Quantum CNN Application for Image Classification: A Comparative Study with Classical CNNs

Joanna Pedretti
New York Institute of Technology
jpedrett@nyit.edu
February 2025

Abstract

This study investigates the performance metrics and computational tradeoffs of both classical and quantum convolutional neural networks (CNN) when given noisy input images as training data. The experiments aim to determine whether or not quantum CNNs are more robust to noisy data compared to their classical counterparts. The design of the experiment begins by generating artificial Gaussian noise onto the Fashion MNIST dataset, then comparing the accuracy and other performance metrics of each model two times: first with the clean dataset, then again when noise is added at a high severity. The results show promise that quantum CNNs are more robust to noisy and limited data, and that quantum machine learning techniques have the potential to match or possibly outperform the test accuracy of classical models once the limitations of quantum computing are solved.

Keywords CNN, Quantum Machine Learning, QCNN, Image Classification, Noisy Data

1 Introduction

Quantum computers are still in their developmental stages, but have already shown promising advantages over classical computers in certain tasks. The ability to implement principles of quantum physics, such as superposition and quantum entanglement, allow quantum computers to perform natural parallel processing and provide significant speedup for some algorithms. But these machines don't always outperform conventional methods, which is why it is important to study the performance comparisons between quantum and classical computers during different tasks. A good starting point for this comparison is understanding the key differences between quantum and classical computing,

Classical computers, like the ones we use in our everyday lives, operate using 'bits' which have a binary value of either 0 or 1. Quantum computers use quantum bits, or 'qubits', to perform computations; and they are able to hold a value of 0, 1, or both 0 and 1 at the same time due to superposition. This superposition is a fundamental principle of quantum physics, and it allows quantum computers to perform multiple computations at once by exploring multiple inputs/solutions simultaneously [1]. Qubits can also utilize quantum entanglement: if two qubits are entangled, the state of one qubit will directly affect the other any time it changes [2]. Between these two principles, quantum computers are able to perform parallel computations naturally as opposed to classical computers needing multiple processors or multithreading techniques. Quantum algorithms are created with circuits, where quantum gates are applied to qubits in order to rotate them and manipulate their quantum state to perform computations [1]. For convolutional neural networks (CNN), where the classical CNN uses convolution layers and pooling layers, a quantum CNN will implement those layers with quantum circuits. Parameterized quantum gates are used to build these circuits, and the parameters are able to be tuned to assist with minimizing the loss function for the model [3].

After understanding the potential power of quantum computers, it is clear that studying the best use cases for quantum computing is an important area of research. Finding the tasks in which they excel compared to conventional methods would provide valuable insights into the possibilities of advancement in the future. Other works have proven the potential for quantum algorithm supremacy over classical methods in tasks such as optimization, cryptography, and simulations that are too complex for classical computers to handle [1]. Some research has also been done to investigate the efficiency of quantum CNNs on image processing and pattern recognition tasks, and although there are some challenges, many benefits have been discovered. To begin the computations, classical data points must be encoded to qubits so they can be seen as quantum states. This encoding maps the data to a higher dimension in Hilbert space (where quantum computers can naturally operate), and allows quantum CNNs to find complex patterns within the data [13]. The convolution layer in quantum CNNs is very strong at determining patterns between data points that may not be detectable through classical algorithms. The convolution circuit works by coupling pairs of qubits through rotations and entanglement [3], which strongly correlates the data encoded into each qubit together. This allows the circuit to generate highly complex filters for feature extraction that would be impossible for classical convolution due to the utilization of quantum entanglement [11]. The pooling layer in quantum CNNs is important for reducing the computational complexity and cost of the circuit, by reducing the number of qubits on which we perform quantum gates [5]. For example, the information on two qubits could be pooled into one qubit while retaining the most important data, and therefore we would only need to perform gate operations on the one remaining qubit moving forward. This method effectively cuts the dimensionality of the circuit in half and reduces the number of parameters that the quantum CNN needs to learn on, which allows improved speed and cost efficiency [3]. One final benefit of quantum machine learning models is the ability to learn efficiently from small, sparse, or noisy training data. Related studies have shown that quantum based models were able to achieve similar or higher test accuracy than classical models when given less training data [16], and that the models were less likely to suffer from overfitting [13]. This high accuracy continued in experiments where the training data had some imperfections [11], demonstrating the potential for quantum computers to be more robust to noisy data.

Other researchers have tested multiple architectures of quantum CNNs and hybrid quantum-classical CNNs to view performance comparisons on image processing tasks. Each study faced similar challenges during their research, mainly involving the hardware limitations of current developmental models of quantum computers. One of the biggest challenges at the moment is the amount of possible errors that can occur during computations, due to the difficulty of keeping qubits stable in their environment [4]. They are sensitive to many environmental factors such as temperature and frequencies that could affect the quantum state of the qubit during computations. There are fields of study dedicated to quantum error correction, but as of now it is still difficult to add more qubits into the current models of quantum computers and be able to error correct every qubit at once [4]. Code can be simulated on classical computers instead of run on quantum processors in an attempt to avoid errors, but the simulations may take much longer to run. Therefore many related studies have struggled to scale their experiments on quantum computers properly [11, 12, 14, 15], and their research has been limited by the amount of qubits currently available for computations. But regardless of these hardware limitations, many benefits have been observed during the experiments and the potential of utilizing quantum computers for machine learning tasks is clear. Across almost all of the related studies researchers have observed faster training times, more efficient model learning, enhanced hyperparameter tuning and quick convergence on higher test accuracy compared to classical CNN implementations for the same image classification task.

In this paper, classical CNN models are compared to simple quantum CNNs, and performance metrics are computed for each implementation during two classification tasks. Gaussian noise is artificially added to the Fashion MNIST dataset from TensorFlow, where a script allows the generation of corruption at different levels of severity. All models are run twice for each implementation, starting with a clean dataset and then again with a high severity of corruption on the data. This allows a comparison between the original ability of each model to accurately classify the images, and the differences when the same model encounters very noisy input. A scaled down version of the images are used in the quantum CNN models due to the limitation of the number of qubits able to be simulated on classical machines. One classical implementation extracts features from the full size 28x28 pixel images, whereas the quantum models use 4x4 pixel images while still retaining the most important identifiers for classification. Two additional classical models also use the same 4x4 pixel images in order to perform a fair comparison. The quantum models are simulated through classical processors, and all models are run using the T4 GPU on Google Colab.

The results of these code experiments show promise for the future of quantum machine learning once the development of these computers is complete. The quantum CNN model performing binary classification obtained similar or higher performance metrics in every category compared to its classical counterpart. When trained on the clean dataset the models performed similarly, but when the noisy dataset was introduced, the quantum model's test accuracy only decreased by 3% whereas the classical model dropped by 11%. For the second experiment performing multiclass classification, again both the quantum and classical models performed similarly in all categories of performance metrics. The impact of noisy data caused the quantum CNN test accuracy to decrease by 9% and the classical CNN to decrease by 13%. Once the many limitations of quantum computing have been solved, there is a strong possibility that future work on more complex quantum models will show even better performance.

2 Related Works

This section explores studies that investigate the performance of classical and quantum convolutional neural networks used for different image classification situations. These machine learning models are used in a variety of fields, for a variety of tasks, and each comes with its pros and cons. Some common issues arise throughout all the studies, which allow us to determine important factors to consider when doing a performance comparison. For example, issues that occurred during training of the model will have a direct impact on the accuracy of the classification task during testing, etc. Reviewing these related works gives a basis for factors that are considered in this study and experiments.

2.1 Classical CNN (CCNN)

Classical CNNs have already proven to fundamentally advance the field of machine learning, particularly in their ability to perform automatic feature extraction and efficient pattern recognition. Many studies have explored improvements in CNN architectures, optimization techniques, and methods of deep learning CNNs, which all contribute to the continual improvement of these model's accuracy and efficiency. But there are also many limitations which are explored through research, where new methods aim to prevent issues like overfitting from occurring. Overfitting may happen if training of a model isn't stopped at a certain point, and the model becomes too accustomed to the training data but is not able to accurately predict new data that it encounters. Especially with noisy input images [7], the model might learn based on the noise and overfit to only noisy inputs. On the other hand, CNN models can also underfit (not learning enough) while having weakly annotated training images that lack information about finding the object in the photo [7]. Limited or sparsely populated datasets can also cause underfitting, but this can be counteracted by using data augmentation methods to artificially increase the amount of training data [10]. One final issue to discuss is situations such as the Vanishing Gradient Problem, which occur as many hidden layers are added into the architecture, especially in deep learning. As the feature dimensions shrink after every layer of the CCNN, the weights begin to shrink and become so low that the model does not learn anymore. Without specific architectures and specific optimization techniques, issues like this may greatly affect the accuracy of a CCNN model [6].

While there may be disadvantages to CCNNs, they have proven to be good for saving time on tasks that are usually performed manually by humans or by other machine learning models, for example in [8] and [10]. While using other machine learning models, doing manual feature extraction sometimes proves to be difficult because of other objects in the background of images, or effects from light/rain/other weather [10]. But the automatic feature extraction module in CCNNs has very advantageous results over the other machine learning methods. In order for a CNN model to learn faster, a method called transfer learning may be implemented. This allows researchers to use pre-trained weights on their model, providing a strong advantage in learning on a new dataset because of the model's knowledge of a previously trained dataset [7]. Transfer learning is very helpful to train quicker, improve accuracy, and converge on optimal hyperparameter values faster [6]. But depending on the complexity of the dataset, there could be multiple thousands of hyperparameters to optimize! It has been shown in [9] that using Hyperparameter Importance Assessment methods on a CCNN model can lead researchers to save time during tuning and only focus on the parameters which affect the model most.

In summary, there are many possible limitations of CCNNs such as overfitting or underfitting based on the quality of the training dataset, not having the correct architecture or optimizer for the model, and time constraints in tuning hyperparameters. Many researchers through the years have studied and created new methods to improve accuracy, efficiency, and speed of CCNNs. There are currently many great models of CCNNs and deep learning CNNs which help us perform tasks in a variety of fields. Table 1 provides an overview of some interesting papers that study classical CNNs, detailing the model architectures, dataset(s) used, results and issues that occurred during their research.

Table 1. Overview of Classical CNN Related Works (* indicates public datasets)

Reference Number	Model Architecture	Dataset(s)	Results	Issues/Limitations
[6]	VGG16, ResNet 18, ResNet 34; with and without transfer learning	SipakMed (pap smear) dataset*	All CCNN models always performed better when using transfer learning	Choice of optimizer significantly affects model performance
[7]	Hover-Net model with preactivated ResNet 50	Simplified versions of MoNuSAC and PanNuke*	Having a correctly annotated validation set is key to avoid overfitting	Annotation noise or weakly annotated data could cause underfitting or overfitting
[8]	MobileNetV2 and VGG16	Healthy/Defective Fruits dataset	Multi-input architecture with both RGB and silhouette images had best test accuracy	Segmentation errors in training data
[9]	N-RReliefF hyperparameter importance assessment conducted on multiple models	10 different image classification datasets	Tuning only most important hyperparameters saves time and resources	Needs to be tested on more models, including deep learning models
[10]	Multiple CNN and deep CNN models	Multiple plant and plant disease datasets	Using data augmentation can help when data is limited	Limited/sparse training data, overfitting issue significantly lowering test accuracy

2.2 Quantum CNN (QCNN)

In comparison to CCNNs, QCNNs utilize quantum circuits to create the feature extraction/pooling modules, and classification models. Hybrid QCNNs (HQCNNs) may use different combinations of classical/quantum features, such as quantum convolution and pooling layers but a classical MLP for classification, or vice versa. These models may be tested through simulations on classical computers, such as in [11, 13, 15], or run on actual quantum computers through the cloud [12, 14]. In simulations we may receive higher accuracies during training and testing, due to the fact that the simulations may not include real-world noise and errors that occur on real quantum computers. The comparison between simulated results and real quantum processor results can be seen in [14], where their QCNN model achieved 99% test accuracy during simulation and only 63% test accuracy when run on IBM Quantum devices through the cloud. Another issue with the current state of quantum models is that there are a limited number of qubits in the hardware of developmental quantum computers. In order to keep the hardware error-free and noise-free, researchers have started small and are building up to add more qubits [4]. Therefore the current ability to program quantum circuits is limited based on the number of qubits available during the experiment, depending on if the experiment is simulated vs. run through the cloud, and which quantum processor they choose through the cloud. For smaller datasets, such as the Iris dataset used in [13], algorithms can work fine using a 4-qubit circuit by encoding one feature per qubit. But in many other classification tasks, researchers struggled to properly define their quantum circuits due to the amount of qubits available [11, 12, 15]. Some workarounds for now include scaling down the input image sizes or adding a fixed circuit depth regardless of the number of qubits [14]. But as the development of quantum computers progresses and more qubits become available for use, we will see experiments that are able to have much larger circuits and process even more complex datasets.

Although these limitations currently exist, we are still able to witness a variety of benefits from utilizing quantum and hybrid quantum-classical CNNs for image classification and pattern recognition tasks. For example, quantum computers are very efficient at processing complex data with high dimensionality. Their ability to manipulate qubits in Hilbert space allows the possibility of testing both real and imaginary values for each feature, “potentially doubling the number of trainable parameters with the same sample size requirements”[15]. By using parameterized quantum circuits, where gates are composed of real-number parameters accompanied by unitary matrices, we could also use classical optimizers on the (hyper)parameters still if we choose [11]. Although quantum computers are very efficient at optimization tasks due to their natural parallelism [1], utilizing classical optimization methods could be beneficial for resource cost reduction. This advantage in parameter tuning speeds up the training process already, but quantum machine learning models also have proven to train more accurately off of limited/scarce data [16]. In [14], researchers trained a QCNN to search for rare occurrences in a dataset. Along with some real labels of the rare occurrences, researchers added superpositions of the known labels into their training data [14], which would not be possible on a classical CNN that cannot account for superposition. Even though the number of labels for these occurrences were small, the QCNN model still was able to learn quickly and accurately. Evidence has shown that QCNNs can learn more efficiently on complex data sets as well. Due to the entangling gates in the feature extraction circuits, QCNNs can generate highly complex filters (kernels) that are impossible for CCNNs [11]. This allows for the extraction of information and patterns that CCNNs might miss. It also makes QCNNs more robust to noisy input images [11], and less likely to overfit on the training data [13]. Transfer learning can be used between QCNN models for similar datasets as well, which could further increase the accuracy and speed during training [12].

In summary, some of the main advantages seen across various papers include quicker training times, better hyperparameter tuning, higher test accuracy/ability to prevent overfitting, and faster convergence on optimal solutions. There are still many issues with quantum computing hardware that researchers need to overcome in order for us to see these machines at their full potential. Table 2 provides an overview of some impactful papers that study QCNNs and HQCNNs, detailing the model architectures, dataset(s) used, results and issues that occurred during their research.

Table 2. Overview of Quantum CNN Related Works (* indicates public datasets)

Reference Number	**Model Architecture**	**Dataset(s)**	**Results**	**Issues/Limitations**
[11]	4 qubits, quantum convolution layers, quantum max pooling, classical dense layers, SGD optimizer	DICOM Brain Tumor MRI scans, REM-BRANDT*	HQCNN outperformed CNN in test accuracy, reached optimal accuracy in two-thirds the epochs	Limited number of qubits available, had to scale down input image sizes
[12]	Classical deep feature extraction module, quantum classifier	COVID-19 Radiography Dataset (CRD)*	98.1% test accuracy running on IBMQ-QASM quantum processor	Limited number of qubits available, quantum circuit depth
[13]	4 qubits, angle embedding, quantum convolution layer, CCNN classifier	Iris Dataset*	QCNN trained over 20 epochs, obtained 100% test accuracy by epoch 16	Iris dataset is less complex than many other dataset choices
[14]	Fixed circuit depth regardless of number of qubits, quantum convolution, pooling	Quantum Many-Body Scars (QMBS), with added superpositions of known scars	QCNN achieved 99% test accuracy in simulations, 63% test accuracy on IBMQ devices	Noise/errors on quantum computer, limited data available, high computational cost
[15]	16 qubits, QCNN-LSTM, amplitude encoding, quantum convolution and pooling, quantum dense layers, classical classifier	Collected data of patients diagnosed with MS between 2006-2023	Quantum models showed greater efficiency in train time, and slightly higher precision and recall than classical models	Limited number of qubits available due to memory constraints, bias in their collected data

2.3 Applications

Each of the studies above harness the power of convolutional neural networks to perform image processing tasks and classification of images. This technology is particularly useful in situations that require automated feature extraction in order to save time on finding results. CCNNs are already very useful for this automation, but QCNNs are trying to make the process even more efficient and time saving. Multiple papers involve utilizing QCNNs for medical imaging classification tasks. This is due to the fact that medical imaging data usually has high dimensional features, and can be very noisy (ex. If a patient is moving while a scan is being taken) [11]. The enhancements of quantum machine learning allow models to learn more off less data [16], be more robust to noisy inputs and less likely to overfit [11], and find complex patterns/relationships between multiple features [13]; as stated in the previous section. Therefore, quantum computing may be a viable choice in situations such as medical diagnosis. It is also very beneficial for time-sensitive situations, such as detection of diseases in crops [10], food [8], animals, humans [6, 11, 12], and even forecasting the gradual progression of illnesses [15]. Utilizing CCNNs is already helpful in these fields, but there is promise of greater advancement with fully developed quantum computers using QCNN models. QCNNs also proved to have higher test accuracy in every study shown in section 3.2, most of which occurred in less epochs than their classical counterparts. When testing the model shows rapid convergence to high accuracy, it indicates that the model efficiently captured the features of the dataset while also ensuring optimal generalization to unseen data [13]. This generalization is very important in the results of all machine learning models, because it allows the model to be used for real-world applications that may have vastly different distributions in data compared to the training sample. Some QCNN models were even able to learn with the same efficiency off of limited/sparse training data [14, 16]. Many of the CCNN models struggled in these situations, so quantum computing again shows promising advantages.

Aside from image classification, the strong pattern recognition abilities of QCNNs can be utilized for tasks such as quantum phase recognition and quantum error correction. These are complex applications of QCNN, shown in [17], that are not explored as often as image classification tasks in other related works. Nonetheless, these applications show very promising potential for solving problems of a quantum nature that classical machine learning techniques would have a difficult or impossible time executing. Related research has been able to provide evidence that QCNN models can avoid overfitting and have enhanced learning from even small datasets [18]. The QCNN circuit in [17] was only created to recognize one dimensional quantum phases, but the model can easily learn to detect in higher dimensions in the future due to the high generalization capabilities of quantum computers. The quantum error correcting model also showed high potential compared to other known error correction methods, observing similar or reduced error rates each time the QCNN method was tested [17]. These applications show that QCNNs could have promising advantages in various tasks aside from only image classification.

3 Problem Definition

Compared to other works of a similar topic, this study aims to determine the specific performance differences between classical and quantum CNN implementations for a classification task with noisy or 'corrupted' input images. In theory (and observed in some related works [11, 16, 18]), quantum computers may be more robust to noisy, limited, or poorly annotated images and can still show high classification accuracy despite the imperfections. In other works on classical CNNs such as in [7, 8, 10], many researchers noted that their models struggled in situations where the input images had some level of noise, sparsity, or segmentation errors. Aside from those limitations CCNNs perform quite well for most image classification tasks, therefore a question is raised to determine whether or not QCNNs could potentially show a quantum advantage when dealing with noisy data.

Many related works have used MNIST datasets as a proof of concept for the QCNN functionality and performance, so this study focuses specifically on corrupted versions of those input images. It aims to investigate if QCNNs are actually more robust to noise compared to CCNNs, while also comparing the original classification accuracy between the two models using non-corrupted images. This is to demonstrate the validity of the original models before adding noisy inputs, to ensure that any issues that occur after the severe noise is generated is in fact due to the noise and not the model itself. The choice to add Gaussian noise to the input images is meant to simulate what corruption may occur in real-world data, such as in medical scans or satellite imagery. By focusing on realistic input noise, this study contributes to the research in finding reliable machine learning models for practical tasks.

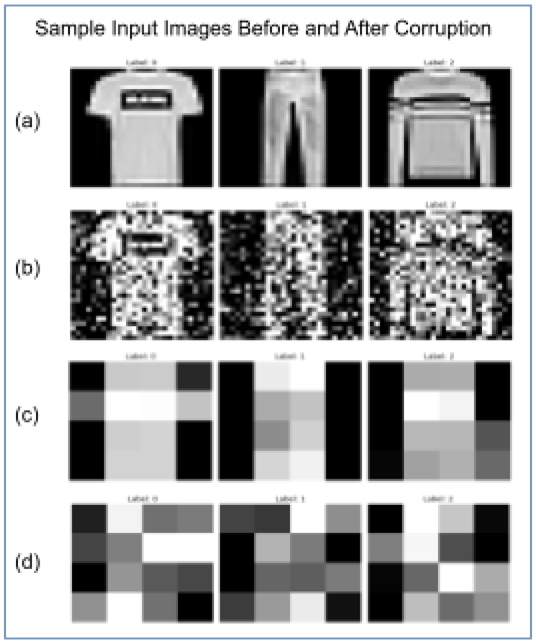
There are three main research questions that guide this study. First, how is the classification accuracy of CCNNs and QCNNs impacted by the generation of Gaussian noise onto their training data? Both CNN models are already proven to work well by other researchers, but it is important to view how the noise will truly impact the models. This relates to the second question, whether or not QCNNs show increased robustness to noisy inputs compared to CCNNs. Many studies have proposed that quantum computers have a high tolerance for noise or imperfections in input data, therefore trying to empirically prove this case would provide insight into the potential uses of quantum computing. Finally, this study investigates what trade offs occur when using classical vs quantum computers for convolution tasks. There are pros and cons to each model architecture and discovering the differences that occur during these experiments will help determine if using these architectures is practical for real-world applications.

4 Approach

In this paper a CCNN architecture with three convolutional blocks is compared to a simple QCNN implemented with 16 qubits. Both models are tested with the same dataset, specifically the Fashion MNIST dataset from TensorFlow Keras Datasets. It contains 10 classes of 28x28 grayscale images, with 60,000 training examples and 10,000 testing examples [19]. For the QCNN implementation, the images were scaled down to 4x4 pixels due to the limitation of the number of qubits able to be simulated in a reasonable time. All 10 classes were used for both models during their respective multi-class classification tasks. A dataset with this structure was chosen because of the image size and the one channel grayscale. Since the images are already low resolution, scaling them down to 4x4 pixels will not lose as much information compared to higher resolution images. The grayscale images also allow easy encoding to qubits in the QCNN model, without needing to implement extra qubits to handle additional RGB channels. Furthermore, artificial Gaussian noise was added to the dataset and each model is tested to determine the specific performance differences between the clean and noisy inputs. Both CCNN and QCNN architectures are defined through a function which was called separately to create clean and noisy models, in order to prevent any implicit transfer learning that could occur by running a model twice in one code file. This allows a fresh model to attempt to train on noisy data without any previous knowledge of the images in the dataset. First each model is run with clean data to ensure the validity of the model before adding noise, and then they are run again with the noisy version of the dataset to compare the performance. If there are issues such as underfitting or overfitting during the tests with clean data, that would indicate that there is a fundamental issue with the model and the noise alone is not impacting the accuracy. If the models are able to classify the images with decent accuracy before corruption is added, we will be able to observe the impact such noise has on a strong model and its classification accuracy.

4.1 Dataset Corruption

The collection of Fashion MNIST images are artificially corrupted by a script which creates Gaussian noise on an image with the choice of varying noise severity. Gaussian noise was the chosen corruption method because it allows the distortions to be evenly distributed throughout each image. For both the CCNN and the QCNN models, images are first preprocessed involving scaling the images to the desired size and normalizing the pixel values. Then a function for adding corruption is defined, where the mean value is set to 0 and the standard deviation is determined by the severity of noise selected in the parameters of the function, inspired by [23]. An array of Gaussian noise is generated in the same shape as the input image array, and then applied by altering the input pixel values with the chosen severity level. The modified pixel values are also clipped to ensure that they remain within the original normalization range from the preprocessing step. Examples of the images are shown, where (a) and (b) depict the 28x28 images and



(c) and (d) are samples of the 4x4 pixel versions. From the images it is evident that while the 4x4 pixel versions may look abstract, the essential features of the images are mostly preserved. The human eye may not be able to fully distinguish between classes at such low resolution, but the computer can still extract meaningful patterns from the data. The Gaussian noise added to the 4x4 images is much more impactful compared to the 28x28 images, but still the computers are able to distinguish between different classes relatively well.

4.2 Model Architectures

The following sections detail the design and architecture of all classical and quantum CNN models used in these experiments. As stated earlier, multiple models are created to use the full size 28x28 images or a scaled down version of 4x4 images to perform multiclass or binary classification tasks. Regardless of image size, the data is all normalized to obtain pixel values between 0 and 1. Each model is trained over 10 epochs, using a batch size of either 128 (for the full size classical model and 4x4 binary classification models) or 32 (for the 4x4 multiclass classification models using subsets of the data). When subsets are taken, the original 6:1 train-to-test ratio of the dataset remains consistent. All models use the Adam optimizer for adaptive learning rates, and either Sparse Categorical Cross Entropy (for multiclass tasks) or Binary Cross Entropy (for binary tasks) loss functions. All models are compared with the same performance metrics, first by graphing the training and validation accuracies and then further computing the Precision, Recall, and F-1 score. All code for each model was compiled and run on the T4 GPU from Google Colab. When each model is tested twice (clean vs noisy data) no modifications are made to any of the models between the two tests.

4.2.1 Multiclass Classification, 28x28 Images

The first classical CNN model created for these experiments accepts 28x28 single channel grayscale images, the full size data for Fashion MNIST. There are three convolutional blocks, the first contains 32 3x3 filters with ReLU activation followed by batch normalization, max pooling with a 2x2 filter and a stride of 2, and a dropout layer (50%) to help prevent overfitting. The second and third blocks each have 64 3x3 filters with ReLU activation, and batch normalization to add extra stability and attempt to enhance performance. The fully connected layer begins with flattening to transition from convolutional layers to dense layers. There is a 128 neuron dense layer and ReLU activation, followed by a dropout layer for regularization, and finally a 10 neuron dense layer with softmax activation in order to output the multiclass classification. A similar model can be seen in [20], which was also used for performing classification tasks on MNIST datasets. The architecture of this model appears to be appropriate for the given task and dataset, and similar architectures are widely used for grayscale datasets. A deep learning model was not chosen for this experiment due to the simplicity of the dataset, in order to prevent issues such as overfitting with a model that is too complex for the given data. In the scope of this paper, there is no QCNN model that directly compares to this CCNN model. In order to encode each pixel of a 28x28 single channel image to qubits, it would require (28*28*1) = 784 qubits using the encoding method implemented in this paper. That is outside the current amount of qubits that can be simulated on classical processors, therefore additional CCNN models are created to have a more fair comparison with the QCNN models.

4.2.2 Binary Classification, 4x4 Images

A QCNN with 16 qubits is reasonable to simulate on classical processors, therefore 16 qubits in a 4x4 grid are mapped to 4x4 single channel grayscale images for the first binary classification model. The qubits are encoded based on a threshold for each pixel value, where if the normalized value is greater than 0.5, an initializing gate will be placed on that qubit. A readout qubit is created for the classification output and initialized with Pauli-X and Hadamard (H) gates. The circuit itself contains two parts, first a layer of entangling gates (XX and ZZ interactions) connects the nearest-neighbor qubits. Then parameterized rotational gates (Ry and Rz) are applied in the second layer, utilizing L-2 regularization to stabilize the model and prevent overfitting. One final Hadamard gate is applied to the readout qubit before measurement, and then the model outputs the result of the binary classification. This QCNN model has 67 trainable parameters and the circuit depth is kept small in order to reduce the simulated training time. A similar model can be found in [21], but some modifications were made to add more layers and regularizers in the model for this paper.

The CCNN model for binary classification has 97 trainable parameters, which is a much more fair comparison with the 64 parameter QCNN model. The model is very simple to keep the number of parameters minimal. One convolutional layer with 4 2x2 filters is applied to accept 4x4 grayscale images, then flattening is performed to pass the data into dense layers. The first fully connected dense layer has 2 neurons, followed by a 1 neuron output dense layer to get the binary classification results. There are no pooling or dropout layers introduced in this model because the size of the data is already very small with only 16 pixels total, so further reducing dimensionality could have adverse effects. A similar method of comparing models by their number of trainable parameters is shown in [22], and the model in this experiment is based on their CCNN model with low trainable parameters. Some modifications are made to account for Fashion MNIST being a slightly more complex dataset than MNIST, but the model overall is still very simple in order to minimize the number of trainable parameters and provide a fair comparison with the binary QCNN model.

4.2.3 Multiclass Classification, 4x4 Images

The quantum CNN performing multiclass classification has a more complex circuit with higher depth, in order to fully encapsulate features from all 10 classes. The model uses a 16 qubit nearest neighbor entanglement architecture with parameterized rotational quantum gates, similar to [15] and also inspired by [18], and has 290 trainable parameters. The same binary threshold encoding method from the previous QCNN model is also implemented here, and the structure of the circuit is similar as well. But in this multiclass model, two additional rotational gates (Ry and

Rz) are added, as well as a full second layer of entanglement and parameterized gates. Therefore where the previous model had one entangling layer and one set of Ry and Rz gates, this QCNN model has two entangling layers and two sets of four parameterized gates per qubit in the circuit. Each parameterized gate has added L-2 regularization for stability, and the PQC layer acts as a quantum feature extractor before the output is fed to two classical dense layers for classification. A dense layer with 8 neurons and ReLU activation is utilized to process the quantum output, and an output layer with 10 neurons (for the 10 classes in this dataset) is used to output class probabilities. In this QCNN model, measurements are taken for each individual qubit using the Z-basis for classification.

The classical CNN model for multiclass classification with 4x4 pixel images is a very simplified version of the original CCNN that classifies the full size images. The structures both have three layers, but in this model there are only 4 filters in the first convolutional block and 8 filters in the second and third blocks. All filters have a 2x2 size to accommodate the smaller images, and one pooling layer is implemented with 2x2 average pooling and padding to stop the dimensionality from decreasing too much. Average pooling was used for the smaller data size because it allows the model to retain a wider range of spatial information instead of only keeping the maximum activation value like in max pooling. In order to keep the number of trainable parameters to a minimum (318 total), the fully connected layer of this CCNN involves only flattening and a 10 neuron dense layer with softmax activation for the output classification. The dense layer has added L-2 regularization to be more comparable in stability to the QCNN.

5 Experimental Results

In order to compare machine learning models, performance metrics such as accuracy, loss, precision, recall, and F1-score can be computed. During the training of convolutional neural networks, the accuracy and loss for the training and validation data are shown per epoch and updated in real time as the model trains. Afterwards it is tested once more using the test dataset split, to determine the model's generalization accuracy when it comes to previously unseen data. The training and validation accuracy are graphed to visualize the outcome of how the model learns over each epoch. Then the remaining performance metrics are computed using built-in functions from ScikitLearn Metrics, which calculate the following:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

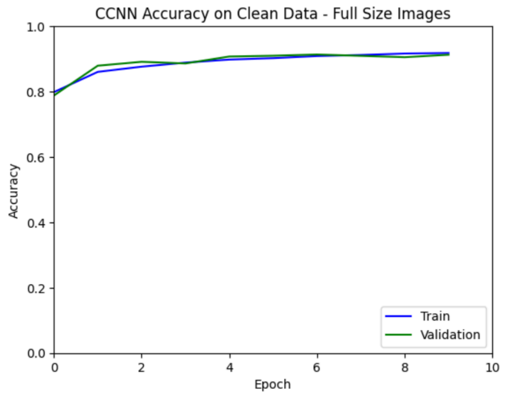
$$\text{Recall} = \frac{TP}{TP + FN}$$

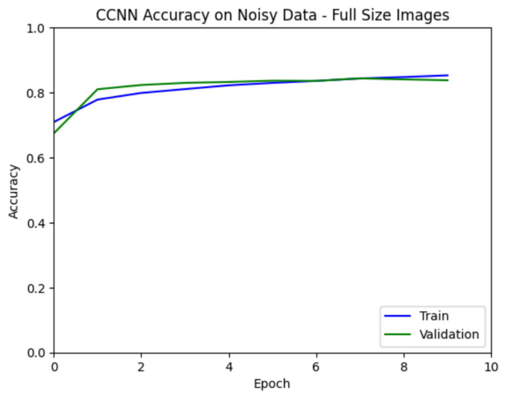
$$\text{F1 Score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

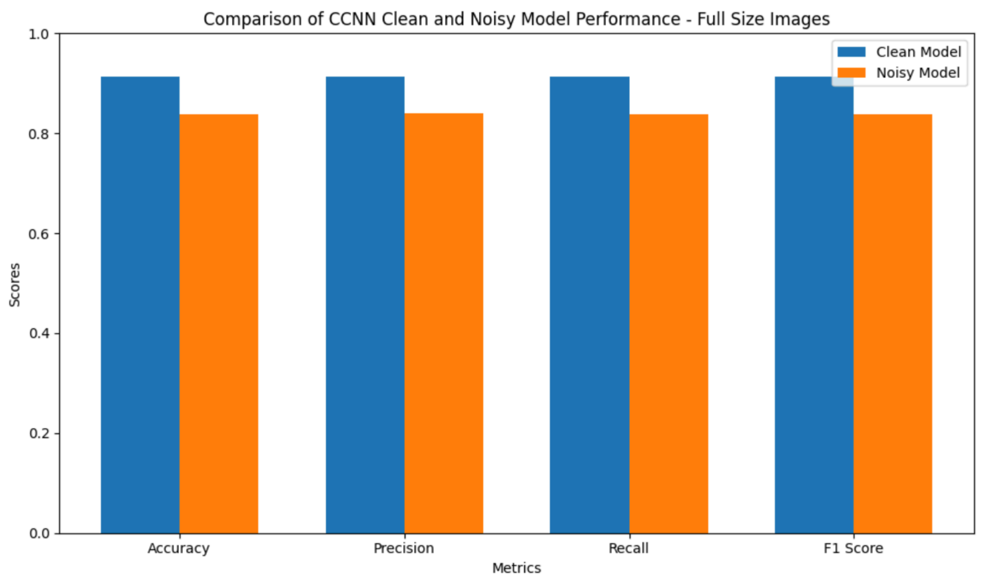
Where TP is true positive, TN is true negative, FP is false positive, and FN is false negative. These values are determined by the ground truth labels of the dataset compared to the predicted values the model outputs. True positives occur when the model correctly predicts the positive class, and true negatives occur when the model correctly predicts the negative class. False positives are errors where the model predicts a value of positive when the ground truth is actually negative, and vice versa for false negatives. In a multi-class classification task, these values are computed individually for each class. Therefore when determining the positive and negative classes, it will consider the current class as 'positive' and all other classes are grouped together as the 'negative' class.

5.1 Multiclass CCNN, 28x28 Images

It was important to first test the base case of this experiment with a CCNN using full size 28x28 images to note the effect that noisy images had on a well functioning model. From the accuracy plot with clean data, it is evident that this CCNN model had no problems performing the multiclass classification on Fashion MNIST which is to be expected. The model had 721, 354 trainable parameters and was able to be optimized and fine tuned to obtain high accuracy. The results showed 91.28% test accuracy, and the other performance metrics such as precision, recall, and F-1 score followed suit. This indicates that the CCNN model was efficient at performing the given classification task and that it is a good candidate to use for this dataset. The training time for this model completed in just under a minute. In comparison, the same model trained on noisy data had 83.79% test accuracy which showed a 7.49% drop, along with a decrease in the other performance metrics. It is clear that the CCNN model was not very robust to noisy inputs, and although the model did not seem to overfit to the noise within the 10 training epochs, the accuracy was still affected by the Gaussian noise added to the images. Despite the added noise, the training time of this model was not affected and still completed in under a minute. These results align with the hypothesis that noisy or limited data will negatively affect classical model performance, similar to the results seen in [7, 10, 18]. The bar chart shows all four performance metrics for both clean and noisy data, and we can observe a noticeable drop in all metrics for the noisy model compared to the model trained on clean data.

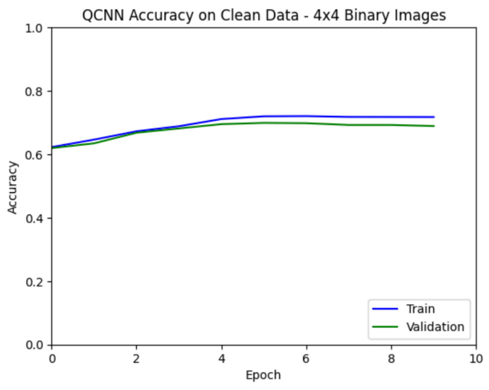


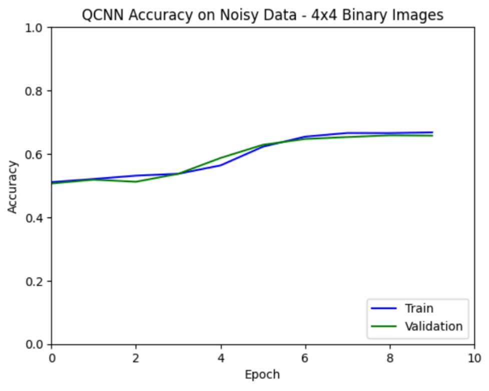




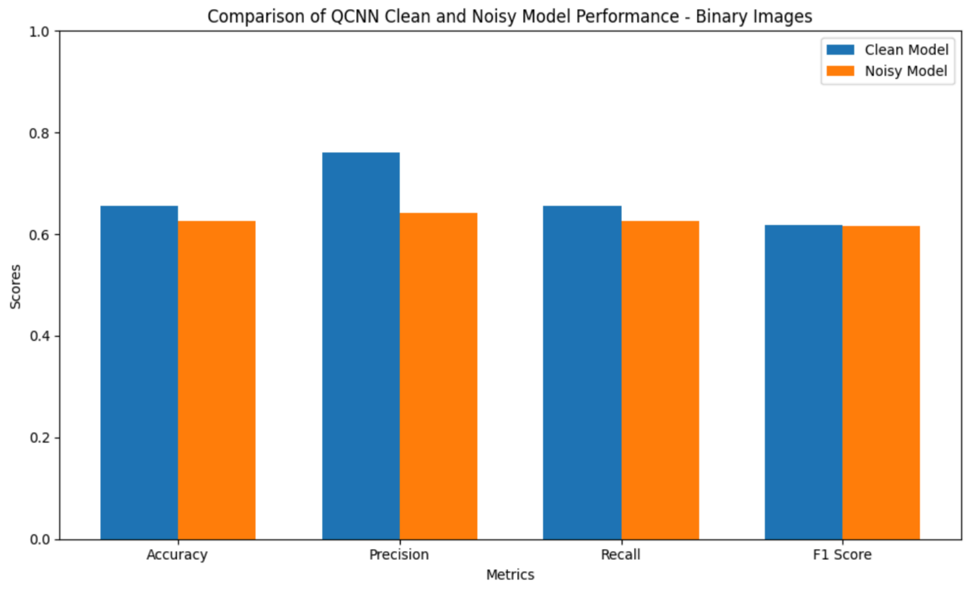
5.2 Binary QCNN and CCNN, 4x4 Images

The first quantum CNN model in this experiment used 4x4 images to perform a binary classification task with two out of ten of the dataset classes selected for training and testing. The training time for 10 epochs took over two hours due to the fact that this quantum circuit was simulated on classical processors (using the T4 GPU on Google Colab), which is one current limitation of quantum computing. The model was limited to only 64 trainable parameters because increased circuit depth leads to increased training time. Although training seems very slow compared to CCNNs, the quantum model's training and validation accuracy quickly converged to 65.65% within the first few epochs. It begins to plateau around epoch 5, therefore early stopping could have been implemented to end training once the model stopped significantly improving. Early stopping was not added to this model because all CNN models in these experiments were trained over 10 epochs for consistency and fair comparison of results. Even though the accuracy plateaued, overfitting was not deemed a significant issue in this QCNN model because the validation accuracy only begins to drop at the last few epochs. Similar plateauing of accuracy can be found in the results of [18], where the model reaches peak accuracy early on and doesn't show significant improvement over additional epochs. When trained on the noisy dataset, the QCNN model proved to be more robust to noise and obtained 62.65% test accuracy, which is only 3% less than

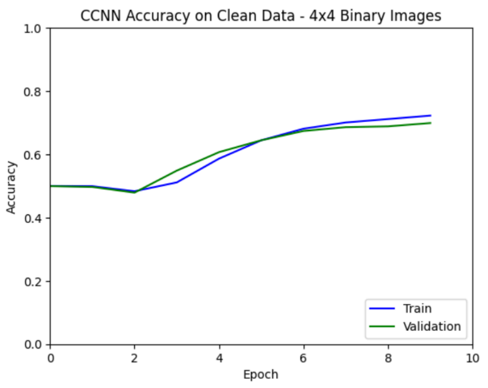




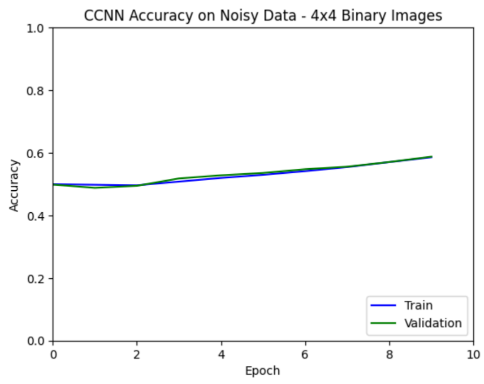
training on clean data. The noisy model took longer to converge and only began to plateau at around epoch 7, and no overfitting occurred. This result is impressive because the severity of Gaussian noise added to a 4x4 image is much more impactful compared to the full size image. When we view the sample images with and without noise, it is evident that Gaussian noise on a 4x4 image almost completely obstructs the identifying features of the image to human eyes. But due to superposition, entanglement, and parameterized quantum gates, the QCNN model was still able to learn with decent efficiency and show more robustness to noise than the CCNN model with full size images. In the bar chart shown, it is noted that the largest gap between performance metrics for the clean vs noisy QCNN models was in precision, where the clean model had an 11.75% gap from the noisy model. This result suggests that false positives are being affected by the added noise compared to false negatives. The clean model had less false positive classifications overall compared to the noisy model, but for the rest of the metrics, both models performed similarly and relatively well considering the limitations.

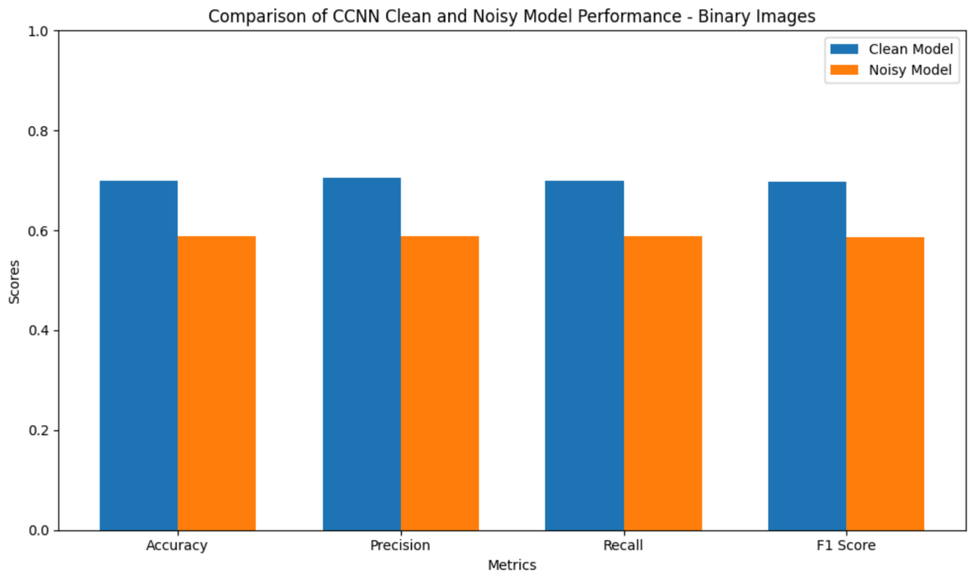


The simple classical CNN model which also used 4x4 images for binary classification did not perform as well when noise was introduced to the data. This model was created with 97 trainable parameters to be more fairly compared to the binary QCNN model. For both clean and noisy models the training time was again very fast, taking only a minute to complete. Despite the quick training, the clean model obtained 69.90% accuracy after 10 epochs which is very similar to the QCNN clean model results. Slight overfitting begins towards the end of training as we can see the validation accuracy begin to drop, but it is not deemed a significant problem with the model. This model took



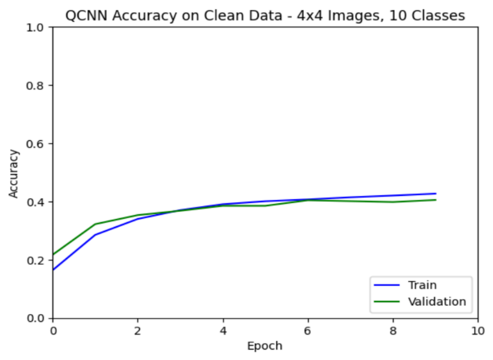
longer to converge to its maximum accuracy, which only occurred around epoch 9. This lower accuracy is expected compared to the CCNN model with 28x28 images because the dataset is now limited to 4x4 size and only 12,000 training and 2,000 testing examples since two classes are extracted for binary classification, similar to [22]. But the noise introduced to that data had a more significant impact compared to the QCNN model. The noisy CCNN model accuracy dropped to 58.75%, and the other performance metrics were also negatively affected. There was an 11.15% difference between the clean vs noisy accuracy results on this model, which is much larger than the QCNN model performing the same task. These results align with the information indicating that CCNN models have a harder time learning from noisy and limited data. The bar chart is used to further visualize the performance metric comparison, and it is clear that the results are not as strong as the comparable QCNN model.

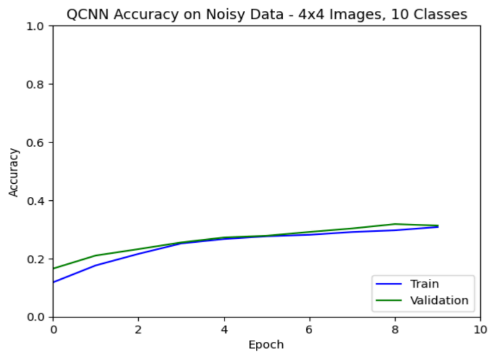


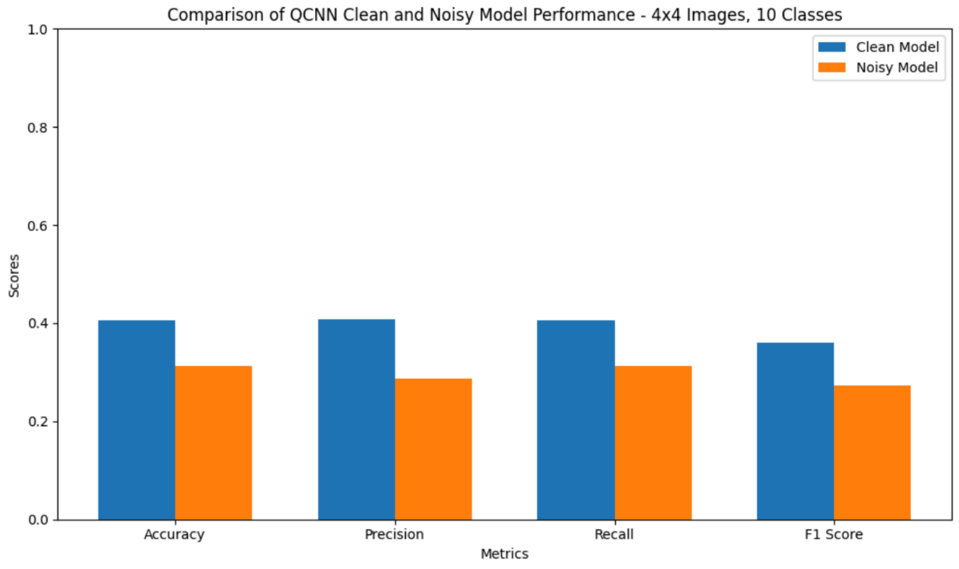


5.3 Multiclass QCNN and CCNN, 4x4 Images

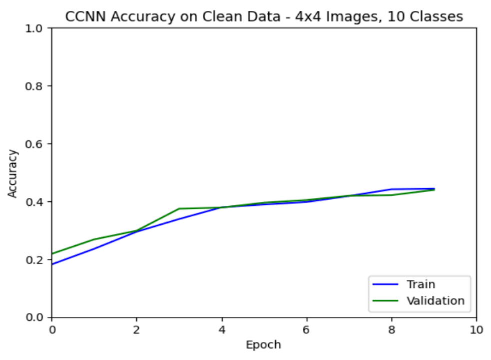
In the second set of experiments, all 10 classes of Fashion MNIST were used to train new quantum and classical models on clean and noisy data. Both models performed similarly in this case, but the quantum CNN did show a slightly higher robustness to noise. This 290 parameter quantum model took over two hours to train for 10 epochs while being simulated on classical processors, which was slightly longer than the training time of the binary classification QCNN due to the increased number of trainable parameters. The test accuracy after 10 epochs was 40.50% for clean data and 31.30% for noisy data. These results indicate that the model had a hard time performing multiclass classification on 4x4 pixel images, and that noise did have a significant impact on the classification task since the severity of noise on 4x4 pixel images is higher than on a better resolution image. The training and validation curves converged quickly and were stable for both the clean and noisy QCNN models, but due to the limitation of simulating qubits this quantum circuit does not appear to be complex enough to fully encapsulate the features of this 4x4 multiclass dataset. Nonetheless, the impact of noisy data only caused the test accuracy to drop by 9.2%, which is less than the comparative CCNN model. In every metric category for noisy data this QCNN model outperformed the comparison CCNN model, but for the clean data metrics, the CCNN was a few percent higher. Although neither model had high accuracy for this classification task, the quantum robustness to noisy data is still apparent in these results. The bar chart below shows the comparison of the performance metrics of the QCNN model in this experiment for reference.

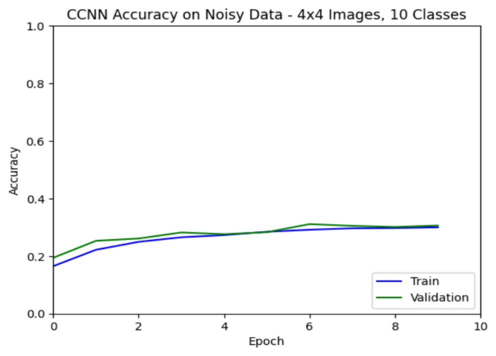


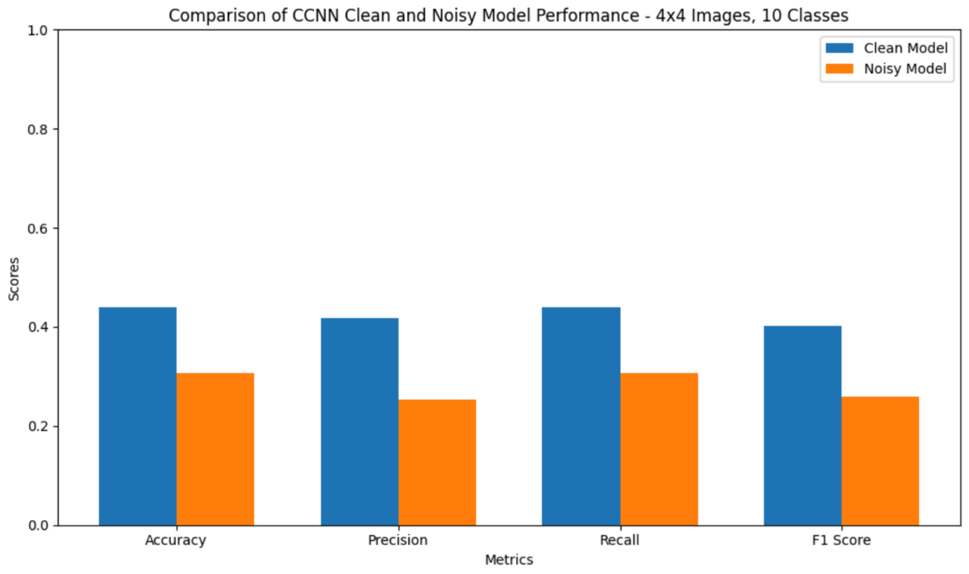




The convergence of the clean and noisy classical CNN models occurred at about the same rate as the QCNN model, but the validation curves from the CCNN appear less stable. This classical model achieved 43.90% test accuracy on clean data, which is 3.4% higher than the comparative quantum model, but it could be due to the CCNN having 318 trainable parameters (28 more than the quantum model). But this higher accuracy led to a larger gap between the clean and noisy models in all performance metrics, and the noisy CCNN metrics were lower than the noisy QCNN metrics in every category. The CCNN ended up with 30.60% test accuracy when trained on the noisy dataset, which was a 13.3% decrease. This again aligns with the theory that quantum computers have a higher robustness to noisy inputs. The low accuracy from both models overall is most likely caused by the size of the dataset since subsets were taken, therefore there were only 600 training and 100 testing examples per class (10 classes) and both the QCNN and CCNN model had a difficult time learning from this small dataset. As seen in the graphs and the bar charts these two models performed very similarly and were both negatively impacted by the addition of Gaussian noise to the data. Overfitting does not seem to be an issue but the ability of the models to learn on this small 4x4 dataset was not sufficient enough to obtain higher accuracy for either clean or noisy datasets. These models could have still benefited from early stopping to end training before the 10 epochs were complete since the model did not show signs of significant learning after the first few iterations.







6 Conclusions and Future Work

Based on these results, it could be assumed that if a QCNN model was created with the same number of trainable parameters as the full size CCNN model classifying 28x28 images, it has potential to also outperform that model in terms of robustness to noise. Due to the limitations mentioned earlier, such a QCNN cannot currently be simulated on classical processors. But the above results show promise that quantum computing methods for machine learning tasks can match or even have superior accuracy in the future compared to their classical counterparts, especially when it comes to noisy, limited, or sparse datasets. The goal for this research was not to fully optimize any model, but to compare robustness in terms of number of trainable parameters. If further optimization was implemented to all of the experimental models, both CCNN and QCNN performance metrics would likely improve. But for the purpose of this research, the results are suitable evidence to prove the hypothesis that quantum computers are more robust to noisy input data.

In future work on this project, the main priority would be to scale up the number of qubits used in the quantum circuits. This could involve running the circuits on a real quantum computer over the cloud, or waiting until research allows an efficient method to simulate more qubits on a classical processor. More qubits would allow for the use of datasets that contain higher resolution images and RGB color images. At the moment, to perform binary encoding on each pixel of a 4x4 RGB (three channel) image to qubits, it would take (4*4*3) = 48 qubits which was not feasible to simulate in a reasonable amount of time. And as the number of pixels in the images increases, the number of qubits will continue to go up. Other encoding methods could be used to reduce the number of qubits as well, which could be investigated in future research. As the development of quantum computers continues to progress, more complex circuits and datasets can be tested. Next, other methods of corruption could be added to the images and the models could be run again in order to test robustness to different types of noisy inputs. There are some open-source scripts on GitHub [23] for image corruption that include methods for generating many realistic imperfections in photos such as rain, ice, or fog. They may be seen in any photos taken outside or in situations where computer vision is used, such as the cameras on self-driving cars. Therefore testing with these corruption methods could be useful for improving the QCNNs ability to classify real-world data. Other CNN models could also be tested for both quantum and classical implementations, or more adjustments could be made to the existing models to further prevent overfitting and increase the classification accuracies. For example, early stopping could be added to the current models to end training once validation accuracy begins to drop indicating overfitting. More optimization techniques could be implemented to all models in this paper to further research the robustness with different datasets. But overall, the findings of this paper provide a baseline for experiments to compare the performance of QCNNs versus CCNNs.

References

[1] The IoT Academy. (2024, July 16). *Classical computing vs quantum computing.* https://www.theiotacademy.co/blog/classical-computing-vs-quantum-computing/

[2] Garisto, D. (2022, June 8). *What is quantum entanglement?* IEEE Spectrum. https://spectrum.ieee.org/what-is-quantum-entanglement

[3] Qiskit Community. *Quantum convolutional neural networks.* Qiskit. Retrieved January 20, 2025, from https://qiskit-community.github.io/qiskit-machine-learning/tutorials/11_quantum_convolutional_neural_networks.html

[4] Swayne, M. (2023, March 24). *What are the remaining challenges of quantum computing?* The Quantum Insider. https://thequantuminsider.com/2023/03/24/quantum-computing-challenges/

[5] Oh, S., Choi, J., & Kim, J. (2020). A tutorial on quantum convolutional neural networks (QCNN). *arXiv*. https://arxiv.org/pdf/2009.09423

[6] Khozaimi, A., & Mahmudy, W. F. (2024). *New insight in cervical cancer diagnosis using convolution neural network architecture*. arXiv. https://doi.org/10.48550/arXiv.2410.17735

[7] Gálvez Jiménez, L., & Decaestecker, C. (2024). *Impact of imperfect annotations on CNN training and performance for instance segmentation and classification in digital pathology*. arXiv. https://doi.org/10.48550/arXiv.2410.14365

[8] Chuquimarca, L., Vintimilla, B., & Velastin, S. (2024). *Classifying healthy and defective fruits with a multi-input architecture and CNN models*. arXiv. https://doi.org/10.48550/arXiv.2410.11108

[9] Wang, R., Nabney, I., & Golbabaee, M. (2024). *Efficient hyperparameter importance assessment for CNNs*. arXiv. https://doi.org/10.48550/arXiv.2410.08920

[10] Lu, J., Tan, L., & Jiang, H. (2021). Review on convolutional neural network (CNN) applied to plant leaf disease classification. *Agriculture, 11*(8), 707. https://doi.org/10.3390/agriculture11080707

[11] Ajlouni, N., Özyavaş, A., Takaoğlu, M., & et al. (2023). Medical image diagnosis based on adaptive hybrid quantum CNN. *BMC Medical Imaging, 23*, 126. https://doi.org/10.1186/s12880-023-01084-5

[12] Eswara Rao, G. V., Rajitha, B., Srinivasu, P. N., Ijaz, M. F., & Woźniak, M. (2024). Hybrid framework for respiratory lung diseases detection based on classical CNN and quantum classifiers from chest X-rays. *Biomedical Signal Processing and Control, 88*, 105567. https://doi.org/10.1016/j.bspc.2023.105567

[13] Iqbal Tomal, S. M. Y., Shafin, A. A., Afaf, A., & Bhattacharjee, D. (2024). *Quantum Convolutional Neural Network: A Hybrid Quantum-Classical Approach for Iris Dataset Classification*. arXiv. https://arxiv.org/abs/2410.16344

[14] Chen, S., & Yao, N. (2023). *Uncovering Quantum Many-body Scars with Quantum Machine Learning*. arXiv. https://arxiv.org/abs/2409.07405

[15] Mayfield, J. D., & El Naqa, I. (2024). *Evaluation of QCNN-LSTM for Disability Forecasting in Multiple Sclerosis Using Sequential Multisequence MRI*. arXiv. https://arxiv.org/abs/2401.12132

[16] Hibat-Allah, M., Gyurik, C., Coyle, B., Aboussalam, M., Romero, J., & Killoran, N. (2024). A framework for demonstrating practical quantum advantage: Comparing quantum against classical generative models. *Communications Physics, 7*(1), 68.

[17] Cong, I., Choi, S., & Lukin, M. D. (2019). Quantum convolutional neural networks. *Nature Physics, 15*(12), 1273–1278. https://doi.org/10.1038/s41567-019-0648-8

[18] Mordacci, M., Ferrari, D., & Amoretti, M. (2024). Multi-class quantum convolutional neural networks. *arXiv*. https://arxiv.org/pdf/2404.12741

[19] TensorFlow. *Fashion MNIST*. TensorFlow. Retrieved January 20, 2025, from https://www.tensorflow.org/datasets/catalog/fashion_mnist

[20] Chollet, F. (2020). *Classifying MNIST images with a convolutional neural network*. Keras. https://keras.io/examples/vision/mnist_convnet/

[21] Farhi, E., & Neven, H. (2018). Classification with quantum neural networks on near-term processors. *arXiv*. https://arxiv.org/pdf/1802.06002

[22] TensorFlow Quantum. (n.d.). *MNIST classification*. TensorFlow. Retrieved January 20, 2025, from https://www.tensorflow.org/quantum/tutorials/mnist#3_classical_neural_network

[23] Hendrycks, D. (2019). *Robustness: A repository for testing the robustness of neural networks.* GitHub. https://github.com/hendrycks/robustness/blob/master/README.md