME

The Bayesian information criterion is a method to measure the efficiency of a ML model with respect to the number of parameters used. It was used to improve the efficiency of a NN by combining the Bayesian inference criterion with LIME.

First, a global dictionary of importance was developed through averaging all the different local explanations for a given word in a set of words. Through this method, earlier introduced in the theory, a global measure of importance was developed. The main concern shown by literature was the magnitude of the spread of the data, due to multiple local approximations. This spread was measured through the mean of the standard deviation of all words inside of a given weight range.

Afterwards, a filtered model was trained without the features that, on average, were lower than an arbitrary threshold of importance. This was done through different threshold values, aiming to find the most efficient model, which would be determined by BIC.

This methodology was initially implemented for both LIME and Q-LIME π. However, the latter was dropped due to its limited accuracy and the low number of features/qubits that can be simulated at a time. Additionally, even though both techniques work in a similar theoretical framework, the lack of development of the latter makes a direct comparison between both misleading, as explained in section 2.5.3.

If the features explained indeed have the low importance LIME claims, the efficiency of the retrained models will improve by lowering the BIC value. This technique also enables the direct measurement of the impact any modifications done on an XAI technique may cause. If these modifications lead to retrained models with a lower BIC value than with the unmodified XAI over several random sets of samples, they are likely to be beneficial for its explanations.

4 Results and Discussion

4.1 Iris Classification

The explanations for a local prediction made by the 2 local XAI techniques earlier introduced are the following:

Through a simple neural network, an accuracy of 92.92% was achieved on the validation dataset:

Method	**Sepal Length (cm)**	**Sepal Width (cm)**	**Petal Length (cm)**	**Petal Width (cm)**
Q-LIME π	0.10366189	0.01569134	0.98544151	−0.0016377
LIME	0.14283611	−0.11564152	−0.14165877	−0.26782914

Table 1: Feature contributions for sample 0 of the Iris dataset, comparing Q-LIME π with 100 shots and LIME explanations using a simple sequential Neural Network.

Through the QCNN, an accuracy of 94.69% was achieved on the classification dataset, with the following explanations:

Method	**Sepal Length (cm)**	**Sepal Width (cm)**	**Petal Length (cm)**	**Petal Width (cm)**
Q-LIME π	−0.33864572	−0.35680134	−0.75520078	−0.23955829
LIME	−0.01097924	0.05558249	−0.66151549	−0.05394883

Table 2: Feature contributions for sample 0 of the Iris dataset, comparing Q-LIME π with 100 shots and LIME explanations using a Quantum Convolutional Neural Network.

These results show how no 2 explanations have a high resemblance with one another. This is partially due to the QCNN having a different architecture from the NN, and as such, their inner workings are bound to differ. However, these results also showcase how Q-LIME π fails to match LIME's output, which may point to how underdeveloped it really is.

On another note, these results also give insights into the architecture of the ML models used. Even when explained through LIME, with both models having been trained to similar accuracies, the same sample has significantly different weights associated with its features when comparing the classical ML model with the quantum one.

4.2 IMDb Sentiment Classifier

Due to the huge amount of features present in each film review, and the limitations present in Q-LIME π, the results for all the features can't be easily shown in tables like in the earlier

section. However, with the aim of illustrating both techniques, the following heatmap shows the predictions obtained from both Q-LIME π and LIME for a sample, only explaining the 15 most important features:

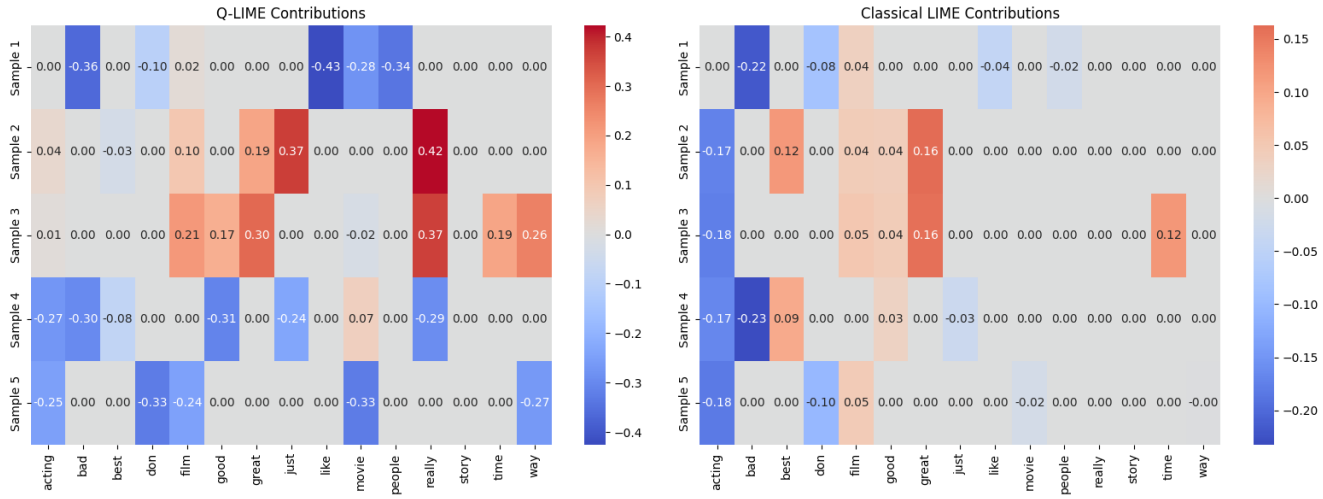

Figure 3: *Heat map of Q-LIME π vs. LIME of a logistic regression sentiment classifier with an accuracy of 68%: Feature's Overlap: 76%, time to compute: LIME-3.056 sec. vs. Q-LIME 0.1154 sec., number of samples the model was trained on: 500, LIME number of samples: 300, stopwords: True, features explained: 15.*

It can be seen how both methods choose relatively similar features to explain. However, as discussed in section 2.5.3 important differences in the process make this comparison purely illustrative.

How much both techniques actually differ also depends on how the perturbed words are chosen. As LIME is widely accepted as a good local explainer, it can be used as a benchmark for this comparison. As such, the higher the similarity between the selected features to perturb, the more accurate Q-LIME π method is likely to be.

Q-LIME π can perturb locally active and non-active features at the same time, or just locally active ones. However, it is unclear which approach would be more beneficial towards the explanations. The following table shows the results of a test experiment, aimed towards answering this question:

Q-LIME π flip of features	**Overlap in feature selection with LIME**
Only active features	64 %
Both active and non-active features	42 %

Table 3: Table describing the average overlap in feature selection when flipping different features, measured through 10 different samples

As seen in the figure, perturbing solely actively local features had a higher similarity with the

results obtained from LIME than perturbing both active and non-active features in the locality. This can be easily understood, as inactive features are features that aren't present in the sample, and perturbing them would likely make the prediction non-local and, as such, unreliable.

These results also give insights into how important the features extracted by CountVectorizer are, when compared with LIME. This being a method commonly used for feature extraction from text.

4.2.1 Global Dictionary of Importance and Model Optimization with a Neural Network

Training a neural network for predicting the sentiment of a film review with high accuracy and a not too complex architecture proved to be harder than expected. This is likely due to the complexity found in NLP. The architecture of a neural network treats every feature as independent, and as such, grasping nuances like 'not bad' can prove difficult. Regardless of this, a NN was successfully trained to predict the sentiment of the dataset samples with an accuracy of 82%. Afterwards, a global dictionary, mapping a set of features with their importance was computed through the explained methodology. In this dictionary, the distribution of words with respect to their importance was expected to follow an exponential distribution, as highly important words tend to occur much less frequently. The word-weight distribution was therefore plotted:

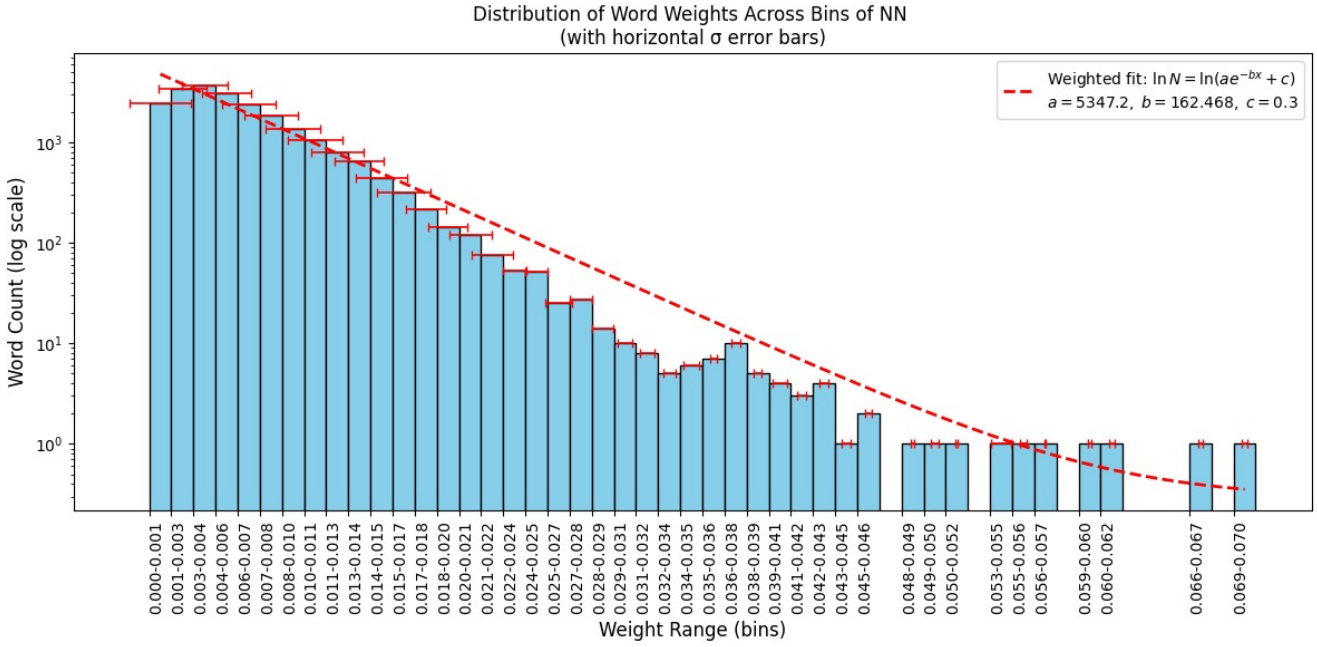

Figure 4: *Weight distribution of the global dictionary, computed with a neural network and LIME. The standard deviation was measured through a comparison of 5 global dictionaries.*

The mean standard deviation was selected as a measure of the spread of the data as it keeps the same units as the data, making these results more interpretable. It was computed through the creation of 5 different sets of explanations. One of these explanations was selected to work as the 'original dictionary', which was plotted on the blue bars. The words belonging to it were separated into bins, and the standard deviation for each weight range bin was calculated by comparing the weights of every word belonging to a bin, with the explanations found in the other dictionaries for this same word. These standard deviations were then averaged to obtain an overall value, which is the mean of the standard deviation for that weight range. The same process was repeated for every bin.

Figure 4 shows how the word-weight distribution of the dictionary follows an exponential distribution, which was expected. It must be noted that due to the y-axis being plotted on a logarithmic scale, the typical, curved shape of exponential growth changes from a curve to a line.

After this global dictionary of importance was implemented through averaging LIME output, the features were filtered by different thresholds of importance, aiming to train an optimal, efficient model. These thresholds are shown in the following table:

These thresholds of importance were gradually changed in an exponential manner, as this was the distribution of weight shown by figure 4.

The model was retrained for every threshold of filtered data. After assessing the resulting accuracy of each retrained model, the data was plotted against the number of features used to train it. The results are shown in the following graph:

Threshold	Number of Features
0.065	2
0.06	3
0.05	8
0.04	23
0.035	46
0.03	68
0.025	149
0.02	407
0.015	1259
0.01	3900
0.0075	6664
0.005	11093
0.004	13455
0.003	16160
0.002	18702
No threshold	22226

Table 4: Number of features selected at various thresholds for the NN

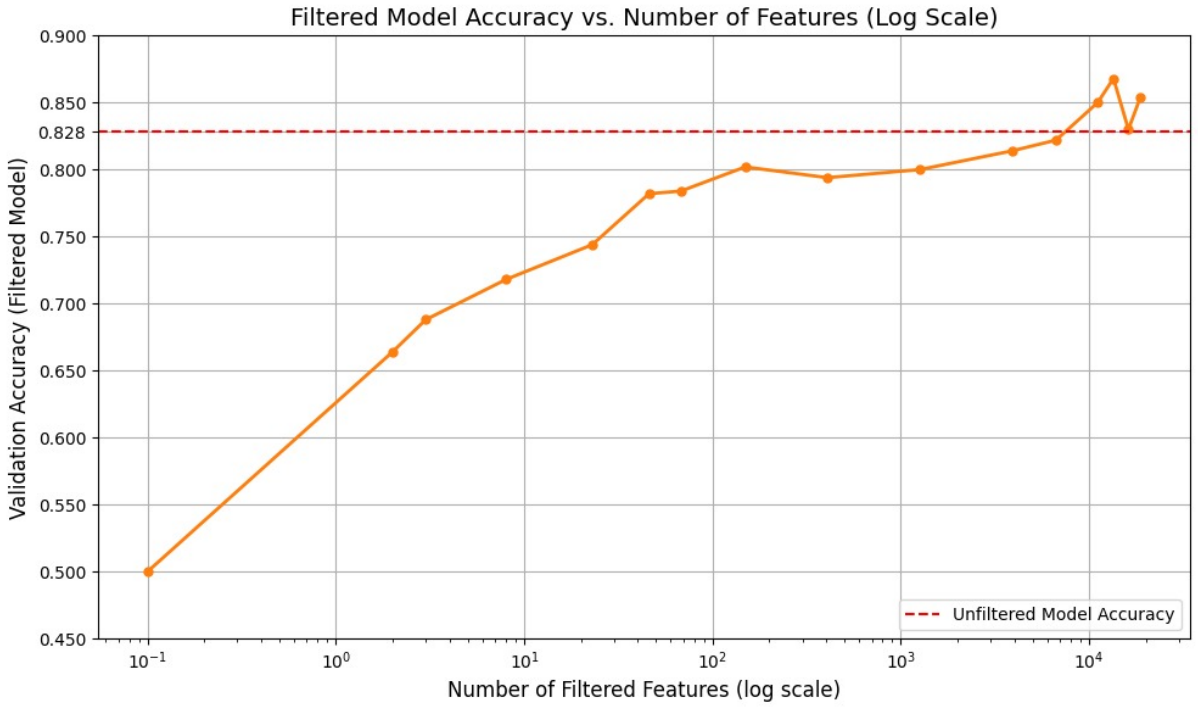

Figure 5: *Accuracy of the retrained models as a function of the amount of features used during training using a NN*

This graph shows how the accuracy grows exponentially when the number of features used has a relatively high importance. However, when the accuracy of the unfiltered model is reached, this growth plateaus, which is expected. This is due to the fact that from that point, the features are mostly treated as noise. To achieve a higher average accuracy, a more complex NN would have to be constructed to capture more complex patterns in the data.

It is remarkable, however, that after aggressively filtering the number of features and retraining the new model with only the 2 most important features, 'worst' with an importance of 7%, and 'bad' with an importance of 6.63%, the NN was able to maintain an accuracy of 66.4% on its predictions.

By calculating and plotting the Bayesian Inference Criterion for all of these retrained models, the optimal model can be retrieved from the minima of the plot.

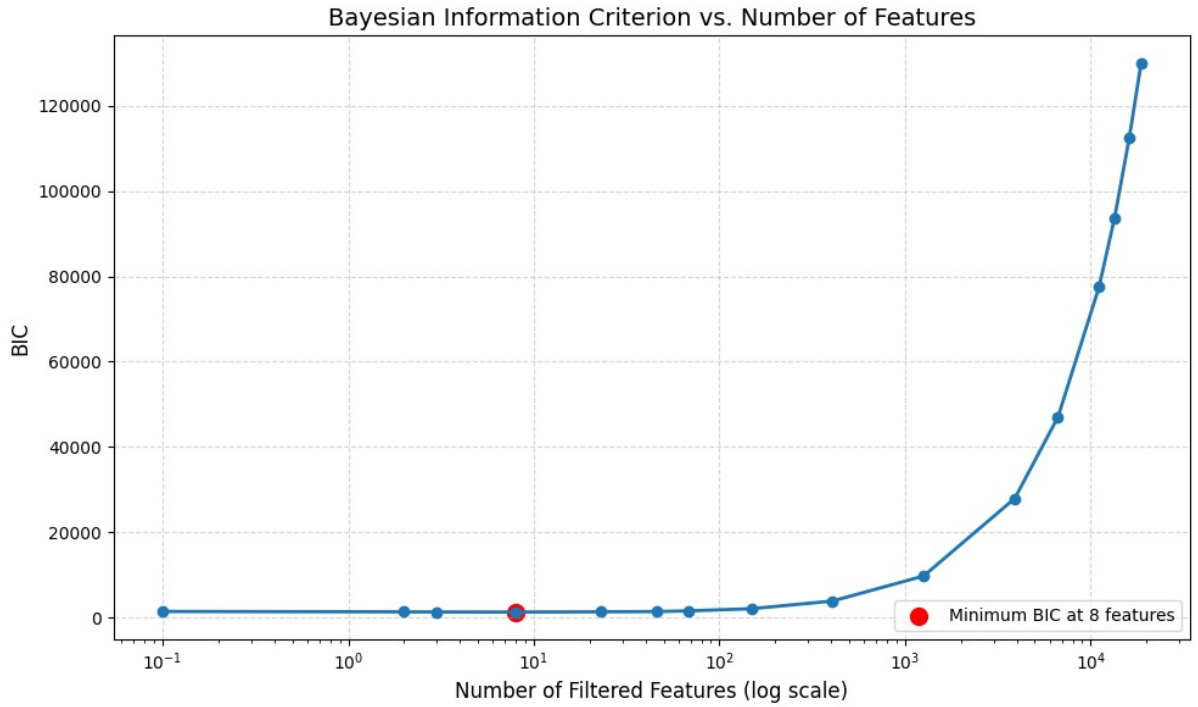

Figure 6: *Graph displaying the Bayesian information criterion vs number of filtered features in a logarithmic scale*

The selected, optimal model had an accuracy of 71.8%, while it was trained on only the 8 features it deemed most important. Namely: 'great', 'love', 'waste', 'awful', 'dull', 'worst', 'bad' and 'poor'. Manually going over them shows how these words indeed tend to carry a big importance when trying to determine the sentiment attached to a sentence.

This is a remarkable feat, as identifying the most important features not only gives a better understanding of how the model makes its predictions and deepens semantic understanding, it also enables the possibility of retraining the model using only these key words. For a NN, this

may seem like a trivial application. However, other models like QCNN are often very limited by the number of features and, as such, can greatly benefit from such a feature selection.

4.2.2 Global Dictionary of Importance with a Quantum Convolutional Neural Network

Due to time restrictions, computing a similar model optimization method with a QCNN with an unchanged amount of data was proven unrealistic, as just the training of one QCNN model takes several hours, and creating a global dictionary with the same amount of samples used in earlier graphs is estimated to take 2 days of computation. This process would have to be repeated at least 5 times to mimic the weight distribution plot shown before, as multiple dictionaries are needed to calculate the spread of the data through the mean standard deviation of each bundle.

Instead, a smaller subset of data consisting of only 100 samples was selected for training and feature selection purposes. With the aim of gaining initial insights into whether the QCNN would show a similar global exponential word-weight distribution, as the one observed on the NN, and whether this distribution would show similar mean standard deviations in similar intervals of word weights.

Due to time constraints, the QCNN model could not be retrained on the filtered features found through the NN global dictionary. Instead, a previously trained QCNN with 100 features selected by CountVectorizer was used, with a validation accuracy of 63.5%. Afterwards, the distribution of word-weights was calculated and plotted. If a vectorizer hasn't been fitted on a word, making explanations of that word with LIME will most likely result in unreliable predictions. Due to this, with the vectorizer only fitted on 100 features, the ML model was limited in the number of words it could explain. Therefore, the following plot has a significantly smaller number of words with respect to the one earlier done with the NN model.

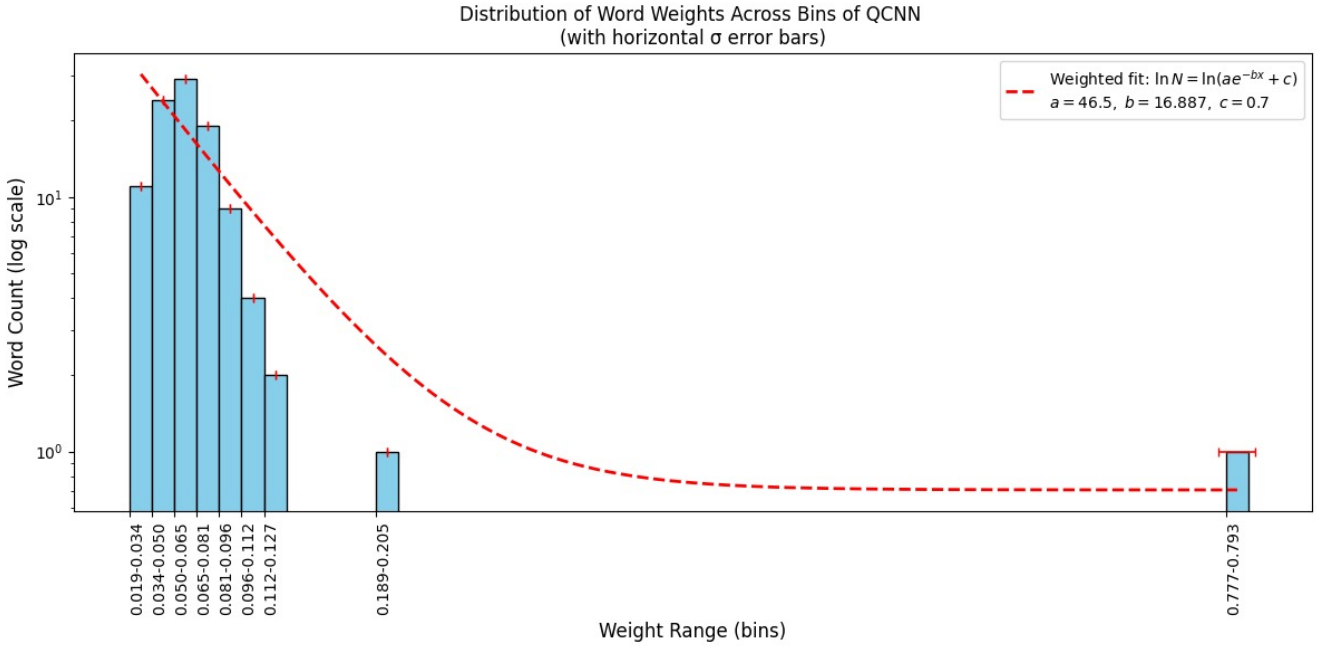

Figure 7: *Weight distribution of the global dictionary, computed with a quantum convolutional neural network and LIME. The standard deviation was measured through a comparison of 5 global dictionaries.*

As expected, the 100 words plotted follow an exponentially decreasing curve, shown by the fitted line. However, due to the small number of words, and the low accuracy of the model, these explained word bins may be subject to lower accuracy than the results obtained in earlier sections. In order to increase the accuracy of the model, it would certainly be beneficial to only use the 100 most important words earlier selected through the NN global explainer, in the QCNN training, as they are bound to be significantly more important than the words selected by CountVectorizer.

To minimize the variance shown by the plot, it would likely suffice to simply use a similar amount of samples for training as used in the NN, which were 25 times more numerous (2500 samples).

As done with the NN, the model was retrained through filtered data, using only the words with a higher importance than the threshold applied. The specific thresholds used are shown in the following table:

This table shows how the model was retrained through a very limited number of thresholds when comparing it to the amount used in the NN. This was due to the poor accuracy results found on all of the retrained models, which are shown on the following graph:

Threshold	Number of Features
0.13	3
0.1	7
0.03	91
0.01	100
No threshold	100

Table 5: Number of features selected at various thresholds for the QCNN

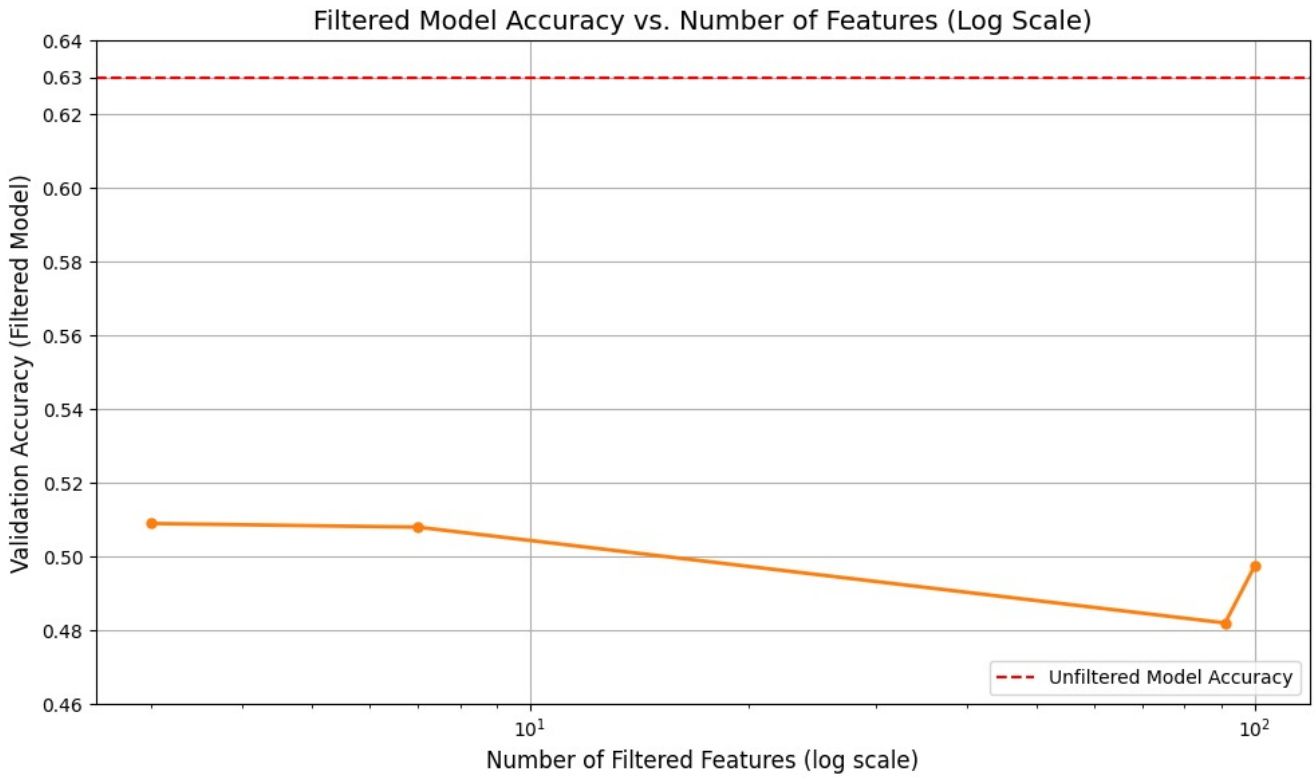

Figure 8: *Accuracy of the retrained models as a function of the amount of features used during training using a QCNN*

A low accuracy is observed for almost all datapoints, even for the unfiltered 0.01 retraining. This is likely due to several reasons. First of all, the original unfiltered model gave a weight of 79.3% to the word 'bad', while the rest of the words were assigned a very low weight in comparison. Therefore, its reasoning was very limited, directly correlating the word 'bad' with a negative sentiment, while the samples that didn't contain this word likely suffered a significant drop in the accuracy of predictions.

The limited number of relevant features available and the low amount of training data used are also reasons for which a higher accuracy was not observed in the retrained models. It must also be noted that the reliability these global dictionaries have when giving an importance to a

word directly depends on the accuracy of the underlying ML models. As such, the unfiltered QCNN model, with an accuracy of only 63.5%, was another limiting factor in this retraining procedure, due to its poor accuracy.

5 Conclusion and Critical Reflection

Much was accomplished through this bachelor thesis. For one, deficiencies present in Q-LIME π that don't allow a direct comparison with LIME were identified, paving the way for its improvement. On this front, the original paper left open the possibility of creating perturbations for both active and non-active features in the locality of the sample. While this choice may depend on the nature of the data used, it has been shown that, at least for our use case, perturbing both makes Q-LIME π less similar to LIME, which was used as a benchmark for comparison. If Q-LIME π were to be successfully published, which would most likely require its development to reach a level similar to LIME, it would become the first quantum algorithm used for explainability, which would undoubtedly cause an impact in the field of XAI.

A comparison between both a classical NN and a QCNN was also conducted, using both the iris dataset and the IMDb film reviews dataset. Both the classical and quantum models gave significantly different results from each other, which was far from ideal. This disparity was mainly caused by the poor accuracy of the QCNN trained.

The NN trained on the IMDb data achieved strong performance, with an accuracy of 82.8%. Using the global explainer on it, the most important features were selected, which were then used for retraining several more efficient models. Among these, the most efficient model was selected through the BIC. In regards to future improvements for the NN, the main recommendation would be enhancing its accuracy. This would involve addressing overfitting, as mitigating it more would enable training for more epochs, allowing the model to converge.

As mentioned in the discussion, the results obtained from the QCNN were far from optimal, with an accuracy of just 63.5%. However, significant improvements could be obtained by increasing the amount of training data. Further modifying its parameters and only training it on the important features obtained from the NN explanation would also benefit its accuracy.

Overall, the most promising direction for future research would be a direct comparison between the explanations given by the used global explainer, which averages LIME output, with different global explainers, in order to evaluate its relative performance.

5.1 Critical Reflection

Even though it ended in a successful manner, this bachelor thesis research could have been improved in many ways. First of all, I was overly ambitious in what I aimed to accomplish

at the time of my proposal. Analyzing several datasets was not possible, and it would not be without spending a significantly longer amount of time, while the results would most likely not have been as good as the ones obtained by focusing on just the IMDb dataset.

The results obtained from the QCNN could have been improved had I had more time for computation. However, the scope of the work done in this project was overly ambitious, as I spent a very significant amount of time on the literature, mathematical, and computer science aspects, yet I felt constrained by time. Looking back, I could have benefited from further limiting my research.

I am overall very satisfied in how everything turned out. Especially regarding the results obtained in the optimization of the Neural Network, as they provided exciting insights into the linguistics of the sentiment found in film reviews, and showed how the NN model was indeed capable of making informed decisions, choosing reasonable words as the most important ones.

This bachelor thesis research laid a solid foundation for the future development of Q-LIME π. It explored the differences in interpretability and performance between quantum and classical models, gaining insights into their behavior. This work not only showcases the challenges and potential of explainable quantum machine learning, but it also opens several promising directions for future research in this emerging field.

References

[1] J. A. Baktash, M. Dawodi and G. Chat, 'Gpt-4: A review on advancements and opportunities in natural language processing', *Journal of Electrical Electronics Engineering (JEEE)*, vol. 2, no. 4, pp. 548–549, 2023. DOI: 10.33140/JEEE.02.04.19.

[2] P. Baldi, P. Sadowski and D. Whiteson, 'Searching for exotic particles in high-energy physics with deep learning', *Nature Communications*, vol. 5, p. 4308, 2014. DOI: 10.1038/ncomms5308.

[3] V. Belle and I. Papantonis, 'Principles and practice of explainable machine learning', *Frontiers in Big Data*, vol. 4, p. 688 969, 2021. DOI: 10.3389/fdata.2021.688969.

[4] T. M. Khan and A. Robles-Kelly, 'Machine learning: Quantum vs classical', *IEEE Access*, vol. 8, pp. 219 275–219 294, 2020. DOI: 10.1109/ACCESS.2020.3041719.

[5] S. K. Kandula, N. Katam, P. R. Kangari, A. Hijmal, R. Gurrala and M. Mahmoud, 'Quantum computing potentials for drug discovery', in *2023 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2023, pp. 1467–1473. DOI: 10.1109/CSCI62032.2023.00240.

[6] D. J. Egger, C. Gambella, J. Marecek *et al.*, 'Quantum computing for finance: State-of-the-art and future prospects', *IEEE Transactions on Quantum Engineering*, vol. 1, pp. 1–24, 2020. DOI: 10.1109/TQE.2020.3030314.

[7] 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)*, 2022, pp. 8612–8616. DOI: 10.1109/ICASSP43922.2022.9747675.

[8] Y. LeCun, Y. Bengio and G. Hinton, 'Deep learning', *Nature*, vol. 521, pp. 436–44, 2015. DOI: 10.1038/nature14539.

[9] T. Hur, L. Kim and D. K. Park, 'Quantum convolutional neural network for classical data classification', *Quantum Machine Intelligence*, vol. 4, no. 1, 2022, ISSN: 2524-4914. DOI: 10.1007/s42484-021-00061-x.

[10] F. P. Maria Schuld, *Machine Learning with Quantum Computers*, 2nd ed. Springer Nature Switzerland AG, 2021, p. 4.

[11] D. Wierichs, J. Izaac, C. Wang and C. Y.-Y. Lin, 'General parameter-shift rules for quantum gradients', *Quantum*, vol. 6, p. 677, 2022, ISSN: 2521-327X. DOI: 10.22331/q-2022-03-30-677.

[12] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac and N. Killoran, 'Evaluating analytic gradients on quantum hardware', *Phys. Rev. A*, vol. 99, p. 032 331, 3 Mar. 2019. DOI: 10.1103/PhysRevA.99.032331.

[13] G. Ras, N. Xie, M. van Gerven and D. Doran, 'Explainable deep learning: A field guide for the uninitiated', *Journal of Artificial Intelligence Research*, vol. 73, 2022. DOI: https://doi.org/10.1613/jair.1.13200.

[14] D. Slack, S. Hilgard, E. Jia, S. Singh and H. Lakkaraju, 'Fooling lime and shap: Adversarial attacks on post hoc explanation methods', in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, ser. AIES '20, New York, NY, USA: Association for Computing Machinery, 2020, pp. 180–186, ISBN: 9781450371100. DOI: 10.1145/3375627.3375830. [Online]. Available: https://doi.org/10.1145/3375627.3375830.

[15] C. G. Marco Tulio Ribeiro Sameer Singh, '"why should i trust you?": Explaining the predictions of any classifier', *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016. DOI: 10.1145/2939672.2939778.

[16] D. Garreau and U. von Luxburg, 'Explaining the explainer: A first theoretical analysis of LIME', in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 108, 2020, pp. 1287–1296. [Online]. Available: https://proceedings.mlr.press/v108/garreau20a.html.

[17] M. T. Ribeiro, S. Singh and C. Guestrin, 'Anchors: High-precision model-agnostic explanations', *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018. DOI: 10.1609/aaai.v32i1.11491.

[18] P. Sai Ram Aditya and M. Pal, *Local interpretable model agnostic shap explanations for machine learning models*, 2022. DOI: 10.48550/arXiv.2210.04533.

[19] N. C. Vargas, *Q-lime π: A quantum-inspired extension to lime*, arXiv preprint, 2024. arXiv: 2412.17197 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2412.17197.

[20] I. L. Markov and Y. Shi, 'Simulating quantum computation by contracting tensor networks', *SIAM Journal on Computing*, vol. 38, no. 3, pp. 963–981, 2008. DOI: 10.1137/050644756.

[21] T. B. Brown, B. Mann, N. Ryder *et al.*, 'Language models are few-shot learners', in *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, Curran Associates, Inc., 2020, pp. 1877–1901. DOI: 10.5555/3495724.3495883. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[22] D. Rozado, 'The political preferences of LLMs', *PLOS ONE*, vol. 19, no. 7, e0306621, 2024. DOI: 10.1371/journal.pone.0306621. [Online]. Available: https://doi.org/10.1371/journal.pone.0306621.

[23] L. Shapley and R. C. S. M. CALIF., *A Value for N-person Games*. Defense Technical Information Center. [Online]. Available: https://books.google.es/books?id=QzW6tgAACAAJ.

[24] F. Di Martino and F. Delmastro, 'Explainable ai for clinical and remote health applications: A survey on tabular and time series data', *Artificial Intelligence Review*, vol. 56, no. 4, pp. 5261–5315, 2023, Published 26 October 2022, Issue Date June 2023. DOI: 10.1007/s10462-022-10304-3. [Online]. Available: https://doi.org/10.1007/s10462-022-10304-3.

[25] P. Xenopoulos, G. Chan, H. Doraiswamy, L. G. Nonato, B. Barr and C. T. Silva, 'Gale: Globally assessing local explanations', in *Topological, Algebraic and Geometric Learning Workshops 2022, 25-22 July 2022, Virtual*, A. Cloninger, T. Doster, T. Emerson *et al.*, Eds., ser. Proceedings of Machine Learning Research, vol. 196, PMLR, 2022, pp. 322–331. [Online]. Available: https://proceedings.mlr.press/v196/xenopoulos22a.html.

[26] G. Schwarz, 'Estimating the dimension of a model', *Annals of Statistics*, vol. 6, pp. 461–464, 1978. DOI: Https://doi.org/10.1214/aos/1176344136.

6 Appendix

The repository used in this project can be accessed in this GitHub. In it, you will find all the results, both summarized and as raw data, code, trained models, and sources for the data used throughout this project. Due to the huge amount of files, it is organized by folders.