applied sciences

IMPACT FACTOR 2.5

CITESCORE 5.3

Article

Optimization Strategies in Quantum Machine Learning: A Performance Analysis

Nouf Ali AL Ajmi and Muhammad Shoaib

MDPI

https://doi.org/10.3390/app15084493

applied sciences

MDPI

Article

Optimization Strategies in Quantum Machine Learning: A Performance Analysis

Nouf Ali AL Ajmi * and Muhammad Shoaib *

Department of Information Systems, College of Computer and Information Sciences, King Saudi University, Riyadh 11451, Saudi Arabia
* Correspondence: naalajmi@ksu.edu.sa (N.A.A.A.); muhshoaib@ksu.edu.sa (M.S.)

Abstract: This study presents a comprehensive comparison of multiple optimization algorithms applied to a quantum classification model, utilizing the Cleveland dataset. Specifically, the research focuses on three prominent optimizers—COBYLA, L-BFGS-B, and ADAM—each employing distinct methodologies and widely recognized in the domain of quantum machine learning. The performance of predictive models using these optimizers is rigorously evaluated through key metrics, including accuracy, precision, recall, and F1 score. The findings reveal that the COBYLA optimizer outperforms the L-BFGS-B and ADAM optimizers across all performance metrics, achieving an accuracy of 92%, precision of 89%, recall of 97%, and F1 score of 93%. Furthermore, the COBYLA optimizer exhibits superior computational efficiency, requiring only 1 min of training time compared to 6 min for L-BFGS-B and 10 min for ADAM. These results underscore the critical role played by optimizer selection in enhancing model performance and efficiency in quantum machine learning applications, offering valuable insights for practitioners in the field.

Keywords: quantum computing; quantum machine learning; optimizer; COBYLA; L-BFGS-B; ADAM

Received: 4 March 2025
Revised: 9 April 2025
Accepted: 15 April 2025
Published: 18 April 2025

Citation: AL Ajmi, N.A.; Shoaib, M. Optimization Strategies in Quantum Machine Learning: A Performance Analysis. *Appl. Sci.* **2025**, *15*, 4493. https://doi.org/10.3390/app15084493

1. Introduction

Machine learning has undergone significant evolution, transitioning from solving small-scale pattern recognition problems to efficiently uncovering complex structures within vast datasets and enabling broad generalizations. As the volume and complexity of data continue to grow, the computational demands placed on classical systems have reached a critical threshold, challenging their ability to process information effectively. This rapid advancement in machine learning has spurred the need for alternative computational paradigms capable of overcoming the inherent limitations of classical hardware.

The progression of classical computing has been remarkable, with machines evolving from large, costly systems to compact, high-performance devices due to advancements in hardware design, components, and software optimization. However, as electronic circuits approach atomic scales, the scalability of classical computers is fundamentally constrained by physical and technological barriers. In contrast, quantum computing offers a transformative theoretical framework that promises to surpass these limitations, providing unprecedented computational power and scalability [1–3].

The origins of quantum computing can be traced back to the 1980s when researchers first explored the potential of leveraging quantum properties for computation. Since then, dedicated efforts have been made to understand its computational capabilities and unlock its potential applications [4]. A landmark achievement in this field was Peter Shor's groundbreaking algorithm in 1994, which demonstrated the ability to factorize large

numbers in polynomial time, a task that is infeasible for classical systems [5]. This work not only marked a pivotal moment in quantum computing but also catalyzed widespread interest and further research into the domain.

In 2016, the advent of cloud-based quantum computing (IBM Newsroom) marked a significant milestone, making quantum resources accessible to researchers and developers worldwide. However, the practical implementation of intricate quantum algorithms has been hindered by the limitations of Noisy Intermediate-Scale Quantum (NISQ) devices, which are characterized by noise and restricted qubit coherence [6]. Despite these challenges, the integration of quantum computing with machine learning has emerged as one of the most promising applications of quantum technologies. Quantum machine learning seeks to harness the unique properties of quantum systems to address complex machine learning problems, offering the potential for exponential speedups and enhanced performance [7,8].

Among the innovative methodologies in this field, Quantum Neural Networks (QNNs) have garnered significant attention as a powerful framework for solving real-world challenges [9]. Research has demonstrated the practical applicability of QNNs, moving beyond theoretical constructs to deliver tangible solutions to complex problems [10]. In this study, we employ QNNs as the foundational architecture, highlighting the advanced capabilities embedded within this model.

This paper is structured as follows: Section 2 provides an overview of variational quantum algorithms, the principles underlying QNNs, and the challenges involved. Section 3 delves into the architectural design of the QNN model employed in this study. Section 4 introduces the concept of optimizers and outlines the specific optimizers utilized. Section 5 presents a comparative analysis of the performance of these optimizers, and Section 6 concludes with a summary of the key findings and future directions.

2. Literature Review and Background

2.1. Variational Quantum Algorithms (VQAs)

Variational quantum algorithms (VQAs) have emerged as a leading paradigm for achieving a quantum advantage on Noisy Intermediate-Scale Quantum (NISQ) devices. These algorithms have demonstrated versatility across a wide range of scientific and computational problems, including solving linear systems of equations, simulating quantum dynamics, and determining the ground states of molecules.

VQAs operate within a hybrid quantum–classical framework, leveraging the complementary strengths of quantum and classical computing. The core structure of VQAs involves encoding data into a parameterized cost function, which is then evaluated using a quantum system, such as a quantum simulator or hardware. Subsequently, a classical optimizer is employed to iteratively refine the parameters of the quantum circuit, aiming to minimize the cost function [11–13]. This iterative optimization process enables VQAs to effectively address complex problems while mitigating the impact of noise inherent in NISQ devices. Figure 1 illustrates the general architecture of a VQA that operates through the collaboration of a quantum simulator and a classical computer system. The algorithm takes classical data as its input, processed by a parameterized quantum circuit that encodes the quantum model. Following this computation, the resultant predictions are transferred to the classical component, where a cost function evaluates the performance and iteratively adjusts the model parameters. These optimized parameters are subsequently relayed back to the quantum circuit, forming a recursive feedback loop to minimize the cost metric and enhance model accuracy.

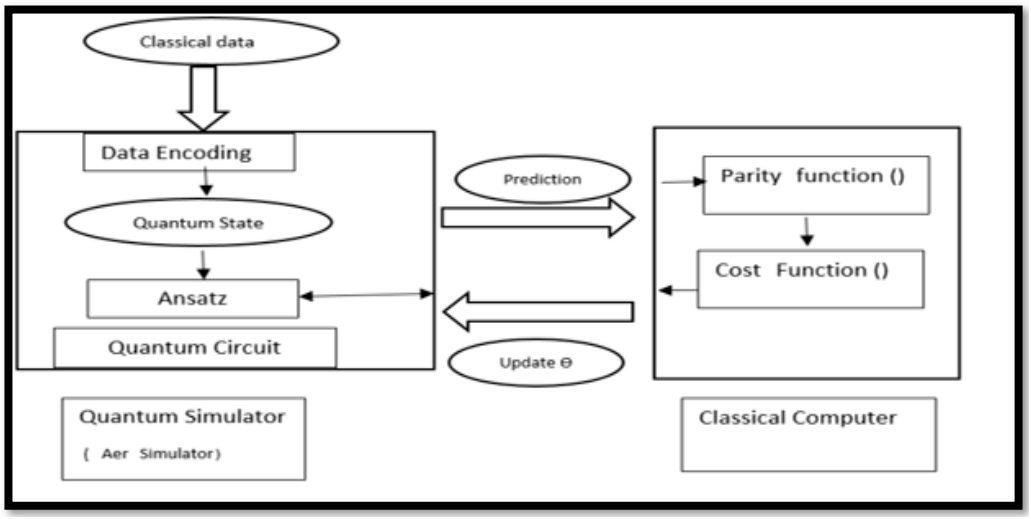

Figure 1. General VQA architecture.

This process, aided by parity measurements—which play a crucial role in quantum machine learning (QML) by transforming quantum state information into classical prediction outcomes—effectively bridges the gap between quantum computation and classical interpretation. This hybrid architecture enables efficient optimization by leveraging quantum computational resources for state space exploration while retaining classical systems for scalable parameter tuning and cost analyses.

2.2. Quantum Neural Networks (QNNs)

Quantum machine learning (QML) represents a synergistic fusion of principles from quantum computing and classical machine learning, intended for the development of novel methodologies that enhance computational efficiency and performance. A key innovation in this domain is the Quantum Neural Network (QNN), a machine learning model that exploits quantum phenomena such as entanglement and superposition to perform computations. QNNs are inspired by the principles of quantum physics and quantum computing, where neurons are represented as quantum states, and transitions between these states are governed by operations derived from quantum logic gates [14].

QNNs hold significant potential to outperform classical neural networks in various applications. Several models and frameworks for designing and implementing QNNs have been proposed, showcasing their ability to leverage quantum computing principles for superior data processing. Compared to classical neural networks, QNNs exhibit enhanced efficiency in recognizing patterns and performing classification tasks. They also require fewer parameters and reduced training times thanks to the parallelism inherent in quantum computing. Furthermore, the use of linear functions in QNN training can mitigate overfitting and improve the generalization capabilities of the network [15–18].

Ezhov and Ventura [19] have extensively discussed the advantages of QNNs over classical artificial neural networks (ANNs). The key benefits include quantum parallelism, exponentially increased memory capacity, faster learning rates, improved performance with fewer hidden neurons, the ability to solve linearly inseparable problems using single-layer networks, and high-speed information processing. These attributes position QNNs as a promising tool for addressing challenges that are computationally infeasible for classical systems.

Despite these advantages, the development of QNNs faces several challenges. Much of the research in this field has focused on mapping classical neural network components to their quantum counterparts. However, this is not always straightforward due to the absence of direct quantum analogs for certain classical components, such as non-linear activation functions. Additionally, all operations in quantum algorithms must be reversible,

imposing constraints on the design of the QNN architecture. As a result, there is no unified consensus on the precise definition of QNNs within the academic community [10,20,21]. Nevertheless, ongoing research continues to explore innovative approaches to overcome these limitations and unlock the full potential of QNNs.

Optimization algorithms are integral to the training of QNNs as they iteratively adjust quantum circuit parameters to minimize the cost function. Among the available approaches, gradient-free optimization methods are particularly notable. These techniques circumvent the need for gradient calculations during backpropagation, making them well suited for noisy or computationally complex quantum algorithms [22]. A prominent example is the COBYLA (Constrained Optimization by Linear Approximation) optimizer, which has demonstrated exceptional performance in various studies. For instance, in [23], COBYLA outperformed two other optimizers—SPSA and SLSQP—highlighting its efficacy. Furthermore, COBYLA was successfully implemented in the QNN framework presented by Ajibosin et al. [23].

In contrast, gradient-based optimizers rely on the computation of gradients within the QNN. While these methods typically demand more computational time compared to gradient-free approaches, they provide the benefit of convergence guarantees, making them a robust choice for certain applications [22]. One widely recognized gradient-based algorithm is ADAM (Adaptive Moment Estimation), which has been effectively utilized in QNN contexts. For example, in a study by Al-Zafar Khane et al., the ADAM optimizer enabled a QNN to achieve optimal performance [24]. Another notable gradient-based method is L-BFGS-B (Limited-memory Broyden–Fletcher–Goldfarb–Shanno with Bound Constraints), which was employed in [25]; the results underscored its effectiveness in enhancing model performance.

The choice of optimization method is highly dependent on the specific problem, dataset characteristics, and quantum hardware limitations. As quantum machine learning research progresses, advancements in optimization techniques will be pivotal to improving the performance, scalability, and practical applicability of QNNs [22].

2.3. Challenges and Limitations in Quantum Machine Learning Optimization

Despite the promising advancements in quantum machine learning, several challenges and limitations persist, particularly in the context of optimization.

A. Noise and Decoherence

Noisy Intermediate-Scale Quantum (NISQ) devices are inherently prone to errors due to noise and decoherence. These factors can significantly impact the performance of optimizers, especially those relying on precise gradient evaluations. For example, ADAM's sensitivity to noise can lead to suboptimal convergence, as observed in our results. Future work should explore noise-resilient optimization techniques, such as error mitigation strategies and stochastic gradient descent estimation methods [26,27].

B. Scalability

While quantum computing holds the promise of scalability, current NISQ devices are limited in terms of their qubit count and coherence time. This limitation restricts the size and complexity of models that can be trained effectively. Optimizers like COBYLA, which require fewer iterations, are advantageous in this context but may not scale well to larger problems. Developing scalable optimization algorithms that balance computational efficiency and accuracy remains a critical area of research [28].

C. Hybrid Approaches

Hybrid quantum–classical optimization approaches, such as VQAs, offer a practical solution for leveraging quantum resources in the NISQ era. However, the interplay between

quantum and classical components introduces additional complexities. For instance, the choice of ansatz and optimizer must be carefully aligned to ensure effective parameter tuning. Future research should focus on designing adaptive hybrid frameworks that dynamically adjust to the problem's characteristics [28].

D. Benchmarking and Standardization

The lack of standardized benchmarks for evaluating quantum optimizers poses a significant challenge. Current studies often rely on custom datasets and metrics, making it difficult to compare results across different works. Establishing a unified benchmarking framework would facilitate more rigorous evaluations and accelerate progress in the field [29].

3. Model Architecture

3.1. Optimization Problem

A model was designed to predict heart disease diagnoses using the Cleveland Heart Disease dataset from the UCI repository for building and training [29]. The Cleveland dataset is widely recognized and frequently utilized by researchers for their studies, as in [30,31]. The task involves classifying input data into two categories: the absence or presence of heart disease. This binary classification problem serves as the foundation for evaluating the performance of quantum machine learning techniques.

3.2. General Architecture

The model follows the principles of variational quantum algorithms (VQAs), which combine classical and quantum computing to solve optimization problems. First, we preprocess the classical data and select relevant attributes. The processed data are then encoded into quantum states, and the subsequent optimization process leverages the hybrid framework of VQAs.

A key component of any VQA is the cost function, which maps trainable parameters (θ) to real numbers. The cost function evaluates how well the quantum circuit approximates the desired solution. Another critical element is the ansatz, a parameterized quantum circuit that defines the structure of the trial wave function. The ansatz plays a pivotal role in determining how efficiently the parameters θ can be optimized to minimize the cost function [11]. The success of the VQA heavily depends on the optimizer's ability to effectively adjust these parameters during training.

3.3. Model QNN

A. Data preprocessing:

A two-step feature selection process was proposed to identify and select the most influential features from the dataset. The first step is based on information gain (IG), which involves calculating the gain ratio for each feature, as shown in the following equation:

$$IG\,(C, X) = H(C) - H\,(C \mid X)$$

Here, H(C) represents the entropy of the class label C, quantifying the level of uncertainty or randomness in the class distribution. It serves as a measure of the unpredictability or variability in the target variable. In contrast, H (C | X) denotes the conditional entropy of class C given a specific feature X, representing the remaining uncertainty about C when X is known. By comparing these entropy measures, we can evaluate the information gain provided by each feature and determine its relevance to the class labels [32].

After computing the information gain for all features, they are ranked in descending order based on their scores. Features with low information gain are considered to have

a minimal impact on the classification task and may be excluded from further analysis (Figure 2).

```
> print(attribute_info_gain)
     thal    Oldpeak      Thali         cp        sex         ca
26.0407411 19.3937946 13.7525317 12.8829666 12.6137863 12.3581672
    Exang        age       chol   trestpbs      slope        FBS
11.0851689  7.3102895  2.8452903  2.3055470  0.2738672  0.2407136
```

Figure 2. IG of the features.

For the second step, we employ Principal Component Analysis (PCA) to reduce dimensionality. PCA is a linear projection technique that transforms data from a high-dimensional space to a lower-dimensional space, minimizing information loss. It is widely used for feature extraction in healthcare classification [30].

To apply PCA, we first standardize and normalize the top eight features (selected based on information gain, IG) using standard scaling. This prevents features with larger scales from disproportionately influencing the PCA results. Then, PCA is used to reduce the dimensionality of the original eight features to four principal components. These components, linear combinations of the original features, capture the essential data while eliminating redundancy, simplifying the input for quantum machine learning (QML) algorithms. The choice of four components was determined empirically. Figure 3 illustrates the contribution of the original features to each principal component:

```
Contribution of original features to each principal component:
        age     sex      cp    thal  oldpeak   exang      ca  thalach
PC1 -0.3154 -0.1488  0.3649 -0.2772  -0.4018 -0.4262 -0.3296   0.4655
PC2 -0.5627  0.5965 -0.1518  0.4168  -0.0264  0.2141 -0.1229   0.2629
PC3  0.2631  0.3582  0.5012  0.2988   0.1537 -0.4302  0.4786   0.1556
PC4 -0.0233  0.0112 -0.4968 -0.1287  -0.5492 -0.1102  0.6273   0.1697
```

Figure 3. Features' contribution to each principal component.

B. Data encoding:

Quantum data encoding is a crucial step in translating classical data into quantum states. Various encoding techniques, such as basis encoding, angle encoding, and amplitude encoding, are employed depending on the application and dataset characteristics [33].

In this model, the Z feature map is used to encode classical data into a quantum state. A four-qubit quantum register, labeled 'q', is utilized, with operations applied to qubits 0, 1, 2, and 3. The encoding process involves the following sequence of operations for each qubit:

1. Apply a Hadamard gate (H);
2. Apply a parameterized Phase gate (P) with a rotation angle defined as 2.0*x[i], where x[i] represents the classical input feature;
3. Apply another Hadamard gate (H);
4. Apply another parameterized Phase gate (P).

This sequence (H-P-H-P) is repeated for all four qubits, creating a complex superposition state that captures non-linear relationships in the data. Figure 4 illustrates the detailed structure of the encoding circuit, which indicates the circuit construction of the Hadamard gate 'h', to create a superposition, and the Phase gate 'p' applied to all four

qubits in succession, from qubit 0 to qubit 3. In the end, four-dimensional classical data are encoded into a four-qubit quantum state using this specific pattern of gates.

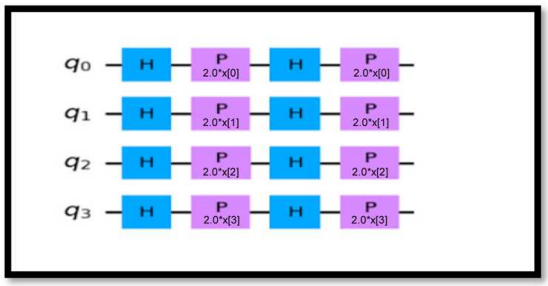

Figure 4. Decomposition of Z feature map gate.

C. Ansatz:

An ansatz in quantum computing refers to a parameterized trial wave function that guides the optimization process. It is implemented as a parameterized quantum circuit, enabling the exploration of various quantum states through adjustments to the trainable parameters (θ). The ansatz influences how these parameters (θ) can be optimized or trained to minimize the cost function in the algorithm. Thus, the choice of ansatz significantly influences the efficiency and effectiveness of the optimization process [11,33].

For this model, the Real Amplitudes ansatz was selected due to its simplicity and computational efficiency. This ansatz consists of layered RY gates and Controlled-X (CNOT) gates, which introduce entanglement between qubits [33,34]. The specific details are as follows:

- The model employs a four-qubit Real Amplitudes ansatz with two layers of gates;
- In each layer, an RY gate is applied to each qubit with rotation angle θ;
- The rotation angles θ are adjusted iteratively during the optimization process to minimize the cost function;
- CNOT gates are applied to create entanglement between qubits [35].

Figure 5 illustrates the Real Amplitudes ansatz unit circuit:

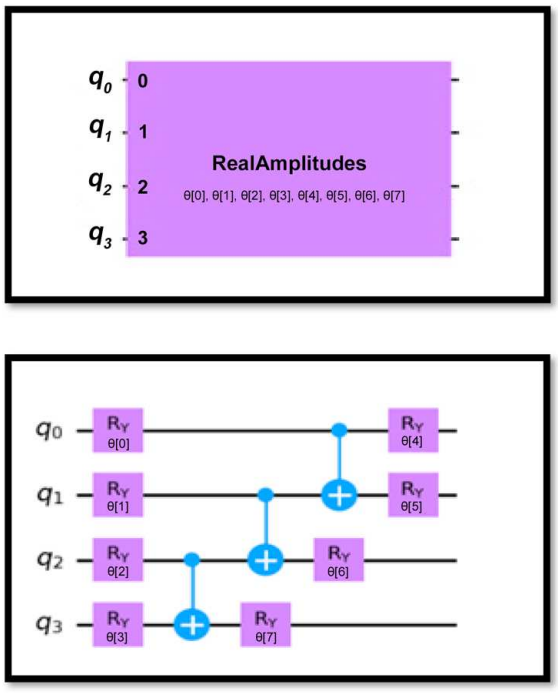

Figure 5. Real Amplitudes ansatz unit circuit.

The mathematical equation is

$$RealAmplitudes = \prod_{i=0}^{n-1} RY(\theta_i^1) \ . \prod_{j=0}^{n-2} CNOT\ (\ j,\ j+1) \ . \prod_{i=0}^{n-1} RY(\theta_i^2)$$

where

- RY (θ) represents the rotation gate around the Y-axis with rotation angle θ;
- CNOT(j, j + 1) denotes the CNOT (Controlled-NOT) gate with control qubit j and target qubit j + 1;
- N is the number of qubits, four in this case.

D. Cost function:

The cost function is a crucial aspect of the VQA, measuring the discrepancy between the quantum circuit's output and the desired result. It quantifies how well the trainable parameters θ approximate the target quantum solution [11]. In this study, the absolute error loss function was employed to evaluate the model's performance. For each data point, the absolute error is computed as the absolute difference between the predicted value and the true label [36]. This metric ensures that the optimization process focuses on minimizing prediction errors across the dataset.

E. Model Building:

The model utilized the Cleveland dataset, which contains 303 instances with 14 features relevant to heart disease. The data were split into 80% for training and 20% for testing.

All experiments were conducted using Colab Python, 3.11.12 with the model running on an Aer Simulator. The simulations were noise-free, employing 1024 shots per circuit execution. Furthermore, a random seed has been set to the common value of 42 to ensure reproducibility.

The following Python packages were used in this research:

- Data Visualization: matplotlib.pyplot and seaborn.
- Data Preprocessing: sklearn.preprocessing (StandardScaler, MinMaxScaler) for scaling, sklearn.model_selection (train_test_split) for splitting, and sklearn.decomposition (PCA) for dimensionality reduction.
- Performance Evaluation: sklearn.metrics.
- Quantum Computing: qiskit for circuit design and simulation and qiskit_machine_learning (NeuralNetworkClassifier, CircuitQNN) for quantum neural network implementation.
- Optimization: qiskit.algorithms.optimizers (L-BFGS-B, COBYLA, ADAM).

4. Optimizers

The choice of optimizer is a critical factor that significantly impacts the performance of the algorithm. This section provides an overview of three widely used optimizers, each employing distinct mechanisms to optimize the parameter θ. They are the Constrained Optimization by Linear Approximation (COBYLA) optimizer, a derivative-free optimization algorithm designed for solving constrained optimization problems; the Limited-memory Broyden–Fletcher–Goldfarb–Shanno with Bound Constraints (L-BFGS-B) optimizer, a gradient-based optimizer belonging to the quasi-Newton methods; and the Adaptive Moment Estimation (ADAM) optimizer, a gradient-based optimizer [37–39].

4.1. COBYLA

COBYLA is a derivative-free optimizer designed for constrained optimization problems, particularly when the objective function's gradient is unavailable. It constructs a linear approximation of the objective function and constraints for n + 1 points in space,

enabling work within a trust region and providing efficient optimization without requiring explicit bounds. COBYLA is well suited for noise-free objective functions and minimizes the number of evaluations, resulting in shorter runtimes [40–42].

4.2. L_BFGS_B

L-BFGS-B is a quasi-Newton method (as described in "Machine Learning in Quasi-Newton Methods") tailored for bound-constrained optimization problems. It avoids the need for the Hessian matrix by iteratively approximating second-order derivatives. This optimizer is robust and effective when using an iterative approach to solve unconstrained, nonlinear optimization problems, especially in scenarios with limited memory resources [43,44].

4.3. ADAM

ADAM is a gradient-based optimizer that adjusts the learning rates for each parameter using estimates of the first and second moments of the gradients. Its momentum mechanism helps stabilize convergence by guiding the gradient updates in the correct direction. While ADAM accelerates convergence and improves generalization, it can sometimes overshoot the minimum due to excessive momentum, potentially leading to suboptimal performance [45,46].

Each optimizer has unique strengths and limitations, and their suitability depends on the specific characteristics of the optimization problem. A comparative analysis of their performance within the context of the model constructed in this study is presented in the subsequent section.

5. Comparative Analysis

This section presents a detailed evaluation and comparison of three optimizers—COBYLA, L-BFGS-B, and ADAM—based on their performance and characteristics in addressing the optimization challenges of the quantum machine learning model constructed in this study. The analysis aims to provide a comprehensive assessment of these optimizers, emphasizing their respective strengths, weaknesses, and overall suitability for the task at hand.

5.1. Performance Evaluation

To comprehensively assess the effectiveness of predictive models optimized using different optimizers, a range of performance metrics—accuracy, precision, recall, and F1 score—were employed in this study. These metrics provide a quantitative evaluation of the model's classification performance and are derived from the confusion matrix, which serves as the foundation for their computation.

Confusion Matrix

The confusion matrix is a fundamental tool for evaluating classification models by organizing predictions into four categories:

$$\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix} \quad (1)$$

where

True positive (TP): the number of instances correctly classified as positive;
False positive (FP): the number of instances incorrectly classified as positive;
True negative (TN): the number of instances correctly classified as negative;
False negative (FN): the number of instances incorrectly classified as negative.
Using these components, the following performance metrics are calculated.

1. Accuracy

Accuracy measures the proportion of correctly classified instances (both positive and negative) out of the total number of instances. It is defined as follows:

$$Accuracy = \frac{TP + TN}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{2}$$

While accuracy provides an overall sense of model performance, it may not be sufficient for imbalanced datasets where one class dominates the other.

2. Precision

Precision evaluates the proportion of true positive predictions among all instances predicted as positive. It is particularly useful in scenarios where minimizing false positives is critical. Precision is calculated as follows:

$$Precision = \frac{TP}{\mathrm{TP} + \mathrm{FP}} \tag{3}$$

High precision indicates that the model makes accurate positive predictions with minimal false positives.

3. Recall (Sensitivity)

Recall, also known as sensitivity, measures the proportion of actual positive instances that are correctly identified by the model. It is especially important in applications where detecting all positive cases is prioritized. Recall is calculated as follows:

$$Recall = \frac{TP}{\mathrm{TP} + \mathrm{FN}} \tag{4}$$

A high recall value signifies that the model effectively captures most of the positive instances.

4. F1 Score

The F1 score is the harmonic mean of precision and recall, providing a balanced measure that accounts for both false positives and false negatives. It is particularly valuable when there is an uneven class distribution. The F1 score is computed as follows:

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5}$$

The F1 score ranges between 0 and 1, with higher values indicating better overall performance [46].

These metrics collectively offer a nuanced understanding of a model's predictive capabilities. By leveraging the confusion matrix as the basis for computation, they enable a detailed assessment of the trade-offs between correct and incorrect predictions across different classes. The subsequent sections present the results of these metrics for models optimized using various algorithms, providing insights into their relative strengths and limitations.

This section evaluates the performance of three optimizers—COBYLA, L-BFGS-B, and ADAM—on the Cleveland dataset. The analysis is based on the confusion matrices derived from each optimizer's predictions, followed by the computation of key performance metrics: accuracy, precision, recall (sensitivity), and F1 score. Additionally, the training times are reported to assess the optimizers' computational efficiency.

5.2. COBYLA Optimizer

Using COBYLA, a gradient-free optimizer, with a maximum of 100 iterations, we obtained the following confusion matrix. Since COBYLA is gradient-free, a learning rate is not applicable.

$$\begin{bmatrix} 25 & 4 \\ 1 & 31 \end{bmatrix} \tag{6}$$

True positive (TP): Twenty-five instances were correctly predicted as positive.

False positive (FP): Four instances were incorrectly predicted as positive.

False negative (FN): One instance was incorrectly predicted as negative.

True negative (TN): Thirty-one instances were correctly predicted as negative.

Using the confusion matrix values, the performance metrics for the COBYLA optimizer were calculated as follows:

$$\text{Accuracy} = \frac{31+25}{31+4+25+1} = 0.918 = 92\%$$

$$\text{Precision} = \frac{31}{31+4} = 0.886 = 89\%$$

$$\text{Recall (Sensitivity)} = \frac{31}{31+1} = 0.969 = 97\%$$

$$\text{F1 score} = 2 \times \frac{0.886 \times 0.969}{0.886 + 0.969} = 0.925 = 93\%$$

Additionally, the model trained using the COBYLA optimizer achieved these results in just 1 min, making it the most computationally efficient optimizer.

5.3. L_BFGS_B Optimizer

We run L-BFGS-B, a gradient-based optimizer with parameter-shift gradients, and set the maximum iterations to 22. This optimizer dynamically adjusts the learning rate, removing the need for a fixed value. The resulting confusion matrix is shown below:

$$\begin{bmatrix} 24 & 5 \\ 2 & 30 \end{bmatrix} \tag{7}$$

True positive (TP): Twenty-four instances were correctly predicted as positive.

False positive (FP): Five instances were incorrectly predicted as positive.

False negative (FN): Two instances were incorrectly predicted as negative.

True negative (TN): Thirty instances were correctly predicted as negative.

Using the confusion matrix values, the performance metrics for the L-BFGS-B optimizer were calculated as follows:

$$\text{Accuracy} = \frac{30+24}{30+5+24+2} = 0.885 = 89\%$$

$$\text{Precision} = \frac{30}{30+5} = 0.857 = 86\%$$

$$\text{Recall (Sensitivity)} = \frac{30}{30+2} = 0.937 = 94\%$$

$$\text{F1 score} = 2 \times \frac{0.857 \times 0.937}{0.857 + 0.937} = 0.896 = 90\%$$

The model trained using the L-BFGS-B optimizer required 6 min, which is significantly longer than the time taken with COBYLA but still relatively efficient.

5.4. ADAM Optimizer

We utilized the ADAM optimizer, a gradient-based algorithm with parameter-shift gradients. The default learning rate of 0.001 was applied, and the optimization process was conducted for a maximum of 22 iterations. The confusion matrix resulting from the ADAM optimizer is presented below:

$$\begin{bmatrix} 1 & 28 \\ 1 & 31 \end{bmatrix} \tag{8}$$

True positive (TP): Only one instance was correctly predicted as positive.

False positive (FP): Twenty-eight instances were incorrectly predicted as positive.

False negative (FN): One instance was incorrectly predicted as negative.

True negative (TN): Thirty-one instances were correctly predicted as negative.

Using the confusion matrix values, the performance metrics for the ADAM optimizer were calculated as follows:

$$\text{Accuracy} = \frac{31+1}{31+28+1+1} = 0.524 = 52\%$$

$$\text{Precision} = \frac{31}{31+28} = 0.525 = 53\%$$

$$\text{Recall (Sensitivity)} = \frac{31}{31+1} = 0.9688 = 97\%$$

$$\text{F1 score} = 2 \times \frac{0.5254 \times 0.9688}{0.5254 + 0.9688} = 0.6808 = 68\%$$

The model trained using the ADAM optimizer required 10 min, the longest training time among the three optimizers.

The performance metrics for all three optimizers are summarized below.

Table 1 presents a comparative analysis of three optimizers across various performance metrics, revealing significant differences in their effectiveness. The confusion matrices are compared below to provide a detailed evaluation of each model's classification performance, highlighting how well each optimizer distinguishes between classes:

Table 1. Performance metrics of optimizers.

Optimizer	Accuracy	Precision	Recall	F1 Score	Training Time	Converges	Loss
COBYLA	92%	89%	97%	93%	1	100%	8.20%
L-BFGS-B	89%	86%	94%	90%	6	100%	11.84%
ADAM	52%	53%	97%	68%	10	100%	47.54%

An interpretation of the results in Table 2 shows the following:

COBYLA exhibits the best performance with the highest Correctly Identified Positive (25) and Correctly Identified Negative (31) counts, and the lowest Incorrectly Identified Negative (1) and Incorrectly Identified Positive (4) counts. This suggests that it is the most accurate optimizer for this specific problem.

L-BFGS-B performs slightly worse than COBYLA, with a slightly lower Correctly Identified Positive (24) and Correctly Identified Negative (30) count and slightly higher Incorrectly Identified Positive (5) and Incorrectly Identified Negative (2) counts. However, it still performs considerably better than ADAM.

ADAM shows a very poor performance. While it has a high Correctly Identified Negatives count (31), its Correctly Identified Positives count is extremely low (1), and its Incorrectly Identified Positives count is very high (28). This indicates that ADAM is incorrectly identifying a large number of negative instances as positive.

Table 2. Confusion matrices of optimizers.

Optimizer	True Positive	False Positive	False Negative	True Negative
COBYLA	25	4	1	31
L-BFGS-B	24	5	2	30
ADAM	1	28	1	31

Figures 6 and 7 present comparisons of the confusion matrices and performance of the three optimizers.

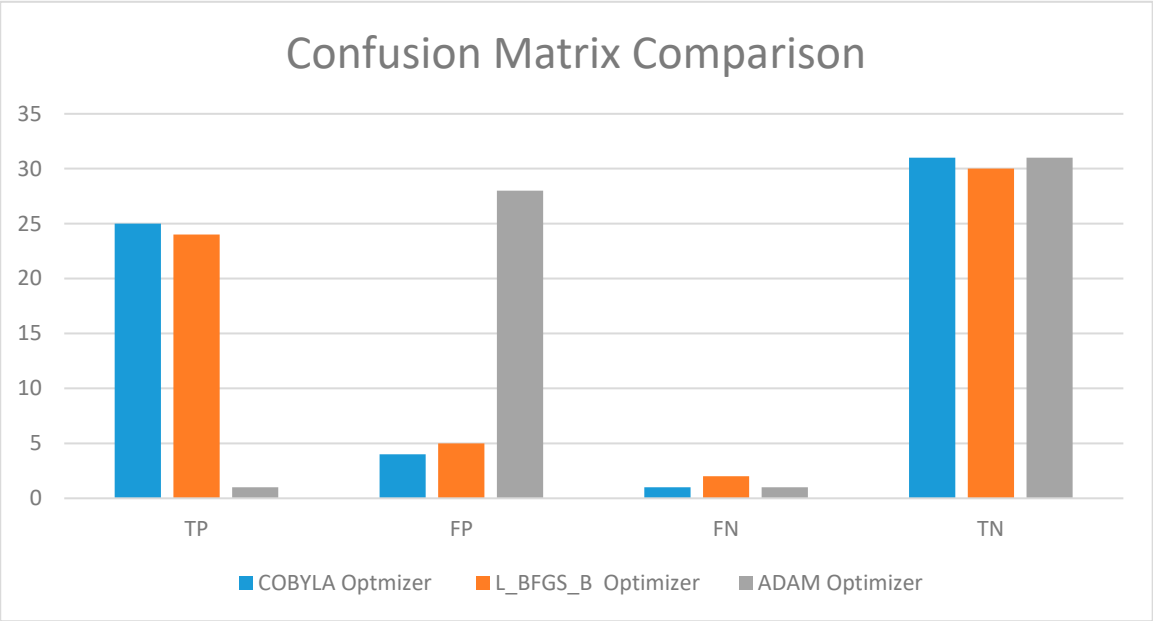

Figure 6. Comparison of confusion matrices.

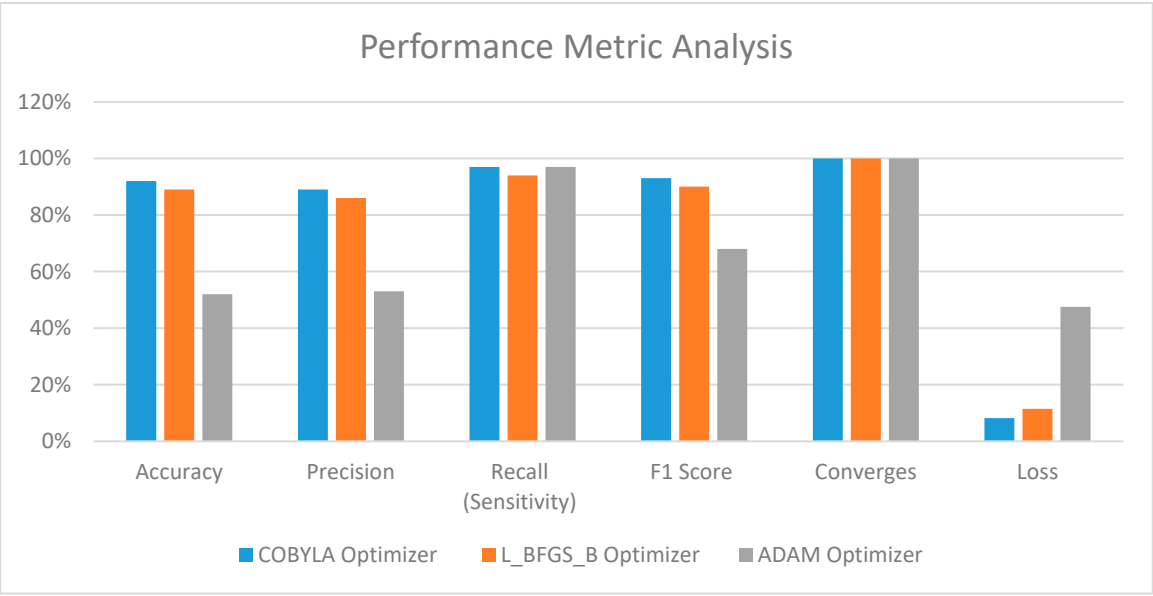

Figure 7. Performance analysis.

These findings highlight the critical role played by optimizer selection in achieving both high performance and computational efficiency in quantum machine learning models.

Optimizer Performance Analysis

A comparative evaluation of the three optimizers—COBYLA, L-BFGS-B, and ADAM—reveals distinct strengths and weaknesses in their performance across various metrics. This analysis focuses on accuracy, precision, recall, and F1 score, providing a comprehensive understanding of the optimizers' suitability for the quantum classification model applied to the Cleveland dataset.

1. Accuracy: COBYLA emerged as the top performer with an impressive accuracy rate of 92%, making it the most reliable optimizer for this task. L-BFGS-B followed with a respectable accuracy of 89%, while ADAM lagged significantly behind at 52%. This stark difference underscores COBYLA's superiority in correctly classifying instances.
2. Precision: Precision measures the proportion of true positive predictions among all positive classifications. COBYLA led once again with a precision rate of 89%, demonstrating its ability to effectively minimize false positives. L-BFGS-B achieved a slightly lower precision of 86%, maintaining reasonable performance. In contrast, ADAM exhibited a markedly low precision of 53%, indicating a high rate of incorrect positive predictions.
3. Recall (Sensitivity): Recall evaluates a model's ability to identify all actual positive instances. Both COBYLA and ADAM achieved a recall rate of 97%, showcasing their efficiency in minimizing false negatives. L-BFGS-B trailed slightly with a recall of 94%, which, while still strong, suggests a marginally higher tendency to miss positive cases.
4. F1 Score: F1 score provides a balanced measure of precision and recall, offering insight into the overall effectiveness of a model. COBYLA achieved the highest F1 score of 93%, reflecting its superior ability to balance these two critical metrics. L-BFGS-B exhibited an F1 score of 90%, indicating a good performance but falling short of COBYLA's optimization capabilities. ADAM achieved the lowest value of 68% on the F1 score.
5. Convergence: The performance of the optimizers reveals distinct differences in loss, despite all three—L-BFGS-B, COBYLA, and ADAM—achieving 100% convergence.
6. Loss: COBYLA demonstrated the lowest loss at 8.20%, indicating superior optimization in this context. L-BFGS-B resulted in a loss of 11.48%, which is notably higher than COBYLA's. In contrast, ADAM exhibited a significantly elevated loss of 47.54%.

Discussion

- COBYLA consistently outperformed the other optimizers across all metrics, achieving the highest accuracy (92%), precision (89%), and F1 score (93%). Its recall rate of 97%, tied with ADAM's, indicates a good ability to minimize false negatives. Additionally, achieving the minimum loss in just 1 min of training, COBYLA is the most computationally efficient optimizer, solidifying its position as the optimal choice for this model.
- L-BFGS-B demonstrated moderate performance, with accuracy (89%) and recall (94%) values slightly lower than COBYLA's. While it achieved a precision of 86%, its overall F1 score (90%) indicated balanced but less effective performance compared to COBYLA. Its training time of 6 min was longer than COBYLA's, though still reasonable. Additionally, the loss for L-BFGS-B was 11.48%, indicating a less effective optimization compared to COBYLA's loss but significantly better than ADAM's loss.
- ADAM's performance was notably weaker, particularly in terms of accuracy (52%) and precision (53%). Despite achieving a high recall rate of 97%, comparable to COBYLA's, its poor precision resulted in an imbalanced model, as reflected in the F1 score of 68%. This pattern, characterized by high recall at the expense of precision,

often indicates a strong bias towards predicting the positive class, where the model effectively captures most actual positives but incurs a significant number of false positives, ultimately leading to suboptimal overall accuracy. Furthermore, ADAM exhibited the longest training time of 10 min among the three optimizers, rendering it the least efficient choice in this context. Adding to its poor performance, ADAM demonstrated a significantly higher loss of 47.54%, indicating a substantial discrepancy between predicted and actual outcomes compared to both COBYLA's 8.20% and L-BFGS-B's 11.48% loss.

Although all optimizers achieved convergence (100%), the significant disparity in their final loss underscores the critical impact of optimizer selection in quantum machine learning models, necessitating choices tailored to this domain.

Consistent with findings in prior work by Pellow-Jarman et al. [47], which demonstrated COBYLA's strong performance and ability to achieve low cost values in noise-free simulations, COBYLA's observed efficiency in our study reinforces its potential as a benchmark for similar quantum learning tasks.

6. Conclusions and Future Work

The primary objective of this study was to systematically compare the performance of three prominent optimization techniques—COBYLA, L-BFGS-B, and ADAM—in the context of a quantum machine learning model. Through a rigorous evaluation using key performance metrics such as accuracy, precision, recall, and F1 score, this analysis provides valuable insights into the strengths and limitations of each optimizer for classification tasks.

The results unequivocally demonstrate that COBYLA outperformed the other optimizers in this specific application. It achieved the highest accuracy (92%), precision (89%), recall (97%), and F1 score (93%) while requiring only 1 min of training time. This combination of superior performance and computational efficiency makes COBYLA the optimal choice for scenarios where both accuracy and speed are critical. In contrast, L-BFGS-B and ADAM demonstrated low performances compared to COBYLA. In summary, the findings indicate that COBYLA is the most suitable optimizer for enhancing model performance on the Cleveland dataset, particularly when constrained by time. L-BFGS-B offered moderate performance within a 6 min timeframe, while ADAM proved to be the least effective, especially in terms of its accuracy, precision, and F1 score, despite its longer training duration.

Building on these findings, future research should explore the performance of these optimizers across a broader range of datasets. By testing their efficacy on diverse data types and complexities, we aim to deepen our understanding of how these optimization techniques generalize to different problem domains. This expanded analysis will provide a more comprehensive framework for selecting optimizers tailored to specific applications in quantum machine learning, further advancing the field's capabilities. To extend this study, we propose to investigate the impact of noise and decoherence on optimizer performance in NISQ devices, as well as exploring hybrid optimization strategies that combine the strengths of multiple algorithms.

Author Contributions: Conceptualization, M.S.; Methodology, N.A.A.A.; Validation, M.S.; Formal Analysis, N.A.A.A.; Writing—Original Draft, N.A.A.A.; Writing—Review and Editing, M.S.; Supervision, M.S.; Funding Acquisition, M.S. All authors have read and agreed to the published version of the manuscript.

Funding: The authors extend their appreciation to the Deanship of Scientific Research at King Saud University for funding and supporting this research through its Graduate Students Research Support (GSR).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are openly available in the Cleveland Heart Disease dataset from the UCI repository (https://archive.ics.uci.edu/dataset/45/heart+disease), reference number [29] (accessed on 4 November 2023).

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of the data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Marella, S.T.; Parisa, H.S.K. Introduction to Quantum Computing. In *Quantum Computing and Communications*; IntechOpen: London, UK, 2020; p. 61.
2. Herman, D.; Googin, C.; Liu, X.; Galda, A.; Safro, I.; Sun, Y.; Pistoia, M. A survey of quantum computing for finance. *arXiv* **2022**, arXiv:2201.02773.
3. Ding, C.; Bao, T.-Y.; Huang, H.-L. Quantum-Inspired Support Vector Machine. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 7210–7222. [CrossRef] [PubMed]
4. Deutsch, D.; Jozsa, R. Rapid solution of problems by quantum computation. *Proc. R. Soc. London. Ser. A Math. Phys. Sci.* **1992**, *439*, 553–558.
5. Shor, P.W. Algorithm for Quantum Computation: Discrete Logarithms and Factoring. In Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, 20–22 November 1994; Volume 35, pp. 124–134.
6. Abhijith, J.; Adedoyin, A.; Ambrosiano, J.; Anisimov, P.; Casper, W.; Chennupati, G.; Coffrin, C.; Djidjev, H.; Gunter, D.; Karra, S.; et al. Quantum Algorithm Implementations for Beginners. *ACM Trans. Quantum Comput.* **2022**, *3*, 1–92.
7. Khan, T.M.; Robles-Kelly, A. Machine Learning: Quantum Vs Classical. *IEEE Access* **2020**, *8*, 219275–219294. [CrossRef]
8. Huang, H.Y.; Broughton, M.; Mohseni, M.; Babbush, R.; Boixo, S.; Neven, H.; McClean, J.R. Power of data in quantum machine learning. *Nat. Commun.* **2021**, *12*, 2631. [CrossRef] [PubMed]
9. Schuld, M.; Killoran, N. Quantum machine learning in feature Hilbert spaces. *Phys. Rev. Lett.* **2019**, *122*, 040504. [CrossRef]
10. Chakraborty, S.; Das, T.; Sutradhar, S.; Das, M.; Deb, S. An Analytical Review of Quantum Neural Network Models and Relevant Research. In Proceedings of the 5th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 10–12 June 2020.
11. Cerezo, M.; Arrasmith, A.; Babbush, R.; Benjamin, S.C.; Endo, S.; Fujii, K.; McClean, J.R.; Mitarai, K.; Yuan, X.; Cincio, L.; et al. Variational quantum algorithms. *Nat. Rev. Phys.* **2021**, *3*, 625–644. [CrossRef]
12. Qin, J. Review of Ansatz Designing Techniques for Variational Quantum Algorithms. *J. Phys. Conf. Ser.* **2023**, *2634*, 012043. [CrossRef]
13. Lubasch, M.; Joo, J.; Moinier, P.; Kiffner, M. Variational quantum algorithms for nonlinear problems. *Phys. Rev.* **2020**, *101*, 010301. [CrossRef]
14. Kouda, N.; Matsui, N.; Nishimura, H.; Peper, F. Qubit neural network and its learning efficiency. *Neural Comput. Appl.* **2005**, *14*, 114–121. [CrossRef]
15. Jeswal, S.K.; Chakraverty, S. Recent Developments and Applications in Quantum Neural Network: A Review. *Arch. Comput. Methods Eng.* **2019**, *26*, 793–807. [CrossRef]
16. Chalumuria, A.; Kuneb, R.; Manoja, B.S. Training an Artificial Neural Network Using Qubits as Artificial Neurons: A Quantum Computing Approach. *Procedia Comput. Sci.* **2020**, *171*, 568–575. [CrossRef]
17. Garcíaa, D.P.; Cruz-Benito, J. Systematic Literature Review: Quantum Machine Learning and its applications. *Comput. Sci. Rev.* **2022**, *51*, 100619. [CrossRef]
18. Nielsen, M.A.; Chuang, I.L. *Quantum Computation and Quantum Information*, 1st ed.; Massachusetts Institute of Technology: Cambridge, MA, USA, 2004.
19. Ezhov, A.A.; Ventura, D. Quantum neural Networks. In *Future Directions for Intelligent Systems and Information Sciences*; Springer: Berlin/Heidelberg, Germany, 2000; pp. 213–235.
20. Khan, W.R.; Kamran, M.A.; Khan, M.U.; Ibrahim, M.M.; Kim, K.S.; Ali, M.U. Diabetes Prediction Using an Optimized Variational Quantum Classifier. *Int. J. Intell. Syst.* **2025**, *2025*, 1351522. [CrossRef]
21. Yi, Z. Evaluation and Implementation of Convolutional Neural Networks. In Proceedings of the 1st International Conference on Advanced Algorithms and Control Engineering, Pingtung, Taiwan, 10–12 August 2018.
22. Zhu, D.; Linke, N.M.; Benedetti, M.; Landsman, K.A.; Nguyen, N.H.; Alderete, C.H.; Perdomo-Ortiz, A.; Korda, N.; Garfoot, A. Training of quantum circuits on a hybrid quantum computer. *Sci. Adv.* **2019**, *5*, eaaw9918. [CrossRef]
23. Ajibosin, S.S.; Cetinkaya, D. Implementation and Performance Evaluation of Quantum Machine Learning Algorithms for Binary Classification. *Software* **2024**, *3*, 498–513. [CrossRef]

24. Al-Zafar Khan, M.; Al-Karaki, J.; Omar, M. Predicting Water Quality using Quantum Machine Learning: The Case of the Umgeni Catchment (U20A) Study Region. *arXiv* **2024**, arXiv:2411.18141.
25. Aminpour, S.; Banad, Y.; Sharif, S. Strategic Data Re-Uploads: A Pathway to Improved Quantum Classification Data Re-Uploading Strategies for Improved Quantum Classifier Performance. *arXiv* **2024**, arXiv:2405.09377.
26. Preskill, J. Quantum computing in the NISQ era and beyond. *Quantum* **2018**, *2*, 79. [CrossRef]
27. Chaudhury, S.; Yamasaki, T. Robustness of Adaptive Neural Network Optimization Under Training Noise. *IEEE Access* **2021**, *9*, 37039–37053. [CrossRef]
28. Abohashima, Z.; Elhoseny, M.; Houssein, E.H.; Mohamed, M. Classification with Quantum Machine Learning: A survey. *arXiv* **2020**, arXiv:2006.12270.
29. UC Irvine Machine Learning Repositroy. Heart Disease. 1988. Available online: https://archive.ics.uci.edu/dataset/45/heart+disease (accessed on 4 November 2023).
30. Garate-Escamila, A.K.; ELHassani, A. Classification models for heart disease prediction using feature selection and PCA. *Inform. Med. Unlocked* **2020**, *19*, 100330. [CrossRef]
31. Shrestha, D. Advanced Machine Learning Techniques for Predicting Heart Disease: A Comparative Analysis Using the Cleveland Heart Disease Dataset. *Appl. Med. Inform.* **2024**, *46*, 91–102.
32. Han, J.; Kamber, M.; Pei, J. *Data Mining: Concepts and Technique, The Morgan Kaufmann Series in Data Management Systems*, 3rd ed.; Elsevier: Amsterdam, The Netherlands, 2012.
33. Medium. Building a Quantum Variational Classifier Using Real-World Data. Qiskit. 2021. Available online: https://medium.com/qiskit/building-a-quantum-variational-classifier-using-real-world-data-809c59eb17c2 (accessed on 2 February 2024).
34. IBM Quantum Documentation. RealAmplitudes. Available online: https://docs.quantum.ibm.com/api/qiskit/0.30/qiskit.circuit.library.RealAmplitudes (accessed on 1 February 2024).
35. Mangini, S. Variational Quantum Algorithms for Machine Learning Theory and Applications. Ph.D. Thesis, University of Pavia, Pavia, Italy, 2023.
36. Terven, J.; Cordova-Esparza, D.M.; Ramirez-Pedraza, A.; Chavez-Urbiola, E.A.; Romero-Gonzalez, J.A. Loss Functions and Metrics in Deep Learning. *arXiv* **2023**, arXiv:2307.02694.
37. Desai, C. Comparative Analysis of Optimizers in Deep Neural Networks. *Int. J. Innov. Sci. Res. Technol.* **2020**, *5*, 959–962.
38. Vishwakarma, N. What Is Adam Optimizer? 2024. Available online: https://www.analyticsvidhya.com/blog/2023/09/what-is-adam-optimizer/#:~:text=The%20Adam%20optimizer,%20short%20for,Stochastic%20Gradient%20Descent%20with%20momentum (accessed on 8 January 2024).
39. Zhu, C.; Byrd, R.H.; Lu, P.; Nocedal, J. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw. (TOMS)* **1997**, *23*, 550–560. [CrossRef]
40. Zamyshlyaeva, A.A.; Bychkov, E.V.; Kashcheeva, A.D. Algorithm for Numerical Solution of the Optimal Control Problem for One Hydrodynamics Model Using the COBYLA Method. *J. Comput. Eng. Math.* **2024**, *11*, 40–47.
41. Powers, T.; Rajapakse, R.M. Using Variational Eigensolvers on Low-End Hardware to Find the Ground State Energy of Simple Molecules. Quantum Physics. *arXiv* **2023**, arXiv:2310.19104.
42. Wendorff, A.; Botero, E.; Alonso, J.J. Comparing Different Off-the-Shelf Optimizers' Performance in Conceptual Aircraft Design. In Proceedings of the 17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Washington, DC, USA, 13–17 June 2016.
43. IBM Quantum Documentation. L_BFGS_B. Available online: https://docs.quantum.ibm.com/api/qiskit/0.40/qiskit.algorithms.optimizers.L_BFGS_B (accessed on 1 December 2024).
44. Su, Y.; Song, K.; Yu, K.; Hu, Z.; Jin, H. Quantitative Study of Predicting the Effect of the Initial Gap on Mechanical Behavior in Resistance Spot Welding Based on L-BFGS-B. *Materials* **2024**, *17*, 4746. [CrossRef]
45. Brownlee, J. Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. Deep Learning Performance. 2021. Available online: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ (accessed on 26 November 2024).
46. Ruder, S. An overview of gradient descent optimization. *arXiv* **2017**, arXiv:1609.04747.
47. Pellow-Jarman, A.; Sinayskiy, I.; Pillay, A.; Petruccione, F. A comparison of various classical optimizers for a variational quantum linear solver. *Quantum Inf. Process.* **2021**, *20*, 202. [CrossRef]