Quantum and Hybrid Neural Network Noise Robustness

Ganesh Nanduru
Department of Computer Science
University of Virginia
Charlottesville, Virginia
bae9wk@virginia.edu

Alexi Gladstone
Department of Computer Science
University of Virginia
Charlottesville, Virginia
alexi@virginia.edu

Adwait Jog
Department of Computer Science
University of Virginia
Charlottesville, Virginia
ajog@virginia.edu

***Abstract*—Deep learning has demonstrated remarkable performance across a wide range of tasks including natural language generation, video understanding, and speech recognition. Behind this trend of increasing performance, among innovations by researchers in model architecture, training approaches, and data, is scale. The number of floating-point operations (FLOPs) used to train state-of-the-art models has increased more than 1000 fold within the last decade, resulting in an era where computation, particularly a lack of GPUs, are seen as a fundamental limitation towards training better models. Despite this increase in computation, however, modern Neural Networks (NNs) still struggle with generalization and robustness.**

Quantum computers as a computing paradigm offer solutions to both further scaling as well as generalization. Particularly, quantum computers have the ability to perform certain computations exponentially faster than classical computers, a capability deemed quantum advantage. Simultaneously, due to the inherent probabilistic nature of quantum computers, it is possible they offer advantages over classical computers in training models that generalize better and are more robust. Therefore, in this work we investigate the robustness of Quantum Neural Networks (QNNs) to noise.

Our results demonstrate that the performance of modern QNNs trained on simulators cannot compare to the performance of Classical NNs due to the inherent exponential cost per qubit of quantum simulators on classical hardware. This led us to pursue the training of Hybrid NNs, which involve both classical and quantum layers. We find that the performance of QNNs is more invariant to noise than Classical NNs, but that this trend does not hold for Hybrid NNs. We release our code publicly at https://github.com/gnanduru1/qmnist and https://github.com/gnanduru1/torchquantum.

***Index Terms*—Quantum Machine Learning, Noise Robustness, Deep Neural Networks**

I. Introduction

In 2012 the deep learning revolution began, where the field of Machine Learning (ML) became dominated by Neural Networks (NNs) due to their increasing performance [1]. Since then, the scaling of NNs has been responsible for a significant portion of increasing performance on benchmarks [2] [3] [4] [5]. This has been characterized by an exponential increase in the demand for GPUs, as well as a large focus in increasing GPU throughput, which under optimal conditions has increased up to 1000 fold in the last 8 years [6].

One common hypothesis for the future of AI is that continued scaling will bring modern algorithms to achieve broad human-level intelligence, also known as Artificial General Intelligence (AGI) [7]). One such *minimum* estimate of the amount of compute needed to achieve such a performance is $1e35$ FLOPs [8]. However, as estimates of the amount of compute used for state-of-the-art large models are currently around $1e25$ FLOPs [9], this is a 10-order magnitude increase in the number of FLOPs necessary. As Moore's law slows and is predicted to halt [10], this technical challenge of a further 10 order of magnitude increase in computation poses an opportunity for new computing paradigms other than GPUs to achieve or reduce this FLOP count. One such computing paradigm is Quantum computers.

A. Quantum Neural Networks (QNNs)

Modern NNs rely on the gradient descent and backpropagation algorithm to perform optimization with respect to a loss function. Gradient tracking as well as backpropagation compose a significant portion of the amount of FLOPs used to train NNs, while also slowing training due to increasing the amount of memory needed. Researchers have demonstrated that Quantum Neural Networks (QNNs) can achieve a near quadratic or even polynomial speedup for specific portions of optimization problems under specific assumptions [11] [12] [13] [14]. In the context of training and scaling large NNs trained on huge datasets, this offers high promise. For example, given the $1e35$ FLOPs estimate for matching the distribution of human text corpora with a language model, a quadratic reduction would bring the number of FLOPs on a quantum computer down to less than $1e18$.

Additionally, the inherent probabilistic nature of QNNs can offer benefits in specific use-cases. One troubling property of NNs has long been that they are highly nonlinear functions, resulting in several local minima in the loss landscape. In the context of optimization, this poses a challenge as it is possible for NNs to get stuck in local minima, resulting in a non-optimal solution. However, it has been shown that intentional randomization and noise can be helpful in escaping saddle points [15] [16]. Thus, it is possible the inherent probabilistic nature of QNNs could benefit in resolving this issue, as the noise from qubits being probabilistic could add randomization to observed losses.

Similarly, researchers in ML have long focused on making models more robust such as through training invariance to specific data augmentations [17] [18] [19]. One such data augmentation or invariance focus is on noise, as becoming invariant to noise is an inductive bias that often increases the generalization of models without other drawbacks [20]. Intuitively, this makes sense as an image with a limited amount of random noise generally maintains its semantic features (see Fig.4 for lower standard deviations).

II. Related Work

Modern quantum computers during the Noisy Intermediate Scale Quantum (NISQ) era of computing are limited to few tens/hundreds of qubits and suffer from high error rates [21]. Consequently, a common approach towards studying machine learning on quantum computers has been through classical simulators [22]. However, simulating quantum phenomena on classical computers requires memory that scales exponentially with the number of qubits [23]. Additionally, modern NNs rely on huge amounts of neurons to achieve high performance. Thus, the performance of pure QNNs is often low when compared to pure classical NNs.

A. Hybrid Classical-Quantum Neural Networks

One approach towards increasing the performance of Quantum Neural Networks (QNNs) trained on classical simulators has been to introduce both classical and quantum phenomena into the NN. These networks are known as Hybrid Neural Networks (HNNs). Building and training NNs in this manner allows for maintaining the scaling of classical NNs, while studying and reaping the potential benefits of QNNs. In this work, we experiment with pure QNNs, HNNs, and classical NNs.

Several existing works have explored the use of HNNs [24] [25] [26]. [24] focus on a biologically inspired implementation, using spiking neural networks [27]. Alternatively motivated by the parameter efficiency of Convolutional Neural Networks (CNNs), [26] and [25] focus on a Hybrid CNN. This involves learning weights for convolutional filters–an inductive bias we avoid due to becoming less common within recent years.

B. Noise-Robustness for Classification

Many works have focused on the affect of quantum neural networks on noise robustness. Similar to the focus of our work, [24] focus on noise-robust image classification. However, the architecture used is Spiking Neural Networks (SNNs), which differ from our usage of Articial Neural Networks (ANNs). Despite mimicking biological neurons in the brain, however, SNNs have failed to achieve the performance of ANNs on standard benchmarks. [28] proposes an entirely new approach towards increasing adversarial robustness through the inclusion of noise layers. These serve the purpose of making QNNs more invariant to noise–thus leading to more robustness to adversarial noise. Similarly, [29] focus on the affects of quantum noise on robustness in classification tasks. This work primarily focuses on the theoretical perspective, deriving bounds for adversarial attacks based off of depolarization noise in pure quantum networks.

Perhaps the most similar work to ours is in [26], where the authors explore image classification performance and adversarial robustness with hybrid convolutional architectures. Our work differs from this work in our focus on non-convolutional architectures–as convolutional architectures have become less common within the last couple of years [30]. Additionally, our work primarily focuses on invariance and generalization as a measurement for robustness, rather than robustness to adversarial attacks.

III. Motivation

Noise during the NISQ era of quantum computing is often viewed as detrimental. Additionally, qubits are inherently probabilistic, leading to challenges in creating complex deterministic quantum systems. In this work, inspired by both of these apparent "flaws," we question whether these characteristics can be used as an advantage under specific circumstances.

One of the major goals of the field of ML is generalization, aiming to ensure that models perform well on new, unseen data, a cornerstone for deploying reliable and effective systems. Specifically within the subfield of generalization is adversarial robustness. In the context of QNNs, this form of generalization has been studied extensively [24] [28] [26]. These works have demonstrated the effectiveness of QNNs/HNNs in robustness to adversarial noise when compared to classical NNs.

Tangential to the theme of robustness, it has been widely practiced within the Computer Vision (CV) community to intentionally train noise invariance into image recognition models [17] [18] [19]. Invariance can be defined as the ability of a model to maintain consistent output for input data that varies in ways irrelevant to the task, despite potential disturbances or transformation.. Therefore, training invariance into neural networks increases the probability that images captured in the wild with noise are represented in ways that match representations of images from the training distribution. Consequently, invariance often leads to better generalization capabilities [31] [32].

Seeking to investigate a form of robustness other works have not researched, and motivated by the useful characteristics of noise invariance, in this work we focus on exploring robustness through means of invariance. This involves training models to classify properly *regardless* of the different levels of noise. An example of this is shown in Fig. 4, where models would be trained to classify all elements within each row as the same digit.

One reason for the probabilistic nature of qubits improving performance can be understood intuitively when comparing the measurement of the states of qubits to dropout. Dropout in neural networks is an approach that randomly drops the usage of different neurons during training [33]. Despite the fact that dropout introduces stochasticity into neural networks, it has been successful in improving performance across a broad

range of tasks and is widely used [34]. The nature of qubits being probabilistic, and the fact that quantum measurement is a process that samples from a distribution, emulates certain characteristics of dropout. This intuition informed our decision to investigate QNN and HNN noist robustness.

IV. Implementation

We implement our Quantum models and custom dataset in Python using the TorchQuantum [35] library. We fork a version of TorchQuantum with some additional functionality at https://github.com/gnanduru1/torchquantum.

A. Image Preprocessing

To process the image into a 1D input vector, we start by performing 2D average pooling on the image with a kernel size of 6 and a stride of 6, reducing it from a 28×28 matrix to a 4×4 matrix. Pooling is typically used to downsample images for convolutional neural networks. Despite the fact that pooling reduces the image input size, it has been found to preserve the image features while decreasing the memory required to process the image [36]. This is especially helpful towards developing quantum neural networks due to the stricter memory constraints imposed by working with limited qubits. After pooling, we flatten the 4×4 matrix into a 1D input vector of length 16.

B. Models

We implement LayeredQNN and HybridQNN as TorchQuantum modules. To provide a baseline for evaluating quantum neural networks, we implement a classical neural network (ClassicalNN) as a PyTorch module.

Architecture of ClassicalNN

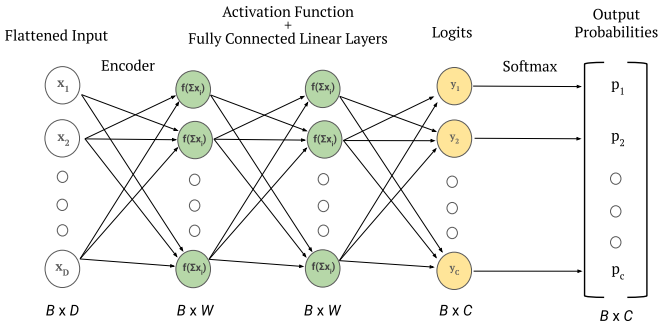

Fig. 1. ClassicalNN reflects a common implementation of a multi-layer perceptron, with a one-dimensional input vector, fully connected hidden layers, and activation functions between each layer [37]. In the diagram, B is batch size, D is the input vector size, W is the width of each intermediate hidden layer, F is the activation function applied to qubits, and C is the number of target classes.

1) Classical Model: The ClassicalNN contains exclusively linear layers with one input layer, one output layer, and a user-specified number of hidden layers. A ReLU activation function is applied after the input layer and each hidden layer. After data is passed through the output layer, output logits are passed through a softmax activation function to generate the probabilities of each target class. An example of the ClassicalNN is shown in Fig. 1.

Architecture of LayeredQNN

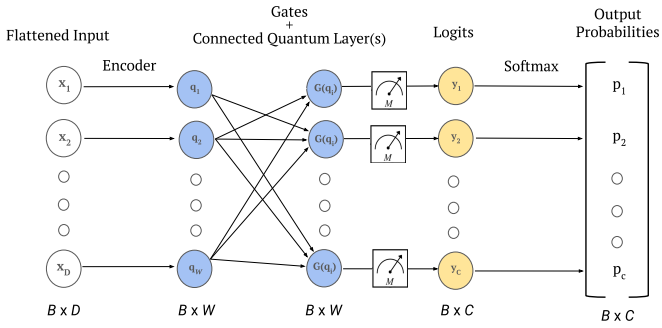

Fig. 2. We present LayeredQNN, a Quantum Neural Network implemented as a TorchQuantum module that accepts batches of 1D input vectors, encodes inputs into qubits, executes trainable gates on the qubits, and measures them into outputs, to which we apply a softmax activation function to normalize into target class probabilities. In the diagram, W is the number of wires abd G is the gate applied to qubits. The rest of the symbols have the same meaning as in Fig. 1.

2) Quantum Model: Fig. 2 displays LayeredQNN, a model with its depth and width specified by input parameters, where depth is the number of quantum layers, width is the number of wires, and each wire represents one qubit. After receiving an input vector, We use TorchQuantum's general encoder to initialize the wires. We pass the numbers from the input vector into the encoder, which applies user-specified gates to each wire with the input vector data as parameters to the gates. We use the single-qubit, single-parameter RX rotation gate as the default operation, but we include results for the RY gate in our ablation testing. Therefore, in our experiments, the input vector initializes the model's qubits by determining their phase angles on the X or Y axis.

Architecture of HybridNN

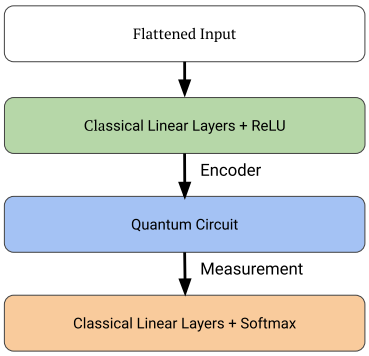

Fig. 3. The HybridNN combines classical and quantum neural networks by placing classical layers before and after a series of quantum layers. The input is processed into a classical layer, and qubit measurement values are passed through a classical layer before a softmax is applied to the output.

3) Hybrid Model: HybridQNN has three classical linear layers in total: two before the quantum layers, and one between the quantum layers and softmax activation function. HybridQNNs initializes its wires with TorchQuantum's general encoder receiving the output of the 2nd classical layer as parameters for its gates. After the wires pass through the final quantum layer, they are measured and passed as inputs into the final classical layer. An example hybrid model is shown in Fig. 3.

V. Experimental Setup

We added Gaussian noise with standard deviations spanning linearly from 0.0 to 4.5, as shown in Fig. 4. Additionally, we include visualizations of Poisson, speckle, and salt-and-pepper noise distributions in Figs. 5, 6, 7. We seed all libraries with randomness to ensure reproducibility of our results.

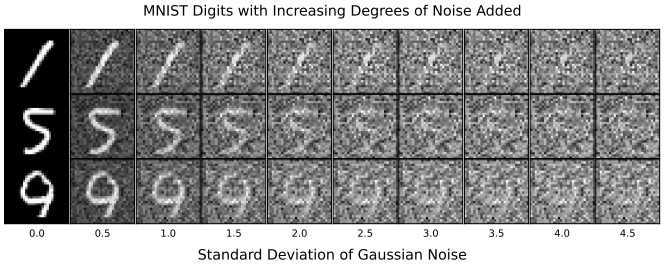

Fig. 4. We apply different degrees of Gaussian noise to each image categorized by the standard deviation of the Gaussian distribution sampled from. This is done in increments of 0.5, spanning from 0 to 4.5. Samples of the digits 1, 5, and 9 are shown.

We run a preliminary experiment to determine an optimal number of quantum layers and wires for the HybridNN. We set up a grid search with quantum layer counts of 1,2,4, and 8, and wire counts of 2, 6, 10, and assess each model on its final test accuracy. We also tried using a wire count of 14 and found this resulted in an out-of-memory error. We use the trainable RX gate in our encoder and quantum layers.

After determining which of the 16 models results in the highest test accuracy, we further verify our result by running two more dense 1-D searches ablating the number of wires and the number of quantum layers, respectively.

A. Hyperparameters

Each run is trained and evaluated on MNIST with no additional noise added. We train each model up to 50 epochs[1] with a batch size of 256 and a training/validation split ratio of $0.9 : 0.1$. We use PyTorch's Adam optimizer with a constant learning rate of 0.005 and a weight decay of 0.0001.

Our main experiment was to assess the test accuracy performance of each of our three proposed models on increasing levels of Gaussian noise added to each image. We also run experiments investigating the models' classification performance on noisy images after being trained on a dataset with no added noise. To test different quantum gates, we run an ablation test on the RX and RY gates for our quantum layers and encoder.

[1]We implement early stopping upon the model experiencing two decreases in validation accuracy in a row. We observe early stopping activates almost every run before 20 epochs.

B. Noise Distributions

To evaluate how different distributions of noise added to MNIST images affect the performance of our models, we perform ablation tests determining the effect ofGaussian Poisson, speckle, and salt-and-pepper noise on model performance.

1) Gaussian noise: To apply Gaussian noise to images, we use PyTorch to generate a tensor with the same shape as the image containing random values sampled from a Gaussian distribution. We scale this tensor by the desired standard deviation and add it to the image.

2) Speckle noise: To apply speckle noise to images, we similarly generate a tensor in the shape of the image containing Gaussian noise, but we multiply this tensor by the original image and then multiply the resulting image by the desired deviation. We finally add the generated speckle noise matrix to the original image. We show examples of speckle noise in Fig. 7.

3) Poisson noise: To apply Poisson noise to images, we use PyTorch's Poisson method, which samples a Poisson distribution using each input as the rate parameter. We scale Poisson noise by multiplying the image pixel values by a factor, adding the Poisson noise to the image, and dividing the pixel values by the same factor. We show examples of Poisson noise in Fig. 5.

4) Salt and pepper noise: To apply salt and pepper noise to images, we first choose a percentage of pixels to apply salt and pepper noise to, p. We then set $p/2$ pixels to black and $p/2$ pixels to white. We show examples of salt and pepper noise in Fig. 6.

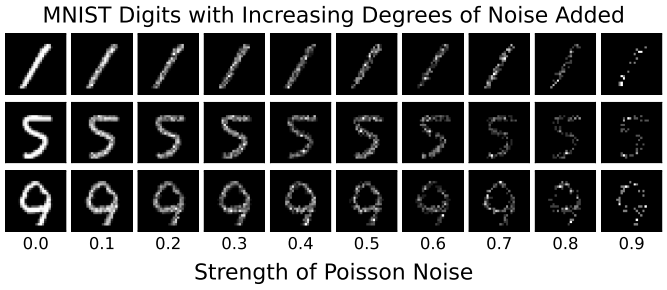

Fig. 5. We apply different degrees of Poisson noise to each image categorized by the rate of the Poisson distribution sampled from. This is done in increments of 0.1, spanning from 0 to 0.9 Samples of the digits 1, 5, and 9 are shown.

VI. Results

We observe in our grid search in Table I that the HybridNN run with 6 wires and 2 quantum layers performs with the highest test accuracy of 97.33. We observe performance is less predictable for HybridNNs with 2 wires, but results are more consistent for HybridNNs with 6 and 10 wires. Although the performance of the 6 wire and 2 layer HybridNN reached

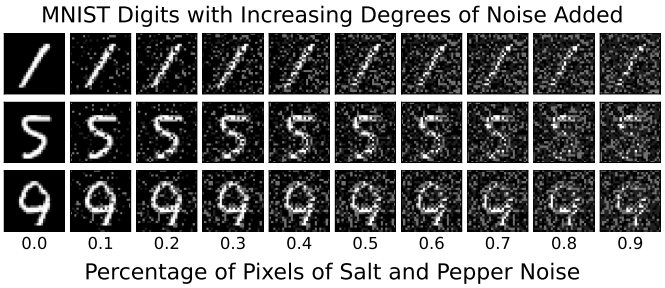

Fig. 6. We apply different degrees of salt and pepper noise to each image categorized by the percentage of pixels with the salt and pepper noise. This is done in increments of 10%, spanning from 0 to 0.9. Samples of the digits 1, 5, and 9 are shown.

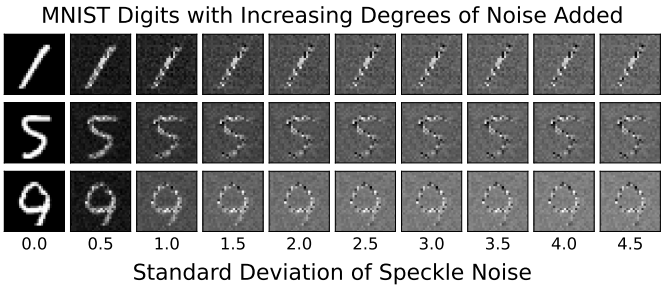

Fig. 7. We apply different degrees of Speckle noise to each image categorized by the standard deviation of the Gaussian distribution sampled from. This is done in increments of 0.5, spanning from 0 to 4.5. Samples of the digits 1, 5, and 9 are shown.

Quantum Layers	Number of Wires		
	2	6	10
1	52.0	92.0	94.7
2	85.3	**97.3**	90.7
4	57.3	93.3	96.0
8	85.3	85.3	96.0

TABLE I
GRID SEARCH OF QUANTUM LAYERS AND NUMBER OF WIRES ON HYBRIDNN TEST ACCURACY

Layers	Accuracy
1	92.0
2	**97.3**
3	94.7
4	93.3
5	**97.3**
6	94.7
7	92.0
8	94.7
9	93.3

TABLE II
TEST ACCURACY OF HYBRIDNN WITH 6 WIRES

Wires	Accuracy
2	85.3
3	88.0
4	86.7
5	93.3
6	**97.3**
7	93.3
8	90.7
9	89.3
10	90.7

TABLE III
TEST ACCURACY OF HYBRIDNN WITH 2 LAYERS

Noise	ClassicalNN		QuantumNN		HybridNN	
	Acc.	Drop	Acc.	Drop	Acc.	Drop
0	92.0	-	20.0	-	97.3	-
0.5	88.0	4.35	22.7	-13.4	92.0	5.48
1	81.3	11.6	22.7	-13.4	85.3	12.3
1.5	74.7	18.8	20.0	0.0	80.0	17.8
2	68.0	26.1	21.3	-6.6	77.3	20.5
2.5	62.7†	31.9	16.0	20.0	54.7	43.8
3	61.3	33.3	16.0	20.0	50.7	47.9
3.5	57.3	37.7	18.7	6.7	46.7	52.0
4	50.7	44.9	13.3	33.4	42.7	56.2
4.5	46.7	49.3	12.0	40.0	42.7	56.2

TABLE IV
TEST ACCURACY AND RELATIVE DROP IN PERFORMANCE OF MODELS TRAINED ON MNIST WITH INCREASING STANDARD DEVIATIONS OF GAUSSIAN NOISE. THE RELATIVE DROP IS THE MODEL'S PERFORMANCE DROP USING THE NOISE LEVEL SPECIFIED WHEN COMPARED TO THE MODEL TRAINED ON MNIST WITH NO NOISE. A NEGATIVE DROP MEANS THE MODEL PERFORMANCE INCREASED WITH THAT NOISE LEVEL. ALL RESULTS ARE SHOWN AS PERCENTAGES. † THIS POINT MARKS THE NOISE LEVEL WHERE CLASSICALNN STARTS TO PERFORM AT HIGHER ACCURACY THAN HYBRIDNN.

the highest accuracy, we observe that HybridNNs with 10 wires generally perform better on the test set, with an average accuracy of 94.3 compared to the 6-wire average accuracy of 92.0.

To confirm our results from the preliminary grid search, we run two additional 1-D grid searches, shown in Tables IIIII. Despite being at a higher granularity, these results corroborate with Table I to demonstrate that HybridNN is best implemented with 6 wires and 2 quantum layers. We observe in Table II that implementing HybridNN with 5 quantum layers and 6 wires replicates the peak accuracy of 97.3 found using 2 quantum layers and 6 wires. However, using more layers requires more memory and time resources, so we proceeded with the more efficient 2-layer, 6-wire model for the rest of our experimentation with HybridNNs.

Table IV shows that the HybridNN reaches the highest test accuracy on experiments with Gaussian noise distributions of lower than 2.5 standard deviation applied to the dataset. However, once the standard deviation reaches 2.5, ClassicalNN performs with higher test accuracy. We find the performance drop, shown in the column beside accuracy for each model, is steeper for HybridNN and ClassicalNN. Aligned with our expectations of QNNs having improved performance due to being probabilistic, we find QNNs to experience the lowest drop in performance after heavier amounts of Gaussian noise is added to the dataset. These results are also visualized in Fig. 8.

When evaluating the models on noisy test sets after removing the noise during training in Table V and Fig. 9, we find the trends observed in Table IV and Fig. 8 are amplified. HybridNN drops 84.9% in relative accuracy compared to the initial test accuracy on the noiseless test set. ClassicalNN, however, drops 63.2% in relative accuracy compared to its initial result. While both of these drops are more extreme than in Table IV, the final test accuracy of HybridNN on the noisiest test set is 14.7%, much lower than the final test accuracy of ClassicalNN, which is 33.3%.

We display a head-to-head comparison of HybridNN with and without noise applied to the training set in Fig. 10. HybridNN consistently performs with higher test accuracy

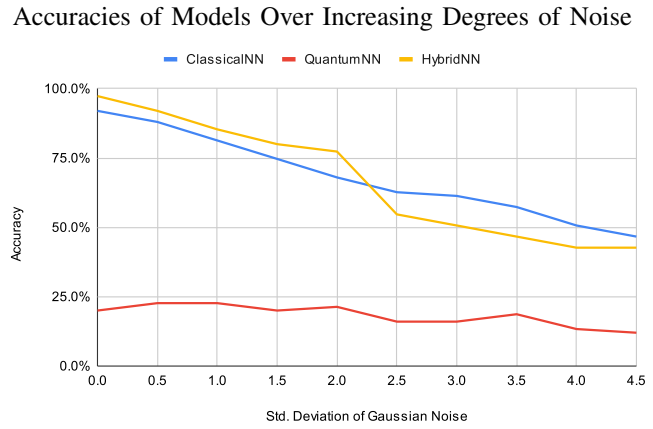

Fig. 8. Visualization of results from Table IV. The x-axis represents standard deviation of Gaussian noise and the y-axis represents final test accuracy score.

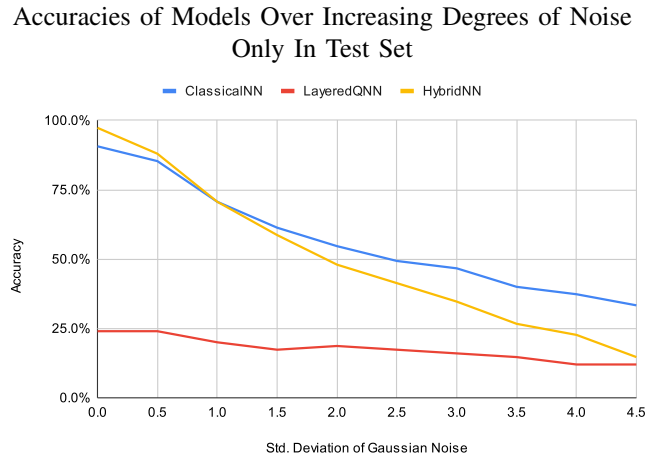

Fig. 9. We provide a visualization of the results from Table V. The x-axis represents the standard deviation of Gaussian noise added only to the test set images, and the y-axis represents test set classification accuracy.

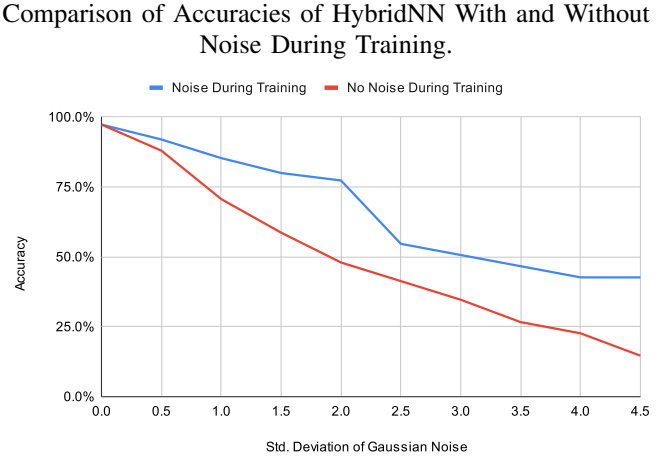

Fig. 10. The blue line represents HybridNN trained with noise applied to both the training and testing set. The red line represents HybridNN trained with noise only applied to the testing set.

Noise	ClassicalNN		QuantumNN		HybridNN	
	Acc.	Drop	Acc.	Drop	Acc.	Drop
0	90.7	-	24.0	-	97.3	-
0.5	85.3	5.9	24.0	0.0	88.0	9.6
1	70.7	22.1	20.0	16.7	70.7	27.4
1.5	61.3	32.4	17.3	27.8	58.7	39.7
2	54.7	39.7	18.7	22.2	48.0	50.7
2.5	49.3	45.6	17.3	27.8	41.3	57.5
3	46.7	48.5	16.0	33.3	34.7	64.4
3.5	40.0	55.9	14.7	38.9	26.7	72.6
4	37.3	58.8	12.0	50.0	22.7	76.7
4.5	33.3	63.2	12.0	50.0	14.7	84.9

TABLE V
IN THIS EVALUATION, WE TRAIN WITHOUT NOISE AND IMPLEMENT NOISE ONLY DURING TESTING.

when it is trained on noisy data.

Noise	RX	RY
0	97.3	93.3
0.5	92.0	92.0
1	85.3	82.7
1.5	80.0	76.0
2	77.3	42.7
2.5	54.7	60.0
3	50.7	48.0
3.5	46.7	46.7
4	42.7	42.7
4.5	42.7	44.0

TABLE VI
TEST ACCURACY OF HYBRIDNN WITH DIFFERENT ENCODER GATES

In Table VI, we perform an ablation test assessing the effectiveness of the RX, and RY gates in the encoder and quantum layers of HybridNN. We find using the RX gate generally yields higher test accuracy than using the RY gate.

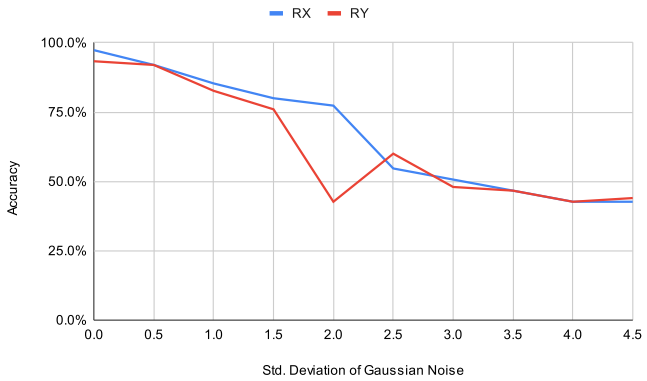

Fig. 11. Visualization of the results of HybridNN using different quantum gates from Table VI. The x-axis represents the standard deviation of Gaussian noise and the y-axis represents test set classification accuracy. As shown in the legend, the curve for RX is in blue and the curve for RY is in red.

In Fig. 11, we visualize the gate ablations from Table VI. We display the comparable test accuracy of HybridNN using the RX and RY gates, aside from a drop in the RY-gate model when the standard deviation of Gaussian noise equals 2.0.

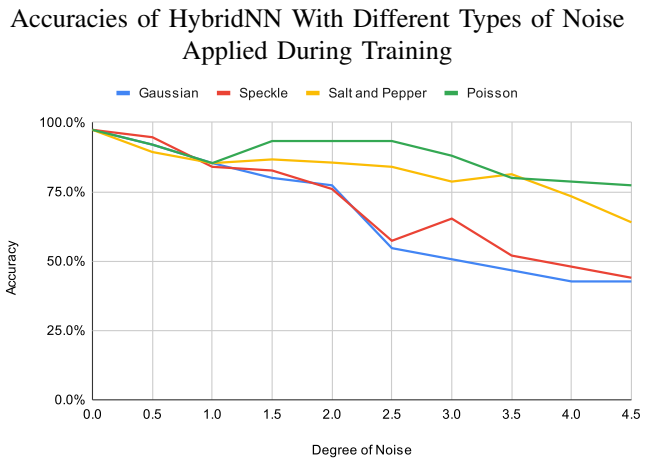

Fig. 12. We report the test accuracies of HybridNN on different types and degrees of noise added. For Gaussian and speckle noise, the degree of noise corresponds to the standard deviation of Gaussian noise used to perturb the image. For salt and pepper and Poisson noise, the degree of noise corresponds to $5x$ the proportion of image pixels affected. For Poisson noise, the degree of noise corresponds to the $5x$ the strength of Poisson noise. We present samples for all types of noise and all degrees of noises in Figs. 4 5, 6, 7

In Fig. 12, we present four experiments corresponding to each noise variation we identified in Section V-B. We observe Gaussian and speckle noise cause the biggest drops in test accuracy as the degree of noise increases, and salt and pepper and Poisson noise cause smaller, but comparable drops in test accuracy as the degree of noise increases.

VII. Discussion and Limitations

A. Discussion

Consistent with previous works, the results in Fig. 8 and Fig. 9 demonstrate that QNNs have an advantage in robustness to noise when compared to Classical NNs. This is apparent through a lower degradation across noise levels–demonstrating better scalability. Remarkably, the performance of QNNs in Fig. 8 and Fig. 9 is starkly similar, despite one set of results being trained on noisy samples while the other was not. This suggests that the properties of QNNs may favor an inherent invariance towards noise during training–as a lack of training on noisy examples did not worsen the scaling of QNNs across noise levels significantly.

Although existing works have found that HNNs can increase robustness to adversarial noise, we found this trend does not occur with invariance to noise, as demonstrated in Figs. 89. One hypothesis for why this may be the case is that invariance to noise can be seen as a stronger requirement than adversarial noise–meaning that the reliance on quantum phenomena to increase performance must be higher when studying invariance than in HNNs for existing works studying adversarial robustness. This could explain why pure QNNs scale well to different levels of noise but HNNs do not in our results.

In Fig. 12, we display the performance dropoff of the HybridNN as the degree of noise increases for four different types of noise. We find that Gaussian noise leads to the steepest decrease in accuracy, closely followed by speckle noise. Salt and pepper noise decreases the accuracy less, and the Poisson distribution decreases the test accuracy the least when added to the image dataset. We find these results consistent with the visual examples of noise perturbations in Fig. 4, Fig. 5, Fig. 6, and Fig. 7. While we may scale Poisson noise to a high degree, it mainly acts to obscure the existing white pixels of the image, and does not transform the surrounding black pixels. This visual difference may enable the model to be more invariant to Poisson noise, as it will not be affected by perturbations to the black pixels surrounding the digits that the other three forms of noise make. While salt and pepper noise significantly obscures most of the images, it doesn't affect every pixel. Gaussian and speckle noise distributions, on the other hand, are added to every single image pixel. We believe HybridNN's test accuracy experiences the steepest drops on these two types of noise because Gaussian and speckle noise affect the most pixels and visibly obscure the digits more than Poisson and salt and pepper noise.

B. Limitations

During the NISQ era, quantum computers are limited in the number of qubits and have high error rates. Consequentially, one of the primary techniques used to study the behavior of quantum phenomena on different algorithms are quantum simulators running on classical computers. This approach was leveraged in this paper, and as such one limitation of our work is the caveats brought about by a simulator. One of the major limitations of classical simulators for simulating quantum phenomena is that the memory usage is exponential with respect to the number of qubits. This limited the model size at which we could conduct experiments.

Additionally, our experiments use an architecture that is not the current state-of-the-art transformer model [38]. We instead focus on Multilayer Perceptron (MLP) variants as MLPs form the basis of almost all widely used modern architectures in deep learning and are more versatile. A possible future work could investigate if our findings extend to the transformer architecture.

VIII. Conclusion

In this work, we investigate the effect of quantum phenomena on the noise invariance of NNs as a form of robustness. To achieve this, we train Quantum Neural Networks (QNNs), Hybrid Neural Networks (HNNs) involving both classical and quantum layers, and Classical Neural Networks to recognize hand-written digits from the MNIST dataset. Naturally, the performance of Classical NNs monotonically decreases for increasing levels of noise. However, the performance of QNNs decreases slower than the ClassicalNN for higher levels of noise. This offers insight into the benefits of QNNs being more invariant to noise due to the probabilistic nature of qubits. However, due to the non-scalability of modern quantum simulators and the limited amount of qubits in modern NISQ computers, the performance of Classical NNs is much higher than QNNs. To achieve a more fair comparison, we also

compare the performance of HNNs to Classical NNs in noise invariance, but find that HNNs are not more robust to noise compared to Classical NNs or QNNs.

REFERENCES

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

[2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[3] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[5] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[6] Mar. 2024. [Online]. Available: https://www.nextbigfuture.com/2024/03/nvidia-increased-compute-power-1000x-in-8-years-to-20-petaflops-in-the-blackwell-gpu.html

[7] D. Patel, "Will scaling work?" Dec. 2023. [Online]. Available: https://www.dwarkeshpatel.com/p/will-scaling-work

[8] M. Barnett, "The direct approach," Apr. 2023. [Online]. Available: https://epochai.org/blog/the-direct-approach

[9] [Online]. Available: https://twitter.com/karpathy/status/1781047292486914189

[10] G. Chirkov and D. Wentzlaff, "Seizing the bandwidth scaling of on-package interconnect in a post-moore's law world," in *Proceedings of the 37th International Conference on Supercomputing*, 2023, pp. 410–422.

[11] C.-M. Alexandru, E. Bridgett-Tomkinson, N. Linden, J. MacManus, A. Montanaro, and H. Morris, "Quantum speedups of some general-purpose numerical optimisation algorithms," *Quantum Science and Technology*, vol. 5, no. 4, p. 045014, 2020.

[12] L. Wossnig, S. Tschiatschek, and S. Zohren, "Quantum-classical truncated newton method for high-dimensional energy landscapes," *arXiv: Quantum Physics*, 2017.

[13] I. Kerenidis and A. Prakash, "Quantum gradient descent for linear systems and least squares," *Physical Review A*, 2017.

[14] P. Rebentrost, M. Schuld, L. Wossnig, F. Petruccione, and S. Lloyd, "Quantum gradient descent and newton's method for constrained polynomial optimization," *New Journal of Physics*, vol. 21, 2016.

[15] H. Daneshmand, J. Kohler, A. Lucchi, and T. Hofmann, "Escaping saddles with stochastic gradients," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1155–1164.

[16] Y. Fang, Z. Yu, and F. Chen, "Noise helps optimization escape from saddle points in the neural dynamics." in *ESANN*, 2019.

[17] A. Bardes, J. Ponce, and Y. LeCun, "Vicreg: Variance-invariance-covariance regularization for self-supervised learning," *arXiv preprint arXiv:2105.04906*, 2021.

[18] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 9650–9660.

[19] B. Bloem-Reddy, Y. Whye *et al.*, "Probabilistic symmetries and invariant neural networks," *Journal of Machine Learning Research*, vol. 21, no. 90, pp. 1–61, 2020.

[20] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[21] S. S. Tannu and M. K. Qureshi, "Mitigating measurement errors in quantum computers by exploiting state-dependent bias," in *Proceedings of the 52nd annual IEEE/ACM international symposium on microarchitecture*, 2019, pp. 279–290.

[22] 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.

[23] K. Young, M. Scese, and A. Ebnenasir, "Simulating quantum computations on classical machines: A survey," *arXiv preprint arXiv:2311.16505*, 2023.

[24] D. Konar, A. D. Sarma, S. Bhandary, S. Bhattacharyya, A. Cangi, and V. Aggarwal, "A shallow hybrid classical–quantum spiking feedforward neural network for noise-robust image classification," *Applied Soft Computing*, vol. 136, p. 110099, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494623001175

[25] J. Liu, K. H. Lim, K. L. Wood, W. Huang, C. Guo, and H.-L. Huang, "Hybrid quantum-classical convolutional neural networks," *Science China Physics, Mechanics & Astronomy*, vol. 64, no. 9, p. 290311, 2021.

[26] S.-Y. Huang, W.-J. An, D.-S. Zhang, and N.-R. Zhou, "Image classification and adversarial robustness analysis based on hybrid quantum–classical convolutional neural network," *Optics Communications*, vol. 533, p. 129287, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0030401823000329

[27] S. Ghosh-Dastidar and H. Adeli, "Spiking neural networks," *International journal of neural systems*, vol. 19, no. 04, pp. 295–308, 2009.

[28] C. Huang and S. Zhang, "Enhancing adversarial robustness of quantum neural networks by adding noise layers," *New Journal of Physics*, vol. 25, no. 8, p. 083019, 2023.

[29] Y. Du, M.-H. Hsieh, T. Liu, D. Tao, and N. Liu, "Quantum noise protects quantum classifiers against adversaries," *Physical Review Research*, vol. 3, no. 2, p. 023153, 2021.

[30] Y. Liu, Y. Zhang, Y. Wang, F. Hou, J. Yuan, J. Tian, Y. Zhang, Z. Shi, J. Fan, and Z. He, "A survey of visual transformers," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[31] X. Zhu, Q. Wang, P. Li, X. Zhang, and L. Wang, "Learning region-wise deep feature representation for image analysis," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–10, 2018.

[32] L. Ericsson, H. Gouk, and T. M. Hospedales, "Why do self-supervised models transfer? investigating the impact of invariance on downstream tasks," *ArXiv*, vol. abs/2111.11398, 2021.

[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[34] A. Labach, H. Salehinejad, and S. Valaee, "Survey of dropout methods for deep neural networks," *arXiv preprint arXiv:1904.13310*, 2019.

[35] H. Wang, Y. Ding, J. Gu, Z. Li, Y. Lin, D. Z. Pan, F. T. Chong, and S. Han, "Quantumnas: Noise-adaptive search for robust quantum circuits," in *The 28th IEEE International Symposium on High-Performance Computer Architecture (HPCA-28)*, 2022.

[36] F. Bieder, R. Sandkühler, and P. C. Cattin, "Comparison of methods generalizing max- and average-pooling," 2021.

[37] E. Herberg, "Lecture notes: Neural network architectures," 2023.

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.