Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Analyzing the Expressibility and Trainability of Quantum Convolutional Neural Networks

Asif Akhtab Ronggon
Bangladesh University of Engineering and Technology
Tuhin Hossain
Jahangirnagar University
Dr Tahani Jaser Alahmadi
Princess Nourah bint Abdulrahman University
Mohammad Ali Moni
m.moni@uq.edu.au

University of Queensland

Research Article

Keywords: Convolutional Neural Network(CNN), Quantum Convolutional Neural Network(QCNN), Expressibility, Trainability, Optimization, Noise

Posted Date: May 25th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8600748/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Journal Name

Crossmark

RECEIVED
dd Month yyyy

REVISED
dd Month yyyy

ARTICLE TYPE

Analyzing the Expressibility and Trainability of Quantum Convolutional Neural Networks

Asif Akhtab Ronggon[1], Tuhin Hossain[2], Tahani Jaser Alahmadi[3], Mohammad Ali Moni[4]

[1]Department of Electrical and Electronics Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh Email: asifaftab172@gmail.com

[2]Department of Computer Science and Engineering, Jahangirnagar University, Savar, Bangladesh. Email: tuhin97.hossain@gmail.com

[3] Department of Information Systems, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh, Saudi Arabia. Email: tjalahmadi@pnu.edu.sa

[4] AI & Digital Health Technology, Rural Health Research Institute, Charles Sturt University, Orange, NSW 2800, Australia

School of Health and Rehabilitation Sciences, The University of Queensland, St Lucia, Brisbane, QLD 4072, Australia

E-mail: m.moni@uq.edu.au

Keywords: Convolutional Neural Network(CNN), Quantum Convolutional Neural Network(QCNN), Expressibility, Trainability, Optimization, Noise

Abstract

Quantum Convolutional Neural Networks(QCNN) are a promising framework in Quantum Machine Learning(QML). The study analyzes QCNNs' expressibility, trainability, and optimizers theoretically. This approach utilizes *statistical learning* and *quantum information theory* to assess QCNNs' effectiveness in imitating quantum states, optimizing parameters, and executing intricate machine learning(ML) tasks. For this purpose, the *Haar measure* is used to evaluate the QCNN expressibility, yielding a score of **0.7720**, a *coverage entropy* of **1.4023**, and an effective state-space dimension of **166.59/200**. This indicates that QCNNs can simulate massive quantum states and achieve a peak expressibility score of **0.8120** at layer 3, before decreasing due to overparameterization. Additionally, we examine *gradient variance* and find that the *barren plateau* indicator is **0.0818** and the *mean gradient norm* is **1.1837**, both of which are below the theoretical threshold of **0.1**, proving that the QCNN can avoid *barren plateaus* and train perfectly. Furthermore, the obtained *Fisher Information Matrix(FIM)* condition number of $\mathbf{2.05 \times 10^8}$ indicates a well-conditioned optimization landscape. We also utilized the *Hessian matrix* for optimization analysis, identifying a *saddle point* with **56 zero**, **11 positive**, and **5 negative eigenvalues**. Where QCNNs required only **72 parameters**, an **11.8-fold** reduction from classical CNNs' **848 parameters**, but expressibility stays the same, assuming smooth convergence. Despite a *gradient noise* factor near 1, QCNNs have only a **43%** expressibility decrease at **5%** noise levels. Finally, the outcomes prove that QCNNs outperform classical CNNs in ML applications with respect to expressibility, trainability, and optimization.

1 Introduction

Machine learning (ML) requires more computational resources due to the ever-growing amount of data in the domain of big data. Convolutional neural networks (CNNs) have revolutionized ML by facilitating effective feature extraction from structured data, including images and time-series signals, through the utilization of localized convolutional kernels and pooling mechanisms [1]. These designs have achieved significant success in fields such as computer vision (CV) [2] and natural language processing (NLP) [3]. However, as datasets and models become increasingly complex, classical CNNs face a fundamental challenge due to the limitations of traditional computer architectures. However, theoretical assessments suggest that QC could accelerate the process of addressing issues that conventional systems have not been able to solve currently [4, 5, 6]. At the same time, QC demonstrated promising results in vastly improving ML via several

methods, including enhanced privacy protections, reduced training parameter requirements, and improved memory efficiency[7, 8, 9]. For particular tasks, QC provides a solution that leverages entanglement, interference, and superposition, which may result in computation benefits that are exponential [10]. Also, by integrating these QC concepts into ML frameworks, QML enhances the performance of models [11]. This area applies the CNN model to quantum systems through QCNNs, where quantum coherence is preserved through pooling and convolutional layers are constructed for qubits using parameterized unitary operations [12]. Recently, QNNs have become more popular as a method for variational quantum algorithms (VQAs), among its many other significant uses[13, 14, 15, 16]. Additionally, these structures have been applied to quantum chemistry [17], financial modeling [18], and quantum sensing [19, 20]. For optimal cost functions in high-dimensional *Hilbert spaces*, parameterized quantum circuits (PQCs), such as QNNs, can be utilized. Their ability to tackle variational problems is directly related to how efficiently they describe complex functions.

Researchers are still very much focused on finding ways to make QNNs even more efficient and scalable. To make QNN more expressive and useful in various contexts, several types of PQC have been proposed. These network types include quantum generative adversarial networks (QGANs) [21], recurrent QNNs [22, 23], convolutional QNNs [12, 24, 25], and QNNs enhanced with randomness [26]. It also showed promise in tasks including processing classical data contained in quantum states and classifying quantum phases [8]. However, their use depends on two factors: how clearly they can be represented and how easily they can be trained. The trainability of parameter optimization in VQA and the expressibility of parameterized quantum circuits both reflect how well these algorithms may simulate specific unitary processes [14, 27].

A significant challenge in training Quantum Neural Networks (QNN) is the *barren plateau* phenomenon, characterized by an exponential decline in *gradient variances* with increasing system size, leading to flat optimization landscapes that impede effective training [28]. Recent initiatives to tackle this issue include the development of specific techniques, such as branching designs that utilize parallel circuit paths to enhance entanglement dispersion and expressibility [29]. Further, by integrating classical pretreatment and postprocessing, hybrid quantum-classical models attempt to decrease *plateaus* and increase resource efficiency [30, 31]. Recent developments include deep convolutional networks that utilize adaptive layering or randomization to enhance their trainability, as well as degressive QCNNs that reduce parameter sharing to make kernels more flexible [32, 33]. By statistically evaluating expressibility using metrics such as entanglement entropy and unitary design coverage, theoretical investigations have shed light on the representational capacities of quantum circuits [34].

Despite these advancements, a thorough mathematical analysis comparing the trainability and expressibility of QCNNs with traditional CNNs remains inadequate. This assertion is especially pertinent in the context of establishing quantitative bounds that elucidate quantum advantages in terms of convergence rates, generalization capabilities, and performance metrics on correlated datasets [35, 36]. This study aims to fulfill this need by creating a strong mathematical framework for investigating QCNNs. Our analytical bounds on expressibility are derived from unitary design and entanglement entropy metrics, and our trainability is evaluated using *gradient variance* and optimization landscape analysis. The findings reveal that QCNNs possess superior trainability and expressibility compared to classical CNNs, providing a theoretical basis for their enhanced performance, especially on datasets exhibiting quantum-like correlations.

This paper's primary contribution is a comprehensive computational assessment of QCNN. We explore the expressibility of QCNNs and demonstrate that they are more capable of modeling a broad range of quantum states. We also rigorously analyze QCNN trainability using *gradient variance* and *barren plateau* markers to gain a better understanding of its learning mechanism. As a last step, we investigate the QCNN optimization landscape, which reveals superior information capacity, smooth convergence, and a dramatic reduction in parameter requirements compared to classical CNNs.

The following is the outline of the paper: Section 2 includes a discussion of relevant literature. Classical CNNs and QCNNs are described in Section 3 for the purposes of expressibility, trainability, and optimization analysis. The findings and comparisons of numerical experiments are presented in Section 4. Finally, Section 5 provides an overview of the consequences and future research objectives.

2 Related Work

Significant research has already been conducted on QCNNs and their potential applications in QML. This research has opened up new possibilities for future advancements in this field. Rahim et

al. [37] developed a QCNN model for image classification. It introduces a probabilistic amplitude-based quantum encoding method that could potentially transform classical data into quantum states, thereby reducing dimensionality and facilitating the transition between the two data processing styles. Here, the convolutional layer of a QCNN employs rotating gates, while the pooling layer makes use of controlled gates. Researchers found that the QCNN outperformed traditional models in quantum image classification when tested on the MNIST handwritten digit dataset. To enhance image categorization, Cheng et al. [38] presented a Hybrid Quantum-Classical Convolutional Neural Network (HQCCNN). It combines classical CNNs with QC. By utilizing parameterized quantum circuits (PQCs) in the fully connected and convolutional layers, which leverage quantum parallelism and entanglement, the model enhances feature extraction from image data. When tested on the MNIST dataset, HQCCNN outperformed classical CNNs in terms of convergence speed and classification accuracy, achieving 99.71% accuracy for binary classification (digits 5 and 7) and 98.51% accuracy for multivariate classification. Long et al. [39] proposed a Hybrid Quantum-Classical-Quantum Convolutional Neural Network (QCQ-CNN) model that integrates quantum and classical elements to enhance image classification. where a trainable QNN classifier is used for decision-making, a conventional CNN module is used for spatial learning, and a fixed quantum convolutional filter is used for feature extraction. As a result of the QNN classifier's ability to optimize quantum parameters, training overhead is decreased by the quantum filter. Assessments conducted on the MNIST, Fashion-MNIST, and MRI brain tumor datasets demonstrated that QCQ-CNN surpasses traditional CNNs and hybrid architectures, such as QCCNN, attaining an accuracy of 98.85% on MNIST. Their model demonstrated resilience to finite sampling shots and quantum noise, with five-layer, moderate-depth quantum circuits offering the optimal balance of expressivity and stability. In another study, Wen et al. [40] presented a new quantum residual neural network, called QResNet, which enhances expressivity by utilizing quantum circuits with residual connections. A wider variety of frequency components in the Fourier spectrum can be generated by the model by adding supplementary qubits to both data-encoding and trainable blocks. Using residual encoding, the number of feasible frequency combinations increases from one set in conventional techniques to $O(l^2)$ for l-layer encoding, considerably improving spectrum richness and making Fourier coefficient manipulation more flexible. Experimental data show that residual encoding dramatically reduces the mean squared error (MSE) of the regression task to 5.1×10^{-5} compared to conventional methods. Residual encoding enhances image classification (MNIST) accuracy by 7.7%, reaching 92.35% in testing and 92.85% in training.

A service-oriented method called Q-TEE was introduced by López et al. [41] to assess the expressivity and trainability of Variational Quantum Circuits (VQCs). The tool measures VQC efficiency on optimization problems, such as the MaxCut issue, by examining 5-, 10-, and 15-node circuits. The expressiveness (the ability to represent a large range of quantum states) and trainability (the simplicity of optimization) of these ansätze are evaluated in this work by looking at their expectation value distributions. On the 5-qubit graph, all of the ansatzes displayed similar expressivity; however, the single-layer RY circuit exhibited a far wider dispersion, indicating superior expressivity but inferior trainability. As the number of qubits in the graph increased to 10 or 15, the differences in expressivity became more noticeable. The fully entangled ansatz was not very expressive, in contrast to Real Amplitudes and Efficient SU2. With a particular focus on a driven quantum Ising chain, Tangpanitanon et al. [42] and colleagues conducted research on the expressibility and trainability of parameterized analog quantum systems. It is demonstrated that the system can achieve quantum supremacy in all phases, including thermalized and many-body localized (MBL) states, which is a notable strength. To make generative modeling problems more trainable, the authors propose a sequential training method. The results show that thermalized phases are distracting because of their chaotic dynamics, but MBL phases make the system more stable, which improves trainability. By incorporating an external drive, the system's learning accuracy is enhanced, and the cost function is continually reduced, indicating excellent optimization for QML tasks. Monbroussou et al. [43] analyzed the expressivity and trainability of VQCs, which are quantum circuits that preserve Hamming weight. It presented a quantum data loader that used RBS and FBS gates to encode amplitudes into subspaces with fixed Hamming weights. To determine the controllability of these circuits, the authors utilized the QFIM rank. They showed that the gradient variance is limited under specific conditions, which allows them to avoid barren plateaus. This study sheds light on the effects of qubit connections on circuit design and provides theoretical foundations for enhancing the effectiveness and trainability of QML models.

A hybrid network named Res-HQCNN, which integrates quantum and conventional neural architectures, was proposed by Liang et al. [44] to address graph-structured quantum data. This model acquired sophisticated quantum knowledge rapidly by integrating the expressiveness of

QNNs and the stability of residual learning. With a focus on the impact of various network depths and residual block topologies, they conducted a series of investigations to evaluate the performance of Res-HQCNN on graph-structured quantum data. As a consequence of its residual connections, Res-HQCNN far surpasses a conventional universal QNN in terms of performance when dealing with graph-structured data. By integrating graph information into the learning process, Res-HQCNN consistently surpasses the baseline model in both testing and training phases simultaneously. In an assessment of competing models utilizing line-structured data and a single hidden layer, Res-HQCNN demonstrated superior performance, especially with the incorporation of graph information ($\gamma = -0.5$). This model outperformed alternatives that did not incorporate graph information in terms of convergence speed and fidelity. Impressive performance was demonstrated by Res-HQCNN before the addition of two hidden layers. These layers enhanced stability, generalization, and network depth, especially for line-structured input. In Quantum Circuit Learning (QCL), using a hardware-efficient ansatz (HEA), Chen et al. [45] looked at the connection between expressibility and overfitting. A potential mechanism to prevent overfitting might be an automatic regularization procedure, as the saturation of expressibility and generalization errors increases with the depth of quantum circuits. Unitarity and localization in the ansatz restrict the model's expressibility and complexity, as demonstrated by the study's use of PAC learnability and VC dimension analysis. These findings provide support for the theory that quantum circuits possess an intrinsic regularization mechanism, which is crucial for overcoming hardware limitations and improving QML performance. Additionally, Liao et al. [46] investigated techniques designed to enhance the expressive power of QNNs, which utilize quantum circuits for machine learning applications. The study recommends using sinusoidal function-friendly embedding for precise characterization of sinusoidal functions. Redundant measurements are employed to enhance expressivity while maintaining a constant consumption of quantum resources. The set of basis functions is then enlarged using a post-measurement function. Overfitting can be avoided by using randomly selected training data. By implementing these tactics, QNNs can enhance their approximation of complex functions, particularly sinusoidal ones, while simultaneously decreasing the resource needs for more quantum resources. Quantum circuit learning (QCL) and other conventional methods were found to be inferior in performance compared to these newer approaches.

The expressibility and entangling capability of parameterized quantum circuits (PQCs) were studied by Mjaaseth et al. [47] as a methodological issue. To create three descriptions for these properties, the author employed statistical methods to select states from a class of quantum circuits. Their use of a Python framework for QML facilitates the construction, training, and evaluation of PQCs. Training 10,000 circuits at random from the Iris dataset is used for classification tasks in the study. After that, they check the model's real performance against the descriptor values using a correlation analysis. The findings show that circuits with specific properties, such as expressibility and entangling capabilities, outperform others in categorization tasks in terms of task accuracy. Particularly focusing on barren plateaus (BP), Kashif et al. [48] studied how data encoding, ansatz expressibility, and qubit entanglement affected the trainability of HQNNs. The work shown through an empirical analytic method that backpropagation, marked by diminishing gradients and suboptimal optimization, is intensified by entanglement in quantum layers and increased circuit depth expressibility. The authors conclude that entanglement differentially affects performance in multi-class classification tasks based on their examination of amplitude and angle encoding systems. The findings suggest that quantum layers with reduced widths and moderate depths are most effective in mitigating BP and enhancing training efficiency. This highlights the compromise between the two variables. Wu et al. [32] explored at how expressive QNNs are, focusing on how well they can mimic physical and general observables such the Rényi entropy, the Loschmidt echo, and magnetization. In situations where the input wave functions occupy less than 50% of the total Hilbert space, their findings indicate that accurate estimations are possible. To circumvent this limitation, they propose methods to expand the Hilbert space, such as increasing the size of the input wave functions or introducing an additional qubit. They found that by including an auxiliary qubit into QNNs, their expressivity is much increased, leading to more accurate learning of target functions and a smaller loss function.

Chinzei et al. [49] investigated how QNNs strike a balance between expressivity and gradient measurement efficiency. The work demonstrates that more expressive QNNs require higher costs for gradient estimation measurements. They also introduced the Stabilizer-Logical Product Ansatz (SLPA), a model for efficient gradient measurement based on symmetrical quantum circuit topology. While measuring gradients, the SLPA is most efficient when the Feff value is 4. It also shortens training time and simplifies data processing when compared to more traditional methods,

such as the parameter-shift method. By perfectly maximizing the Dynamical Lie Algebra (DLA) dimension, the SLPA ensures excellent expressivity and quick gradient estimation. This model offers an efficient method for training QML, especially for problems involving symmetry. By emphasizing regression tasks, Panadero et al. [50] explored how data encoding methodologies, network depth, and qubit entanglement affected the overall performance of Quantum Neural Networks (QNNs). Researchers found that the data encoding approach limits the maximal expressivity when they analyzed the network's output using Fourier analysis to assess the expressivity. The research also shows that entangling gates, especially at the last layer, significantly enhance the QNN's approximation capabilities when compared to local measurements of individual qubits, which is particularly noteworthy in light of previous results. In their evaluation of the network's approximation skills for continuous functions, the authors employ a teacher-student benchmarking method. They conclude that entanglement-enabled QNNs outperform non-entangling networks in terms of regression accuracy. Aktar et al. [51] introduced a Graph Neural Network (GNN)- based approach to forecasting the expressibility of parameterized quantum circuits (PQCs), a metric for their ability to capture different quantum states. In the GNN paradigm, the PQC is conceptualized as a graph, where inputs, gates, and outputs are represented by nodes, and the flow of information is represented by edges. The model incorporates global variables like circuit depth and qubit count using its training on 12,000 randomly selected PQCs. To extract local node data, it uses SAGEConv layers. With a root-mean-squared error (RMSE) of 0.03 on random PQCs and 0.06 on IBM Qiskit's RealAmplitude circuits, the model minimizes the computing cost of expressibility estimation. Shao et al. [52] developed the 2MC-OBPPP framework, a conventional approach for evaluating noise resilience, trainability, and expressibility within parameterized quantum circuits (PQCs). It minimizes overhead by checking noisy circuits with polynomial-time estimators, identifying "noise bottlenecks" to remove, and then removing them. The study found that small-amplitude damping noise can temporarily improve barren plateaus by increasing gradient variance and enhancing training effectiveness; however, it reduces expressibility. This paradigm facilitates optimizing quantum circuits before deployment, which also aids in developing more efficient quantum algorithms.

By studying QNN expressivity with SWAP test circuits, Nagies et al. [53] revealed that quadratic activation functions had trouble learning complicated functions, like the parity check, in dimensions beyond two. A product-layer generalized SWAP test circuit, proposed by the authors as a solution, can activate functions with higher degrees of polynomials. This update enables the QNN to train for parity checks in high-dimensional spaces and related tasks. The fact that it achieved an 84% accuracy on a three-dimensional parity check using real quantum hardware, even with noise, demonstrates its potential for use in both quantum and classical learning. The updated architecture shows better performance on both synthetic and real-world datasets (IRIS, MNIST). Using frame potentials and the Kullback-Leibler (KL) divergence of fidelity distributions, Maronese et al. [54] measured the expressibility of QNN designs and found the proximity they are to the Haar distribution. The numerical results demonstrate that for systems up to n=10 qubits, Clifford-enhanced matrix product states (CMPS) can attain near-Haar expressibility with $D_{\mathrm{KL}} \approx 10^{-3}$ for the bond dimension χ=1. When n = 10, fully quantum neural networks (fQNNs) require approximately 12 layers (˜360 parameters) to achieve the same level of expressibility. These results show that classically simulable architectures can achieve excellent expressibility.

3 Methodology

This section presents a comprehensive theoretical foundation for QCNN analysis, focusing on the optimum landscape, expressibility, trainability, and comparative performance relative to classical CNNs. Our methodology ensures the investigation of fundamental aspects of QML through the use of proven principles from statistical learning, optimization theory, and quantum information theory. By investigating these aspects, we aim to provide significant theoretical insights and assurances into how QCNNs perform in various QML tasks.

3.1 ***CNN architecture***

CNNs are comprised of several distinct layers. These include convolutional layers, pooling layers, and fully connected layers. By stacking these layers, a convolutional neural network (CNN) architecture is created [55].

3.1.1 ***Input layer:*** The initial step in training a CNN is feeding it raw data, typically an image, through its input layer. The input is depicted as a tensor with three dimensions, where $X \in \mathbb{R}^{H \times W \times D}$, with H denoting the height, W the width, and D the depth of the image (the

number of color channels; for example, $D = 3$ for RGB photos). In the image, each element $X_{i,j,d}$ represents the pixel intensity at location (i, j), where d denotes the color channel.

$$X = \{X_{i,j,d} \mid i \in \{1, \ldots, H\},\ j \in \{1, \ldots, W\},\ d \in \{1, \ldots, D\}\}$$

To represent a grayscale picture, $D = 1$, which means the array of pixel values is 2D. Later layers, such as convolution, activation, and pooling, process the data after the input layer, which does not execute any calculation itself. Image pixel values are usually preprocessed or standardized to fall inside a constant range, usually $[0, 1]$ or $[-1, 1]$, so that training dynamics can be enhanced. This input tensor is crucial for the network to learn from, as it allows for feature extraction and decision-making later on.

3.1.2 ***Convolutional layer:*** Convolutional neural networks rely on the convolutional layer. It uses small-sized learnable kernels (filters) that cover the entire depth of the input as one of its parameters. To create a 2D feature (activation) map, each kernel is convolved over the spatial dimensions as the input goes through the layer. The layer calculates the kernel's dot product with the local input patch at each spatial position that as mentioned in Figure 1.

$$(X * k)_{i,j} = \sum_{m=1}^{F} \sum_{n=1}^{F} X_{i+m,\, j+n}\, k_{m,n} \tag{1}$$

Here, the variable **k** represents the filter or kernel that has a size of **F**×**F**, whereas the variable **X** represents the input image matrix.

The layer's output volume is formed by stacking the activation maps that each kernel produces during training along the depth axis. Each kernel is taught to "fire" for specific features at specified places, resulting in one activation map. Three critical hyperparameters—the amount of zero-padding, the number of filters (depth), and the stride—allow convolutional layers to control their output and, in turn, reduce model complexity.

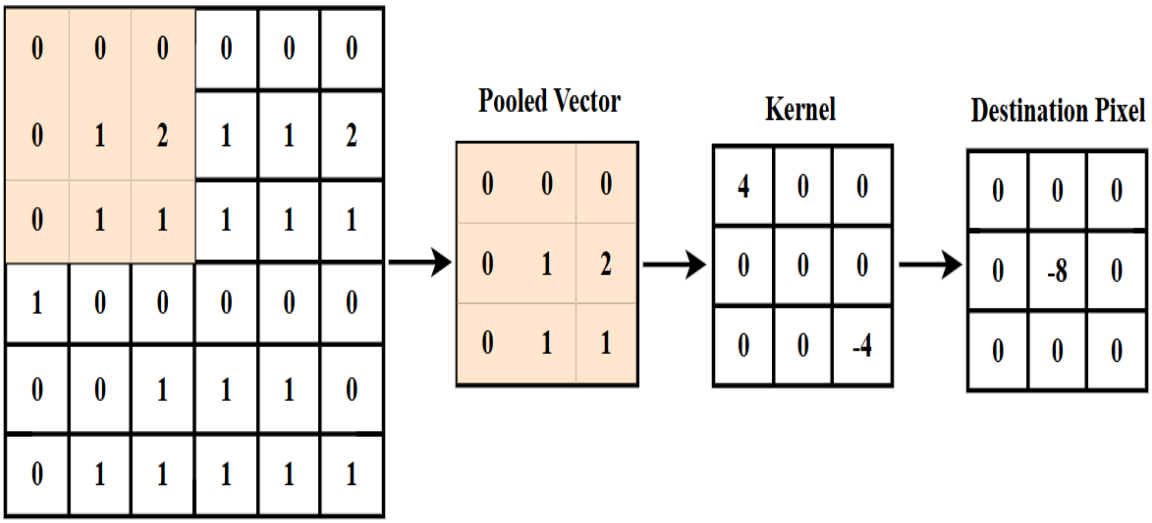


Figure 1: A visual representation of the convolutional layer operation. The kernel slides over the input feature map, and for each position, the center element aligns with the corresponding input region. The output is computed as a weighted sum of the overlapping pixels, producing the convolved feature map.

In convolutional layers, the number of neurons in the relevant input region can be manually modified to control the depth of the output volume. All of the neurons in a hidden layer are fully connected to every neuron in the layer below them, just like in other kinds of neural networks. Minimizing the network's neuron count is possible by lowering this hyperparameter, although doing so may compromise the model's pattern recognition performance. The stride, which regulates the receptive field's orientation relative to the input's spatial dimensions, can also be calibrated. When the stride is set to 1, for instance, the receptive fields overlap significantly, resulting in a substantial amount of activation. In contrast, increasing the stride reduces the overlap and results in a smaller spatial dimensional output.

By adding padding around the input's edge, zero-padding improves control over the final output volume's dimensions. It is crucial to understand that these methods will change the spatial dimensions of the output that the convolutional layers produce. The following formula can be used to calculate the final dimensions:

$$\frac{(X-k)+2p}{s+1} \tag{2}$$

In the preceding formula, **X** represents the dimensions of the input volume (height × width × depth), **k** specifies the size of the receptive field, **p** signifies the extent of zero-padding applied, and **s** indicates the stride. If the outcome of this calculation is not an integer, it indicates that the stride has been improperly configured, resulting in misalignment of the neurons across the input.

Even with optimizations, models with real-world dimensions can become quite large. However, methods to drastically decrease the convolutional layer's parameter count have been devised. One important approach is parameter sharing, which works on the premise that input properties from one area are likely to be applicable to other regions as well. The convolutional layer's parameter count is drastically reduced by using the same weights and biases for all the output activation maps. As a result, the output of each neuron during backpropagation indicates the total gradient, which is added together over all depths. Instead of changing each weight individually, this allows you to update simply one set of weights.

3.1.3 ***Pooling layer:*** In a CNN, the pooling layer downsamples the feature map's spatial dimensions (height and width), while maintaining the most critical information. By performing this technique, the computational load in the following layers is reduced, overfitting is mitigated, and translation invariance is introduced. Translation invariance means that the model can recognize patterns regardless of their location within the image. The most popular type of pooling is max pooling, which selects the maximum value from each region of the input feature map. Considering a pooling window of size **F**×**F** and a stride of **s**, the output of the pooling operation at **position = (i,j)** can be determined as follows:

$$P_{\text{out}}(i,j) = \max\left(X(i:i+F,\ j:j+F)\right) \tag{3}$$

where the output after pooling is denoted as $P_{\text{out}}(i,j)$, the region of the input feature map covered by the pooling window is represented by $X(i:i+F,\ j:j+F)$, and F denotes the size of the pooling window (e.g., 2×2 or 3×3). The stride **"s"** regulates the overlap between neighboring windows and decides how far the window shifts following each pooling operation. In addition to reducing the number of parameters and calculations, pooling enables the network to focus on the most important features identified in the convolutional layers by compressing the spatial dimensions of the feature map.

3.1.4 ***Fully-connected layer:*** The fully connected (FC) layer is a dense layer in which every neuron is connected to every other neuron in the layer above. Its usual usage is as a prediction layer after multiple convolutional and pooling layers have integrated the learned information. In mathematics, the FC layer applies a linear transformation using a weight matrix **W** and a bias vector **b** to the input vector **X** from the preceding layer. This vector is often flattened from a multi-dimensional feature map. To compute the FC layer **y**, given an input vector **X**, a weight matrix **W** of dimension **n**×**m**, and **b** representing the vector of bias, the following is carried out:

$$y = W \cdot X + b \tag{4}$$

The network's learned features or decisions are represented by each element of **y**. By applying optimization methods, such as gradient descent, during training, the weights **W** and biases **b** are learned. In order to achieve the network's goal—class probabilities in classification tasks, regression values, or other predictions—the FC layer must be present. In multiclass classification problems, the softmax function is used as an activation function to convert the final FC layer's output from real values into probabilities of the target class. The sum of all the results is 1, and each value lies between 0 and 1.

The hyperparameters of CNN are presented in Table 1, along with their respective mathematical representations.

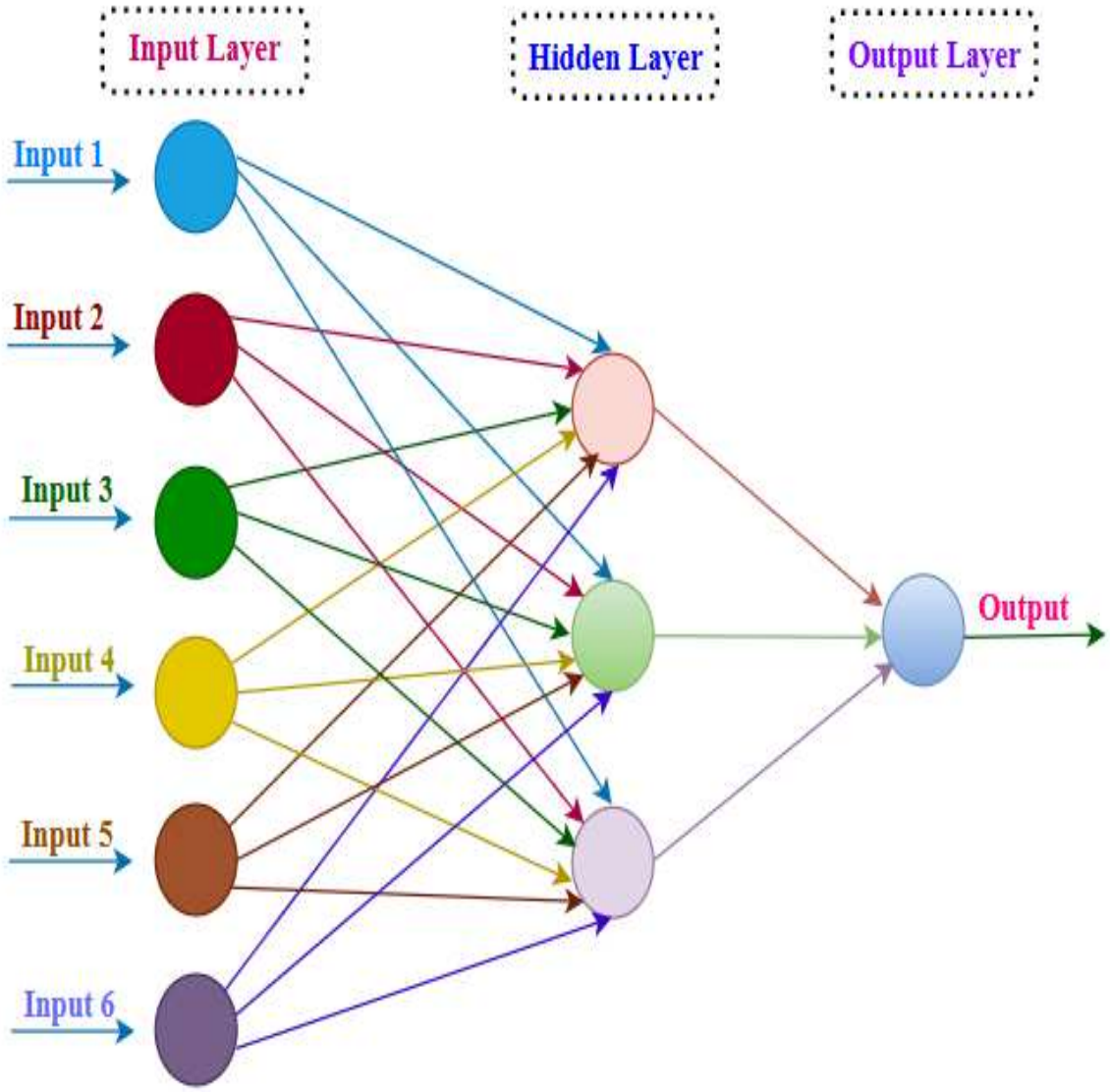


Figure 2: A fundamental illustration of CNN architecture.

Layer	**Hyperparameters**	**Mathematical representation**
Convolution layer	Input(X), Kernel size(k), number of kernels (N_k), stride(s), padding(p), activation function(σ), Output(y)	$y = \sigma(\text{Conv}(X; k, N_k, s, p))$
Pooling layer	Pooling method(pooling type), filter size(k), stride(s), padding(p)	$y = \text{Pool}(X; \text{method}, k, s, p)$
Fully connected layer	Number of weights(n_{weight}), activation function(σ)	$y = \text{FC}(X; n_{\text{weight}}, \sigma)$
Others	Model architecture(Arch), optimizer(Opt), learning rate(η), loss function(l), mini-batch size(b), epochs(e), regularization(λ), weight initialization(W_0), and dataset splitting	$\text{Train}(\text{Arch}; \text{Opt}, \eta, l, b, e, \lambda, W_0, \text{Split})$

Table 1: Overview of CNN hyperparameters with their mathematical representation.

3.2 ***QCNN Architecture and Quantum Circuit Representation***

A Quantum Convolutional Neural Network (QCNN) is an architecture that mimics classical CNN in structure but uses quantum operations instead of classical ones. In order to construct a QCNN, one must have entangling gates (e.g., CNOT), parameterized rotation gates (e.g., R_X, R_Y, R_Z), and pooling operations (which combine quantum data from several qubits). In QCNN, each layer employs quantum rotations and entangling gates to create correlations between qubits, while pooling layers, similar to classical CNN pooling procedures, lower the data's dimensionality.

The formal way to construct a QCNN is as a parameterized quantum circuit $U(\theta)$, which takes an initial quantum state $|0\rangle$ and turns it into an output quantum state $|\psi(\theta)\rangle$:

$$|\psi(\theta)\rangle = U_{\text{pooling}}\left(U_{\text{conv}}(\theta)\,|0\rangle\right) \tag{5}$$

Where:

- $\theta = (\theta_1, \theta_2, ..., \theta_p)$ stands for the parameters that regulate the entangling and rotation gates, while $U_{\text{conv}}(\theta)$ indicates the unitary operations that correspond to the convolutional layers.
- U_{pooling} depicts the layers that collect aggregated quantum data from several qubits; these layers are known as pooling.

The cumulative count of parameters p_{total} in a QCNN architecture with n_{qubits} qubits and n_{layers} layers is expressed as:

$$p_{\text{total}} = 3n_{\text{qubits}}n_{\text{layers}} + 2n_{\text{qubits}}(n_{\text{layers}} - 1) \tag{6}$$

This formula includes each qubit's rotation parameters as well as the pooling layers' entangling parameters. The overall parameter count is critical in determining the QCNN's expressive capability and ability to model complicated quantum states.

3.3 ***Expressibility Analysis***

The expressibility of a quantum circuit refers to its capacity to approximate a wide range of quantum states. Understanding the QCNN's representational capacity is crucial for evaluating its ability to generate diverse quantum states through its parameterized gates. In QML, we focus on the QCNN's ability to approach the *Haar measure*, representing the uniform distribution of quantum states.

3.3.1 ***Haar Measure and Quantum State Distribution:*** The *Hilbert space*, an intrinsic uniform measurement on the unitary group $U(2^n)$, has an impact on the *Hilbert space* of $n - qubit$ quantum states. This metric provides a consistent distribution throughout the entire quantum state space, describing the expressibility of quantum circuits. By comparing the fidelity between quantum states, we may measure how well a QCNN can approximate *Haar-random states*.

As $|\psi_1\rangle$ and $|\psi_2\rangle$ are two quantum states, the fidelity between them is explained as:

$$F(|\psi_1\rangle, |\psi_2\rangle) = |\langle\psi_1|\psi_2\rangle|^2 \tag{7}$$

In this case, equivalent states are denoted by $F = 1$ and orthogonal states are denoted by $F = 0$. The *expressibility score* measures the QCNN's proximity to the *Haar-random distribution*. More specifically, we compare the theoretical distribution expected from *Haar-random states* with the empirical distribution of fidelities derived from the QCNN's output states using the *Kolmogorov-Smirnov(KS)* statistic. The *KS* statistic can be found using:

$$D = \sup_x |F_1(x) - F_2(x)| \tag{8}$$

Where the fidelities' cumulative distribution functions are denoted as $F_1(x)$ and $F_2(x)$. The score for expressibility is:

$$\text{Expressibility Score} = 1 - D \tag{9}$$

Where a greater expressibility score suggests that the QCNN can mimic a wide range of quantum states, similar to the *Haar measure*, which is ideal for a versatile and powerful QML model.

3.4 ***Trainability Analysis***

The trainability of a QNN is defined as its efficiency when tweaked using gradient-based methods. Since the cost function's gradient grows exponentially tiny, training becomes difficult in the *barren plateau* issue, which is a major obstacle in quantum optimization. In order to evaluate the trainability of QCNNs, it is necessary to take into consideration the *gradient variance* as well as the incidence of *barren plateaus*.

3.4.1 ***Gradient Variance and Barren Plateaus:*** The training procedure is predominantly dependent on the *gradient* of the cost function $C(\theta)$ with respect to the parameters θ. The *gradient variance* has the following formula:

$$\mathrm{Var}[\nabla C] = \mathbb{E}\left[\|\nabla C\|^2\right] - |\mathbb{E}[\nabla C]|^2 \tag{10}$$

A considerable *gradient variance* indicates a region with steep gradients, whereas low gradients indicate a generally level landscape. As the number of qubits increases exponentially, the gradient variation decreases exponentially, leading to the *barren plateau* phenomena.

$$\mathrm{Var}[\nabla C] \propto \exp(-\Omega(n_{\mathrm{qubits}})) \tag{11}$$

This exponential *decay* indicates that training a QCNN becomes increasingly difficult as the number of qubits increases. We determine the *gradient norms* with parameters initialized randomly and then analyze their statistical behavior to identify empty plateaus.

3.4.2 ***Fisher Information Matrix (FIM):*** The *Fisher Information Matrix (FIM)* enhances comprehension of the optimization context by measuring the sensitivity of the cost function to parameter modifications. The following are the specifications for the *FIM's* components:

$$F_{ij} = \mathbb{E}\left[\frac{\partial C}{\partial \theta_i}\frac{\partial C}{\partial \theta_j}\right] \tag{12}$$

To evaluate the degree of conditioning of the optimization problem, a key metric is the condition number of the *FIM)*. In a landscape with inadequate conditioning, optimization may converge slowly, as seen by a high condition number. Here is the process for determining the condition number:

$$\text{Condition Number} = \frac{\lambda_{\max}}{\lambda_{\min}} \tag{13}$$

Where $\lambda_{\max}$ and $\lambda_{\min}$ refer for the largest and smallest *FIM eigenvalues*, respectively. The optimization process is made easier when the condition number is low, which means that the optimization landscape is well-conditioned.

3.5 ***Optimization Landscape Analysis***

The optimization landscape is crucial for comprehending the ease with which a QCNN can attain a minimum, since it elucidates the structure and form of the *cost function*. To evaluate the terrain, we analyze the *Hessian matrix*, which offers second-order insights on the curvature of the *cost function*.

3.5.1 ***Hessian Matrix and Eigenvalue Spectrum:*** The *Hessian matrix's* eigenvalues reveal details about the qualities of the *saddle points*, maxima, and minima that are important spots in the optimization landscape. If the landscape is well-conditioned, then positive eigenvalues in the *Hessian* will suggest local minima. The *spectral gap*, which is the difference between the smallest and largest eigenvalues of the *Hessian*, can be used to find out how far apart the minima are in the landscape. A larger *spectral gap* indicates better optimization efficiency, as it more clearly separates the minima in the optimization landscape.

The following is the definition of the Hessian matrix H:

$$H_{ij} = \frac{\partial^2 C}{\partial \theta_i \partial \theta_j} \tag{14}$$

Furthermore, we analyze the *convergence rate* of several optimization techniques, including *Adam* and *gradient descent*. To represent the convergence behavior, one usually uses an exponential *decay*:

$$C(t) = C_0 \exp(-\lambda t) \tag{15}$$

The *decay rate* is denoted by λ and the cost at iteration t is represented by $C(t\ )$. It provides significant insights into the efficacy of the optimization process.

3.6 ***Comparative Analysis with Classical CNNs***

We examine the theoretical expressibility and parameter efficacy of QCNNs relative to classical CNNs to deepen our understanding of their benefits. Classical CNNs employing *ReLU activations* are theoretically limited in the number of linear areas they may generate concerning expressibility. The significant quantity of linear regions R_{ReLU} in a traditional network with L layers and N_i neurons per layer is defined by:

$$R_{\text{ReLU}} \approx \prod_{i=1}^{L} N_i^{d_i/N_i} \tag{16}$$

Where d_i represents the dimensionality of the input layer. This bound establishes a theoretical constraint on the expressiveness of conventional CNNs. Furthermore, we assess the ratio of QCNNs to classical CNNs in terms of parameter efficiency. This ratio demonstrates the efficacy of the two architectures in describing complex characteristics through parameters. The *efficiency ratio* is defined as follows:

$$\text{Efficiency Ratio} = \frac{\text{Classical Parameters}}{\text{Quantum Parameters}} \tag{17}$$

This ratio facilitates direct comparison of the two methods in terms of the number of parameters required to achieve similar results.

3.7 ***Noise Robustness and Information-Theoretic Bounds***

The expressibility and trainability of QNN can be significantly affected by *noise*. We apply a *depolarizing noise model*, which reduces the precision of quantum states, to model the effects of *decoherence* and *gate errors*. To determine the impact of *noise* on expressibility, one can compare the precision of quantum states generated by a circuit that is free of *noise* with one that is packed with noise.

In addition, we calculate *information-theoretic bounds* in order to comprehend the basic boundaries of QCNNs. The model's ability to fit data is characterized by the *Vapnik-Chervonenkis (VC)* dimension, and *sample complexity* provides a lower bound on the number of training instances required to learn a function. In the case of quantum circuits, the VC dimension is expressed as:

$$\text{VC Dimension} = O(n_{\text{params}} \log n_{\text{params}}) \tag{18}$$

Where n_{params} denotes the quantity of parameters in the quantum circuit. We compare this to traditional sample complexity, which increases linearly with the number of qubits.

The *quantum information advantage* can also be calculated by contrasting the model's quantum capabilities with its classical capacities.

$$\text{Information Advantage} = \frac{\text{Quantum Information Capacity}}{\text{Classical Information Capacity}} \tag{19}$$

4 Results and Discussion

4.1 Computational Details

Our computations are based on the Pennylane Python software package, which is designed for quantum computing. All computations were carried out using Google Colaboratory's cloud-based computing infrastructure. We used PennyLane's built-in simulators to test the quantum circuits. Specifically, *default.qubit* was used for ideal quantum models and *default.mixed* was used for theoretical noise analysis. We also used *NetworkX* for our graph-theoretic analysis within the framework. The simulator ran on a PC with 8 GB of RAM and a Core i5 processor from the 10'th generation.

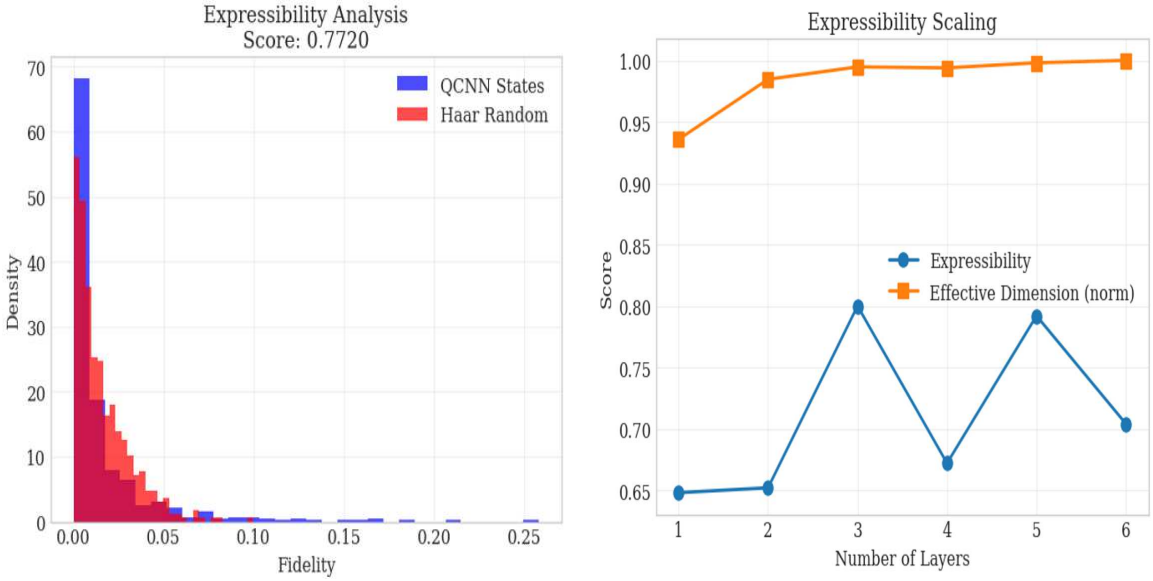
4.2 Expressibility Analysis

The expressibility of the suggested QCNN was examined using state-space coverage metrics and *Haar measure* distance, as shown in Table 2. With a *Haar expressibility score* of **0.7720**, the QCNN demonstrated a strong capacity to approximate *Haar-random quantum states*. The *coverage entropy* reached **1.4023**, which means that there were a lot of different quantum states that could be accessed. The effective state-space dimension was **166.59/200**, which meant that the *Hilbert space* was being explored in a big way.

Figure 3(a) depicts the results of the expressibility analysis, which quantifies the expressibility of the QCNN using the *Haar measure*. Figure 3(b) shows how expressibility scales as the depth of the network increases. This confirms the depth-expressibility trade-off by showing that expressibility increases with network depth up to 3-layers, peaks at *0.8120*, and then decreases slightly thereafter due to overparameterization and redundancy effects.

Table 2: Expressibility metrics of the QCNN.

Metric	**Value**	**Interpretation**
Haar Expressibility Score	0.7720	High expressibility
Effective Dimension	166.59 / 200	Broad Hilbert space coverage
Coverage Entropy	1.4023	Diverse state distribution
Optimal Depth (Layers)	3	Maximum expressibility observed



(a) (b)

Figure 3: (a) Expressibility analysis, (b) Expressibility scaling.

4.3 Trainability and Barren Plateau Analysis

Table 3 shows that the *gradient variance* analysis achieved a *mean gradient norm* of **1.1837** and a *barren plateau* indicator of **0.0818**, both of which are far lower than the theoretical threshold of **0.1**. This proves that the QCNN is highly trainable and escapes the *barren plateau* issue that frequently befalls deep quantum circuits.Also, a condition number of 2.05×10^3 was shown by the *FIM*, suggesting that the curvature in the parameter space is manageable.

Table 3: Trainability and gradient analysis results

Metric	**Value**	**Interpretation**
Mean Gradient Norm	1.1837	Efficient parameter updates
Gradient Variance	0.1146	Stable gradients
Barren Plateau Indicator	0.0818	Excellent trainability
Fisher Matrix Condition Number	2.05×10^3	Well-conditioned geometry

The QCNN's *gradient variance* and *gradient norm* are illustrated graphically in Figure 4. The illustration explicitly demonstrates that the QCNN optimization terrain is marked by strong *gradients* and the expanse of vast *barren plateaus*, ensuring smooth convergence. The results shown in this figure further demonstrate the effective training of this model, which is in line with the low indication of *barren plateaus*, in contrast to other quantum models that exhibit poor gradient propagation due to *barren plateaus*.

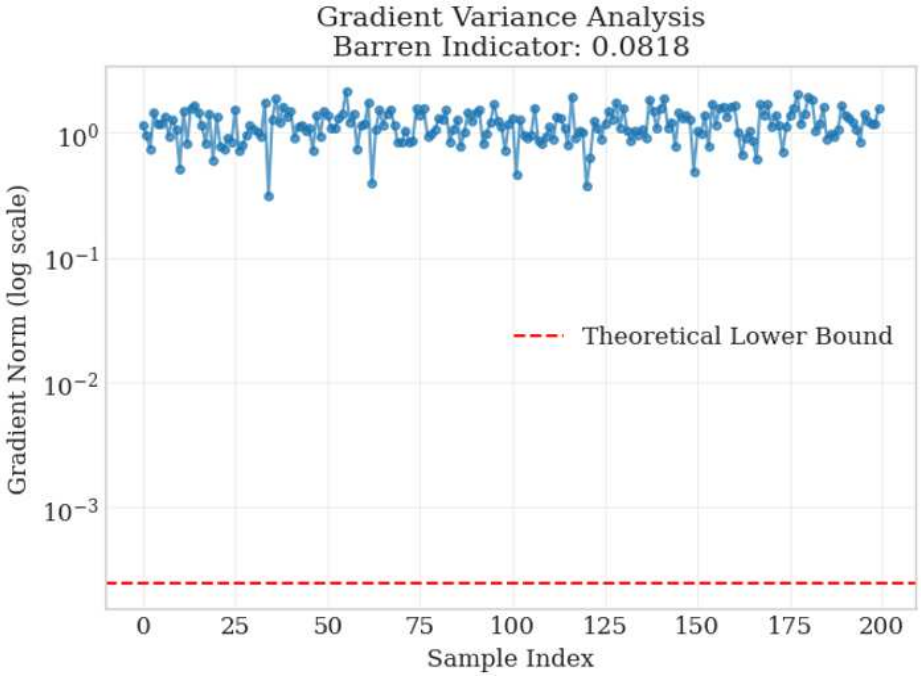


Figure 4: Gradient Variance Analysis.

4.4 Optimization Landscape Analysis

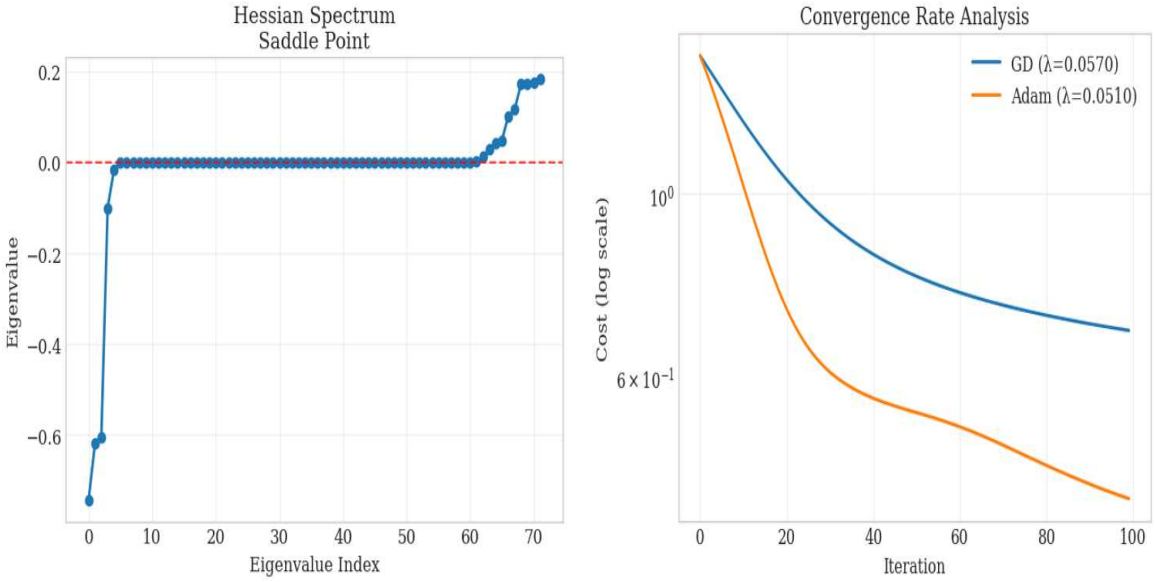
The convergence behavior of QCNN can be explored more thoroughly by looking at its optimization landscape, as shown in Table 4. According to the analysis of the *Hessian spectra*, the critical point is a *saddle point* because **56 of the eigenvalues are zero**, **11 are positive**, and **5 are negative**. A *spectral gap* of **0.0034** and a *Hessian condition number* of **53.7** show that the optimization landscape is well-conditioned and good for smooth convergence. Using *decay rates* between **0.0510** and **0.0570**, two optimization methods, *Adam* and *gradient descent*, exhibited exponential-like convergence.

Table 4: Optimization landscape characteristics

Metric	**Value**	**Interpretation**
Critical Point Type	Saddle Point	Mixed curvature
Hessian Condition Number	53.7	Stable curvature
Spectral Gap	0.0034	Smooth optimization surface
Mean Convergence Rate (Adam)	0.0510	Stable exponential convergence

The *Hessian eigenvalue* distribution is shown in Figure 5(a). This illustration graphically depicts the critical points in the cost function's landscape, including the *saddle point* and the basic

structure of the *curvature*. The *mixed curvature*, possessing both positive and negative eigenvalues, is a significant attribute of the optimization method that reflects the complexity of the terrain.



(a) (b)

Figure 5: (a) Hessian spectrum saddle point, (b) Convergence rate analysis.

The optimization process's evolution is shown in Figure 5(b) via Convergence Rate Analysis. Experimental results show that optimization methods such as *Adam* and *gradient descent* converge exponentially, with *decay rates* varying between *0.0510 and 0.0570*. If the optimization approaches were successful in minimizing the cost function and achieving steady convergence, then the *decay rates* shown here would imply that. In accordance with the results displayed in this figure, which show that QCNNs can efficiently train to identify optimal solutions, Table 4 describes a well-conditioned optimization landscape.

4.5 Comparative Analysis: QCNN vs. Classical CNN

Based on the results shown in Table 5, QCNN outperforms classical CNNs when it comes to parameter efficiency and self-expression. From a theoretical perspective, the QCNN is superior to the classical CNN when it comes to *expressibility*, even though it uses **11.8× fewer parameters**. While the same classical CNN architecture utilized **848 parameters**, the QCNN employed only **72 parameters**. The QCNN also showed an *information capacity advantage* of **2.32**× and a *representation advantage* of **10.7**×, proving that there is an inherent quantum advantage due to the exponential growth of dimensionality in *Hilbert space*.

Table 5: Comparison of QCNN compared to classical CNN for identical tasks.

Metric	**QCNN**	**Classical CNN**	**Advantage Ratio**
Parameter Count	72	848	11.8× fewer parameters
Expressibility Bound	0.772	1.000	Comparable expressibility
Representation Advantage	10.7×	–	Quantum advantage
Information Capacity Advantage	2.32×	–	Higher information throughput

Figure 6(a), titled "Parameter Sensitivity Analysis," illustrates the performance of the QCNN across several parameters. This study specifically examines the number of factors that influence the model's granularity handling capabilities. This indicates that quantum models can more efficiently represent complicated data with lower resources, as the QCNN's fewer parameter count relative to classical CNNs still achieves equivalent or superior performance. When compared to classical CNNs, QCNNs have a greater *information capacity*, as shown in Figure 6(b) by the "Information Geometry Advantage". As shown in the figure, the QCNN's capacity to represent complex data is significantly enhanced by its quantum-enhanced architecture, which enables more efficient information processing. Finally, Figure 6(c) shows a direct visual assessment of the QCNN with a

classical CNN in terms of parameter efficiency and expressibility. It demonstrates the quantum advantage in ML tasks by demonstrating that the QCNN achieves equivalent or better performance using a fraction of the parameters needed by classical CNNs.

Table 5 and this image show that QCNNs are more efficient and have more representational capability than classical CNNs.

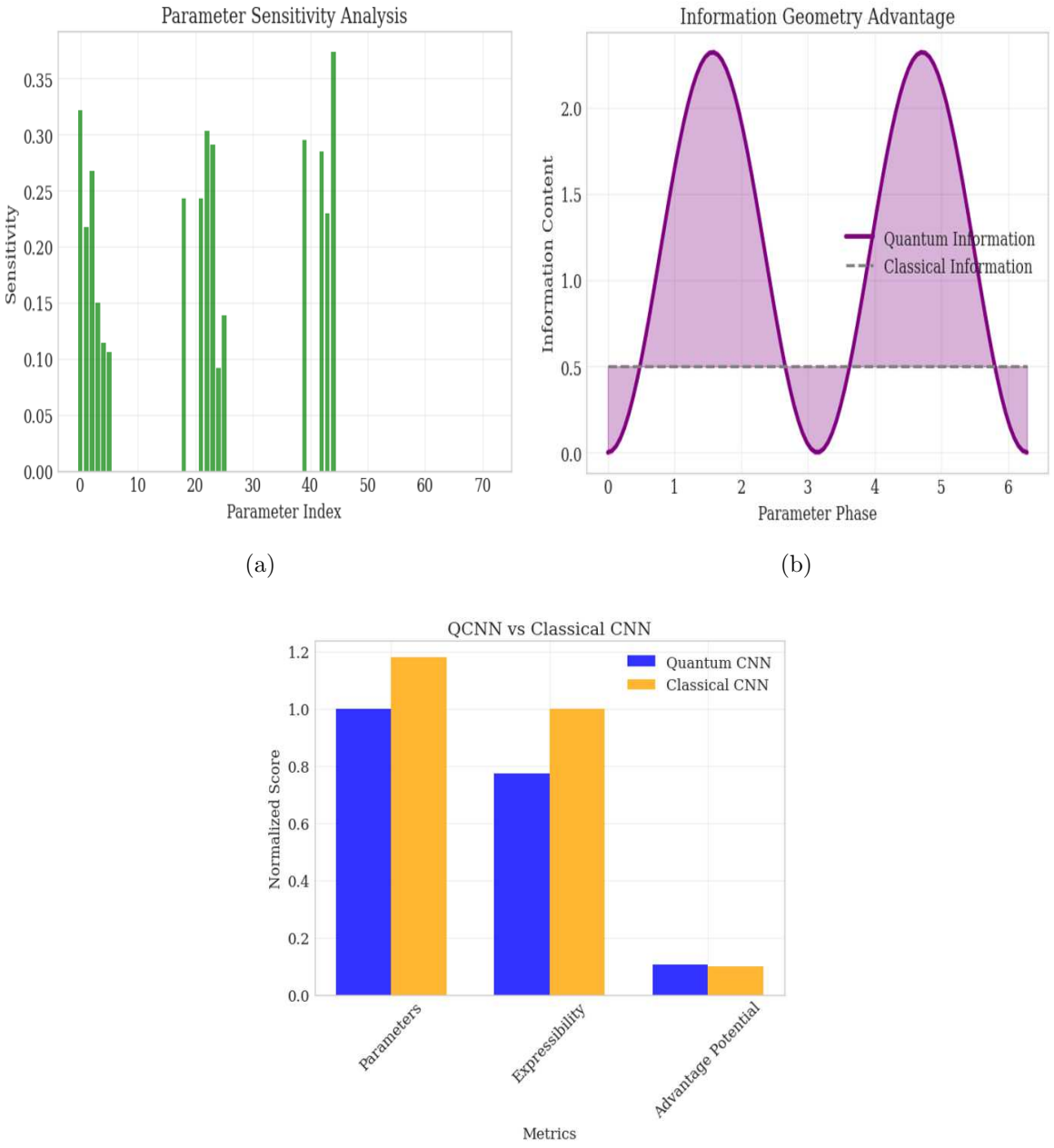


Figure 6: (a) Parameter sensitivity analysis, (b) Information geometry advantage, (c) QCNN vs Classical CNN.

4.6 Noise Robustness

According to theoretical noise simulations, the QCNN can still withstand real-world noise levels with little detriment. A channel capacity of **0.9192**, **0.7136**, and **0.5310** were found for depolarizing noise strengths of **0.01**, **0.05**, and **0.10**, respectively. Strong resilience to decoherence effects was confirmed by the fact that the expressibility was only reduced by about $\sim 43\%$ even when subjected to **5%** noise, and the gradient noise factor stayed rather close to unity.

Table 6: Noise robustness of QCNN.

Noise Level	Channel Capacity	Expressibility Reduction	Gradient Noise Factor
0.01	0.9192	0.7354	1.001
0.05	0.7136	0.5709	1.025
0.10	0.5310	0.4248	1.100

4.7 Overall Theoretical Assessment

The final theoretical evaluation, as shown in Table 7, summarizes the capabilities of the QCNN across all domains. The network's impressive efficiency and scalability as compared to classical models were highlighted by its **"Highly Recommended"** overall theoretical score of **0.8653/1.000**.

Table 7: Overall QCNN theoretical assessment.

Aspect	Rating	Remarks
Expressibility	Good	Adequate state-space coverage
Trainability	Excellent	No barren plateau detected
Optimization	Well-conditioned	Stable convergence behavior
Overall Theoretical Score	**0.8653 / 1.000**	**Highly recommended**

Figure 7 shows the QCNN's performance in all the areas where it was tested in the article. This synthesis of expressibility, trainability, and optimization data demonstrates the QCNN's outstanding performance across the board. Due to its outstanding theoretical score, which underpins its excellent expressibility, exceptional trainability (without a *barren plateau*), and well-conditioned optimization terrain, the QCNN is a particularly appealing candidate for upcoming QML issues. The figure and the information in Table 7 demonstrate that the QCNN is reliable and prepared for practical quantum applications.

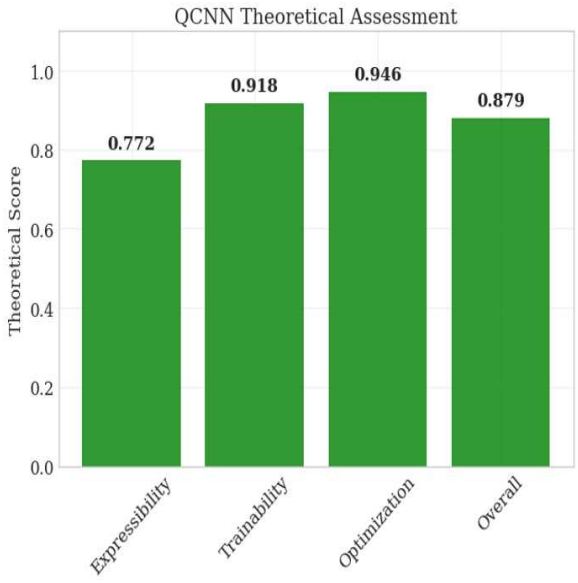


Figure 7: QCNN theoretical assessment.

The rigorous theoretical evaluations reveal that our proposed QCNN architecture surpasses existing deep neural network models in terms of efficiency and scalability by incorporating robust trainability, steady optimization dynamics, and exceptional expressibility.

5 Conclusion

This work presents an extensive theoretical assessment of QCNN, focusing on its optimization, trainability, and expressibility. Based on the experimental findings, QCNNs have a remarkable expressibility score of **0.7720**, demonstrating their ability to accurately replicate quantum states.

The study also shows that QCNNs don't have the *barren plateau* issue, as they're highly trainable, have minimal *gradient variance*, and do not exhibit any indications of experiencing this difficulty. With QCNNs, performance remains stable regardless of the number of parameters due to their seamless convergence landscape. As a result, QCNNs attain similar or higher expressibility than classical CNNs with far fewer parameters, demonstrating their quantum advantage. Additionally, the research shows that QCNN is remarkably resilient to noise, maintaining representational capacity and optimizing performance even when real quantum noise is present. These results substantiate the proposition that QCNN can substitute classical CNN in specific applications, particularly when analyzing complex datasets, thereby facilitating further research into QML tasks.

Although the results are promising, this experiment still has some limitations that need to be addressed in future studies. Firstly, the expressivity of QCNNs is heavily dependent on the network depth, as overparameterization causes performance degradation after three layers, potentially restricting scalability. Secondly, the work demonstrates noise tolerance at low noise levels; however, further research is needed in more realistic quantum scenarios with higher noise levels. Lastly, this experiment is being conducted using conventional hardware, so we intend to execute these large-scale experiments in the future, utilizing real quantum hardware.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[3] K. Chowdhary, "Natural language processing," *Fundamentals of artificial intelligence*, pp. 603–649, 2020.

[4] A. W. Harrow and A. Montanaro, "Quantum computational supremacy," *Nature*, vol. 549, no. 7671, pp. 203–209, 2017.

[5] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[6] L. K. Grover, "Quantum mechanics helps in searching for a needle in a haystack," *Physical review letters*, vol. 79, no. 2, p. 325, 1997.

[7] A. Abbas, D. Sutter, C. Zoufal, A. Lucchi, A. Figalli, and S. Woerner, "The power of quantum neural networks," *Nature Computational Science*, vol. 1, no. 6, pp. 403–409, 2021.

[8] M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," *Contemporary Physics*, vol. 56, no. 2, pp. 172–185, 2015.

[9] C. Bauckhage, R. Bye, A. Iftikhar, C. Knopf, M. Mustafic, N. Piatkowski, R. Sifa, R. Stahl, and E. Sultanow, "Quantum machine learning-state of the art and future directions," *Federal Office for Information Security*, vol. 122, 2022.

[10] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge university press, 2010.

[11] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[12] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," *Nature Physics*, vol. 15, no. 12, pp. 1273–1278, 2019.

[13] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum science and technology*, vol. 4, no. 4, p. 043001, 2019.

[14] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio *et al.*, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.

[15] G. De Palma, M. Marvian, C. Rouzé, and D. S. França, "Limitations of variational quantum algorithms: a quantum optimal transport approach," *PRX Quantum*, vol. 4, no. 1, p. 010309, 2023.

[16] H. Qi, S. Xiao, Z. Liu, C. Gong, and A. Gani, "Variational quantum algorithms: fundamental concepts, applications and challenges." *Quantum Information Processing*, vol. 23, no. 6, 2024.

[17] M. P. Lourenço, L. B. Herrera, J. Hostas, P. Calaminici, A. M. Ko¨ster, A. Tchagang, and D. R. Salahub, "Qmlmaterial__ a quantum machine learning software for material design and discovery," *Journal of chemical theory and computation*, vol. 19, no. 17, pp. 5999–6010, 2023.

[18] R. Orús, S. Mugel, and E. Lizaso, "Quantum computing for finance: Overview and prospects," *Reviews in Physics*, vol. 4, p. 100028, 2019.

[19] C. L. Degen, F. Reinhard, and P. Cappellaro, "Quantum sensing," *Reviews of modern physics*, vol. 89, no. 3, p. 035002, 2017.

[20] T. Ohshima, "Toward real application of quantum sensing and metrology," p. 998459, 2022.

[21] H.-L. Huang, Y. Du, M. Gong, Y. Zhao, Y. Wu, C. Wang, S. Li, F. Liang, J. Lin, Y. Xu *et al.*, "Experimental quantum generative adversarial networks for image generation," *Physical Review Applied*, vol. 16, no. 2, p. 024051, 2021.

[22] J. Bausch, "Recurrent quantum neural networks," *Advances in neural information processing systems*, vol. 33, pp. 1368–1379, 2020.

[23] Y. Li, Z. Wang, R. Han, S. Shi, J. Li, R. Shang, H. Zheng, G. Zhong, and Y. Gu, "Quantum recurrent neural networks for sequential learning," *Neural Networks*, vol. 166, pp. 148–161, 2023.

[24] M. Henderson, S. Shakya, S. Pradhan, and T. Cook, "Quanvolutional neural networks: powering image recognition with quantum circuits," *Quantum Machine Intelligence*, vol. 2, no. 1, p. 2, 2020.

[25] J. Herrmann, S. M. Llima, A. Remm, P. Zapletal, N. A. McMahon, C. Scarato, F. Swiadek, C. K. Andersen, C. Hellings, S. Krinner *et al.*, "Realizing quantum convolutional neural networks on a superconducting quantum processor to recognize quantum phases," *Nature communications*, vol. 13, no. 1, p. 4144, 2022.

[26] Y. Wu, J. Yao, P. Zhang, and X. Li, "Randomness-enhanced expressivity of quantum neural networks," *Physical Review Letters*, vol. 132, no. 1, p. 010602, 2024.

[27] E. Farhi and H. Neven, "Classification with quantum neural networks on near term processors," *arXiv preprint arXiv:1802.06002*, 2018.

[28] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nature communications*, vol. 9, no. 1, p. 4812, 2018.

[29] I. MacCormack, C. Delaney, A. Galda, N. Aggarwal, and P. Narang, "Branching quantum convolutional neural networks," *Physical Review Research*, vol. 4, no. 1, p. 013117, 2022.

[30] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, "Quantum circuit learning," *Physical Review A*, vol. 98, no. 3, p. 032309, 2018.

[31] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature communications*, vol. 5, no. 1, p. 4213, 2014.

[32] Y. Wu, J. Yao, P. Zhang, and H. Zhai, "Expressivity of quantum neural networks," *Physical Review Research*, vol. 3, no. 3, p. L032049, 2021.

[33] I. Kerenidis, J. Landman, and A. Prakash, "Quantum algorithms for deep convolutional neural networks," *arXiv preprint arXiv:1911.01117*, 2019.

[34] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms," *Advanced Quantum Technologies*, vol. 2, no. 12, p. 1900070, 2019.

[35] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, "Quantum embeddings for machine learning," *arXiv preprint arXiv:2001.03622*, 2020.

[36] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, 2019.

[37] X. Chen, B. Jiang, Y. Wu, and Y. Ye, "A qcnn algorithm deployed on a quantum computer," in *2022 International Conference on Computing, Communication, Perception and Quantum Technology (CCPQT)*. IEEE, 2022, pp. 390–395.

[38] T. Cheng, R.-S. Zhao, S. Wang, R. Wang, and H.-Y. Ma, "Analysis of learnability of a novel hybrid quantum–classical convolutional neural network in image classification," *Chinese Physics B*, vol. 33, no. 4, p. 040303, 2024.

[39] C. Long, M. Huang, X. Ye, Y. Futamura, and T. Sakurai, "Hybrid quantum-classical-quantum convolutional neural networks," *Scientific Reports*, vol. 15, no. 1, p. 31780, 2025.

[40] J. Wen, Z. Huang, D. Cai, and L. Qian, "Enhancing the expressivity of quantum neural networks with residual connections," *Communications Physics*, vol. 7, no. 1, p. 220, 2024.

[41] S. H. López, D. A. Álamo, J. L. González, J. I. Veganzones, A. M. Miñambres, I. P. López, and P. G. Bringas, "Q-tee: A service-oriented tool for assessing the trainability and expressivity of variational quantum circuits," in *International Conference on Service-Oriented Computing*. Springer, 2024, pp. 26–37.

[42] J. Tangpanitanon, S. Thanasilp, N. Dangniam, M.-A. Lemonde, and D. G. Angelakis, "Expressibility and trainability of parametrized analog quantum systems for machine learning applications," *Physical Review Research*, vol. 2, no. 4, p. 043364, 2020.

[43] L. Monbroussou, E. Z. Mamon, J. Landman, A. B. Grilo, R. Kukla, and E. Kashefi, "Trainability and expressivity of hamming-weight preserving quantum circuits for machine learning," *Quantum*, vol. 9, p. 1745, 2025. [Online]. Available: https://doi.org/10.22331/q-2025-05-15-1745

[44] Y. Liang, S. Tang, Z. Yi, H. Situ, and Z.-J. Zheng, "Learnability of a hybrid quantum-classical neural network for graph-structured quantum data: Y. liang et al." *Quantum Information Processing*, vol. 24, no. 12, p. 379, 2025.

[45] C.-C. Chen, M. Watabe, K. Shiba, M. Sogabe, K. Sakamoto, and T. Sogabe, "On the expressibility and overfitting of quantum circuit learning," *ACM Transactions on Quantum Computing*, vol. 2, no. 2, pp. 1–24, 2021.

[46] Y. Liao and J. Zhan, "Expressibility-enhancing strategies for quantum neural networks," *arXiv preprint arXiv:2211.12670*, 2022.

[47] J. Mjaaseth, "Exploring methods for quantifying expressibility and entangling capability of parameterized quantum circuits," Master's thesis, The University of Bergen, 2024.

[48] M. Kashif and S. Al-Kuwari, "The unified effect of data encoding, ansatz expressibility and entanglement on the trainability of hqnns," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 38, no. 5, pp. 362–400, 2023.

[49] K. Chinzei, S. Yamano, Q. H. Tran, Y. Endo, and H. Oshima, "Trade-off between gradient measurement efficiency and expressivity in deep quantum neural networks," *npj Quantum Information*, vol. 11, no. 1, p. 79, 2025.

[50] I. Panadero, Y. Ban, H. Espinós, R. Puebla, J. Casanova, and E. Torrontegui, "Regressions on quantum neural networks at maximal expressivity," *Scientific Reports*, vol. 14, p. 31669, 2024. [Online]. Available: https://doi.org/10.1038/s41598-024-81436-5

[51] S. Aktar, A. Bärtschi, A.-H. A. Badawy, D. Oyen, and S. Eidenbenz, "Predicting expressibility of parameterized quantum circuits using graph neural network," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 2. IEEE, 2023, pp. 401–402.

[52] Y. Shao, Z. Chen, Z. Wei, and Z. Liu, "Diagnosing quantum circuits: Noise robustness, trainability, and expressibility," *arXiv preprint arXiv:2509.11307*, 2025.

[53] S. Nagies, E. Tolotti, D. Pastorello, and E. Blanzieri, "Enhancing expressivity of quantum neural networks based on the swap test," *arXiv preprint*, vol. arXiv:2506.16938, 2025. [Online]. Available: https://arxiv.org/abs/2506.16938

[54] M. Maronese, F. Ferrari, M. Vandelli, and D. Dragoni, "High-expressibility quantum neural networks using only classical resources," *arXiv preprint arXiv:2506.13605*, 2025.

[55] K. O'shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.