algorithms

MDPI

Article

Improving 2–5 Qubit Quantum Phase Estimation Circuits Using Machine Learning

Charles Woodrum [1,*], Torrey Wagner [1] and David Weeks [2]

[1] Data Analytics Certificate Program, Graduate School of Engineering and Management, Air Force Institute of Technology, Wright-Patterson AFB, OH 45433, USA; torrey.wagner.2@us.af.mil
[2] Department of Engineering Physics, Graduate School of Engineering and Management, Air Force Institute of Technology, Wright-Patterson AFB, OH 45433, USA; david.weeks@afit.edu
* Correspondence: woodrum.27@osu.edu

Abstract: Quantum computing has the potential to solve problems that are currently intractable to classical computers with algorithms like Quantum Phase Estimation (QPE); however, noise significantly hinders the performance of today's quantum computers. Machine learning has the potential to improve the performance of QPE algorithms, especially in the presence of noise. In this work, QPE circuits were simulated with varying levels of depolarizing noise to generate datasets of QPE output. In each case, the phase being estimated was generated with a phase gate, and each circuit modeled was defined by a randomly selected phase. The model accuracy, prediction speed, overfitting level and variation in accuracy with noise level was determined for 5 machine learning algorithms. These attributes were compared to the traditional method of post-processing and a 6x–36 improvement in model performance was noted, depending on the dataset. No algorithm was a clear winner when considering these 4 criteria, as the lowest-error model (neural network) was also the slowest predictor; the algorithm with the lowest overfitting and fastest prediction time (linear regression) had the highest error level and a high degree of variation of error with noise. The XGBoost ensemble algorithm was judged to be the best tradeoff between these criteria due to its error level, prediction time and low variation of error with noise. For the first time, a machine learning model was validated using a 2-qubit datapoint obtained from an IBMQ quantum computer. The best 2-qubit model predicted within 2% of the actual phase, while the traditional method possessed a 25% error.

Keywords: quantum computing; ensemble models; neural network; quantum phase estimation; post-processing; measurement interpretation

Citation: Woodrum, C.; Wagner, T.; Weeks, D. Improving 2–5 Qubit Quantum Phase Estimation Circuits Using Machine Learning. *Algorithms* **2024**, *17*, 214. https://doi.org/10.3390/a17050214

Academic Editors: Hua-Lei Yin and Nan-Run Zhou

Received: 15 March 2024
Revised: 10 May 2024
Accepted: 11 May 2024
Published: 15 May 2024

1. Introduction

Quantum Phase Estimation (QPE) is a quantum computing algorithm that estimates the phase of an eigenvalue or of multiple eigenvalues of a unitary operator U. To further understand QPE and the problem addressed in this paper, we will first explore some of the mathematical background of quantum mechanics and the derivation of the QPE algorithm [1–4]. Quantum mechanics describes physical particles or groups of particles using states (usually denoted with a "ket" $|\psi\rangle$) in a Hilbert space. For this application of quantum computing, the Hilbert space describes particles or groups of particles with states in a Hilbert space spanned by two basis states, $|0\rangle$ or $|1\rangle$. In this case, the particle or group of particles is called a "qubit", and a general qubit state is denoted $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ with complex α, complex β and $|\alpha|^2 + |\beta|^2 = 1$. When a qubit's state is measured, the wavefunction collapses into one of the basis states with probability $|\langle\phi|\psi\rangle|^2$, where the "bra" $\langle\phi| = \langle 0|$ or $\langle\phi| = \langle 1|$, with each bra being an element in the dual space to the space in which the ket $|\psi\rangle$ lies.

Changing the state of a qubit within the standard QPE circuit is mathematically represented by acting on the state of the qubit with a unitary operator, which preserves the sum

of squares of all vector components. Unitary operators are operators in the Hilbert space such that $UU^\dagger = U^\dagger U = I$, where $U^\dagger$ is the conjugate-transpose of the operator and I is the identity operator. Measurement of a qubit is a non-unitary operation, though, in this context, it is performed at the end of the circuit and is described probabilistically. While not incorporated in this simulation of QPE circuits, another application of non-unitary operations includes circuit noise analysis associated with open system dynamics [4]. It is often necessary to work with composite systems representing more than one quantum object. In this case, basis states of the whole system, $|\psi\rangle$, are Kroncker products (a specialization of finite-dimensional tensor products) of the individual basis states, denoted, for example, with $|\psi\rangle = |0\rangle \otimes |0\rangle \otimes |0\rangle$ or simply as $|0\rangle|0\rangle|0\rangle$ or $|000\rangle$. Operators on the composite state are also Kronecker products of individual operators, so $U\,|\psi\rangle = U_1 \otimes U_2|00\rangle = U_1|0\rangle \otimes U_2|0\rangle$. We can re-express the qubit state written as a series of 0's and 1's into a decimal number by assigning an order to the qubits, then converting the binary string into a decimal number. Thus, we could take the string of qubits $q_1,\ q_2,\ \ldots, q_n$ and create the decimal number $x = \sum_{k=1}^{n} 2^{k-1} q_k$. Most of the paper deals with results written in this decimal form.

The QPE algorithm is used in a number of quantum codes and determines the eigenvalues of a general unitary operator U with corresponding eigenvectors, $|u\rangle$. The eigenvalues have the form $e^{2\pi i\theta}$, where $\theta \in [0, 1)$ is the "phase" of the unitary operator. In this paper, we will deal with a restricted situation where we only have one phase to estimate for each unitary operator.

In order to estimate the phase of some unitary operator, we must begin with a circuit with N qubits split into a first and second register with n and m qubits, respectively. Initially, all qubits in the first register are set to the $|0\rangle$ state and the qubits in the second register are set to an eigenvector of the unitary operator $|u\rangle_2$, where the subscript 2 denotes the fact that this ket refers to the state of the second register. This leads to an initial state of $|\psi_1\rangle = |0\rangle^{\otimes n} \otimes |u\rangle_2$. We apply a Hadamard gate H to each qubit in the first register to yield the state in Equation (2):

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \qquad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{1}$$

$$|\psi_2\rangle = 2^{-\frac{n}{2}}(|0\rangle + |1\rangle)^{\otimes n} \otimes |u\rangle_2 \tag{2}$$

Following this operation, we apply a controlled unitary operation 2^{j-1} times for the j^{th} qubit in the first register. A controlled unitary operation, denoted CU, will apply the unitary operation to the second register if the control qubit is $|1\rangle$ and not apply it if the qubit is $|0\rangle$, with the result shown in Equation (3). This concludes the operations that involve the second register.

$$\begin{aligned} |\psi_3\rangle &= 2^{-\frac{n}{2}} \left\{ \bigotimes_{j=1}^{n} CU^{2^{j-1}} (|0\rangle + |1\rangle) \right\} |u\rangle_2 \\ &= 2^{-\frac{n}{2}} \left\{ \bigotimes_{j=1}^{n} \left(|0\rangle + |1\rangle\, U^{2^{j-1}} \right) \right\} |u\rangle_2 \\ &= 2^{-\frac{n}{2}} \left\{ \bigotimes_{j=1}^{n} \left(|0\rangle + e^{2\pi i \theta 2^{j-1}} |1\rangle \right) \right\} |u\rangle_2 \\ &= 2^{-\frac{n}{2}} \sum_{k=0}^{2^n - 1} e^{2\pi i k\theta} |k\rangle \otimes |u\rangle_2 \end{aligned} \tag{3}$$

The next step in the derivation is to apply the inverse quantum Fourier transform $F^\dagger$. The $F^\dagger$ is based on the inverse discrete Fourier transform (IDFT), which takes a (normalized) vector of complex numbers, say $x_0,\ x_1, \ldots,\ x_{N-1}$, and outputs another vector of complex numbers $y_0,\ y_1, \ldots,\ y_{N-1}$, where $y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{-2\pi i j k / N}$. The $F^\dagger$ takes the quantum state in the numerical basis and transforms it to a superposition of all states in the numerical basis with the coefficient of each basis state in the transformation being the IDFT value. The effect of $F^\dagger$ is to transform the n qubit state $|j\rangle$ to the state $F^\dagger|j\rangle$ shown in Equation (4).

The $F^\dagger$ operator is applied to the first register to obtain the final state of the circuit $|\psi_4\rangle$ before measurement shown in Equation (5):

$$F^\dagger|j\rangle = 2^{-n/2}\sum_{x=0}^{2^n-1} e^{-2\pi ijx/2^n}|x\rangle \tag{4}$$

$$|\psi_4\rangle = F^\dagger|\psi_3\rangle = 2^{-n}\sum_{k=0}^{2^n-1}\sum_{x=0}^{2^n-1} e^{-2\pi ik2^{-n}(x-2^n\theta)}\Big|x\Big\rangle \otimes |u\rangle_2 \tag{5}$$

Measurement in QPE is performed only on the first register. Each of the qubits are measured to produce a string of bits, which is then converted into a decimal number between 0 and 2^{n-1}. The probability of measuring a given number x is the probability of measuring the state $|x\rangle$ in the $|\psi_4\rangle$ equation above, and the probability of such a measurement is the modulus squared of the coefficient of state $|x\rangle$. Thus, the probability of x given θ, denoted $P(x|\theta) = |\langle x|\psi_4\rangle|^2$, is shown in Equation (6):

$$P(x|\theta) = \left|2^{-n}\sum_{k=0}^{2^n-1}\sum_{x'=0}^{2^n-1} e^{-2\pi ik2^{-n}(x'-2^n\theta)}\langle x|x'\rangle\right|^2 = \left|2^{-n}\sum_{k=0}^{2^n-1} e^{-2\pi ik2^{-n}(x-2^n\theta)}\right|^2 \tag{6}$$

When θ can be represented exactly by a decimal string of n qubits (say $0.\theta_1\theta_2\cdots\theta_n$), then this probability distribution $P(x|\theta)$ becomes 1 for the numerical value associated with the string of bits representing θ (in this case $P\left(\sum_{d=1}^{n}\theta_d 2^d\middle|\theta\right) = 1$ with 0 otherwise). However, when this is not possible (like for $\theta = 1/3$), $P(x|\theta)$ will have non-zero values for all x. An example of the final circuit diagram is shown in Figure 1, which depicts a 3-qubit QPE circuit diagram where the unitary operator is a phase gate corresponding to $\theta = 1/3$ in the probability distribution. Note that an X gate is applied to the second register (q3) to put the second register into an eigenstate of the phase gate. For validation of this work, a 2-qubit version of this circuit was run on the IBM Perth quantum computer.

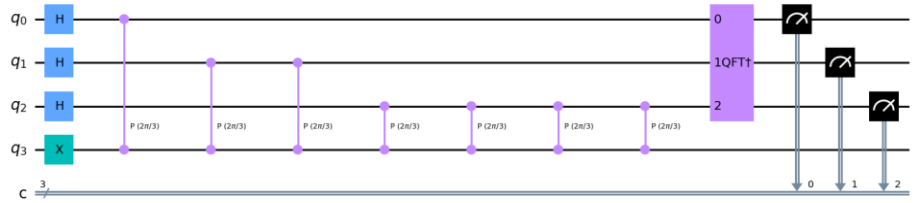

Figure 1. A circuit diagram for a 3-qubit QPE circuit where the unitary operator is a phase gate with phase 2/3—this corresponds to a $\theta = 1/3$ in the probability distribution.

The process of doing QPE is akin to sampling from $P(x|\theta)$ with $x \in \{0, 1, \ldots, 2^n - 1\}$ and using the results to estimate the exact value of θ. The main difficulty with performing QPE on today's quantum devices is that present-day devices are plagued by noise in spite of mitigation attempts [5–7], which adds an unknown term to the distribution. The process of performing QPE on today's devices is similar to sampling from another distribution $\widetilde{P}(x|\theta) = \left|2^{-n}\sum_{k=0}^{2^n-1} e^{-2\pi ik2^{-n}(x-2^n\theta)}\right|^2 + \epsilon(x)$, where $\widetilde{P}(x|\theta)$ is a proper probability distribution, but the form and effect of $\epsilon(x)$ is not known with certainty. It is possible to recreate some types of noise in simulations of quantum circuits, and one type of noise, depolarizing noise, is added in this paper. Examples of $\widetilde{P}(x|\theta)$ are found in Section 2.2.

The datasets were created with 21 levels of depolarizing noise so that the model performance on a variety of noise could be evaluated. The researchers sought to have a single, tunable noise parameter for circuit generation, and depolarizing noise was straightforward to implement. Levels of noise from 0 to 0.2 in steps of 0.01 were chosen to represent the range of noise realistically possible, from ideal at 0 to extremely high at 0.2. It is worth noting that this choice was mostly for practical purposes, since it is computationally expensive

to perform these simulations. Someone could, however, produce more data with finer than 0.01 steps in noise and go beyond a noise level of 0.2.

In the training of the models, the level of noise will be uncertain since the quantum phase is estimated by a model, and it is desirable for a model to have consistent predictive performance regardless of the noise level. An example of this is shown in Figure 2, where the left panel shows a model whose performance varies with noise, and the right panel shows mostly level performance.

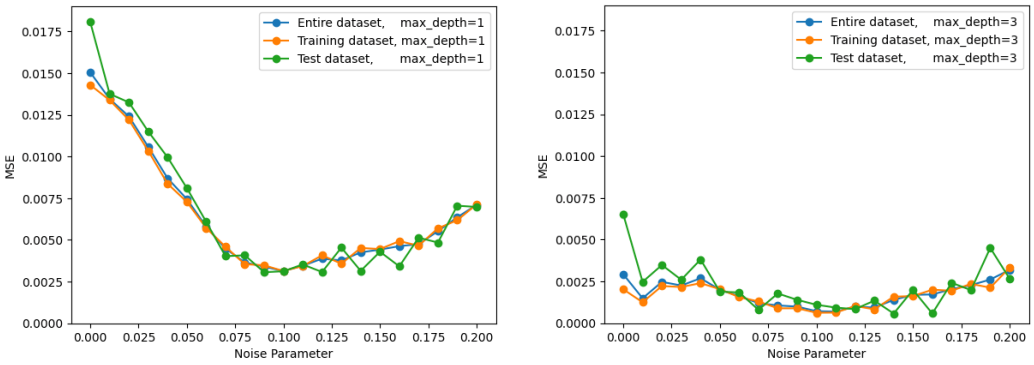

Figure 2. Example of undesired model performance that varies with the level of simulated noise (**left**) and desired level model performance with the level of simulated noise (**right**). The model metric in the *y*-axis is the mean squared error (MSE), and these examples use the XGBoost algorithm with different levels of the maximum depth parameter.

1.1. Data Understanding

The phases of data understanding, data preparation, modeling and evaluation from the cross-industry standard process for data mining (CRISP-DM) were followed in this analysis [8]. The first phase seeks to understand the datasets analyzed in this work. The datasets were generated using a simulated QPE circuit created using IBM's qiskit, for 2-, 3-, 4- and 5-qubit circuits in a process described in Sections 2.1 and 2.2. In a circuit with n qubits in the first register, we measure n qubits once the QPE algorithm described in the introduction is implemented. Each qubit is measured as a 0 or 1. Since there are n of these qubits, we can measure 2^n possible outputs. When this sequence of n qubits is considered as a binary number, it can be converted into a decimal number ranging from 0 if all 0's are measured, to $2^n - 1$ if all 1's are measured. A single "shot" of the circuit, corresponding to one sampling from the probability distribution described in the Introduction, is just one of these values in the range 0 to $2^n - 1$. The same circuit is run repeatedly to sample many times from the probability distribution. An example for a 2-qubit circuit is presented: if we measure it 400 times and 00 is measured 100 times, 01 is measured 200 times, 10 is measured 50 times, and 11 is measured 50 times, then the feature 00 takes on the value 100/400 = 0.25. In a similar manner, feature 01 takes on the value 200/400 = 0.5, and features 10 and 11 take on the value 50/400 = 0.125. Therefore, the vector of features representing one data point would be [0.25, 0.5, 0.125, 0.125], and we would hope to derive an estimation of the phase from this vector of four features. The process is analogous in systems of higher qubits, and the number of features grows like 2^n. In this experiment, we have generated a dataset with these feature vectors for many different circuits with some associated phase and recorded the noise parameter (described in Section 2.2) and the phase in order to train the model.

Each row of the dataset contains the transmitted phase, the added noise level and the features that contain the quantum information. For a 2-qubit system there are four features, and the 3/4/5 qubit systems have 8/16/32 features, respectively. For the 2-qubit dataset, histograms with kernel density estimate overlay are presented in Figure 3 for the transmitted phase (left) and induced noise level (right). Uniform distributions of phase and noise were expected as the transmitted phase was randomly generated, and data were generated uniformly for 21 different noise levels. If these values were unequally distributed,

it would be difficult to verify that model performance was consistent across their range. The datasets for the 2-qubit and 5-qubit systems contained 210,000 rows, the 3-qubit dataset contained 52,500 rows, and the 4-qubit dataset contained 105,000 rows.

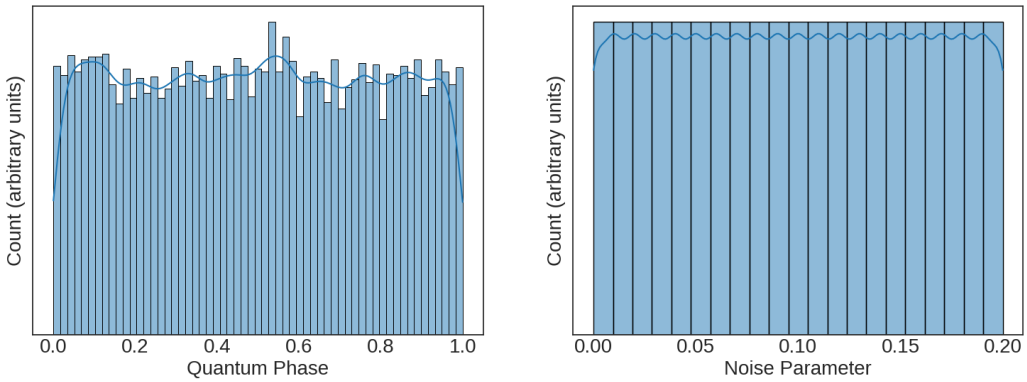

Figure 3. Histogram with kernel density estimate overlay for the transmitted phase (**left**) and induced noise level (**right**) in the 2-qubit dataset. The *y*-axis for both subpanels is the relative count in each vertical bar of the histogram.

It is expected that the distributions of each channel in an n-qubit system will be similar, and Figure 4 confirms this, showing histograms of each channel for the 2- and 3-qubit systems. Verifying this expectation in Figure 4 increased confidence in the integrity of the large datasets. The non-normal distributions are the expected result of the QPE probability (Equation (6)) and the injected noise levels. In each sub-panel of Figure 3, the non-normal distributions are explained as follows:

- Values near 0 should occur most often, while values close to 1 only occur when the phase exactly lines up with a value of $k/2^n$, where k is an integer between 0 and $2^n - 1$.
- We expect a small peak at 0.405 since this corresponds to a value of the phase close to $(k \pm 0.5)/2^n$, where there are 2 peaks of equal size near 0.405 rather than one largest peak, as is the case when θ is not close to $(k \pm 0.5)/2^n$.
- The 4- and 5-qubit system dataset histograms are similar to Figures 3 and 4.

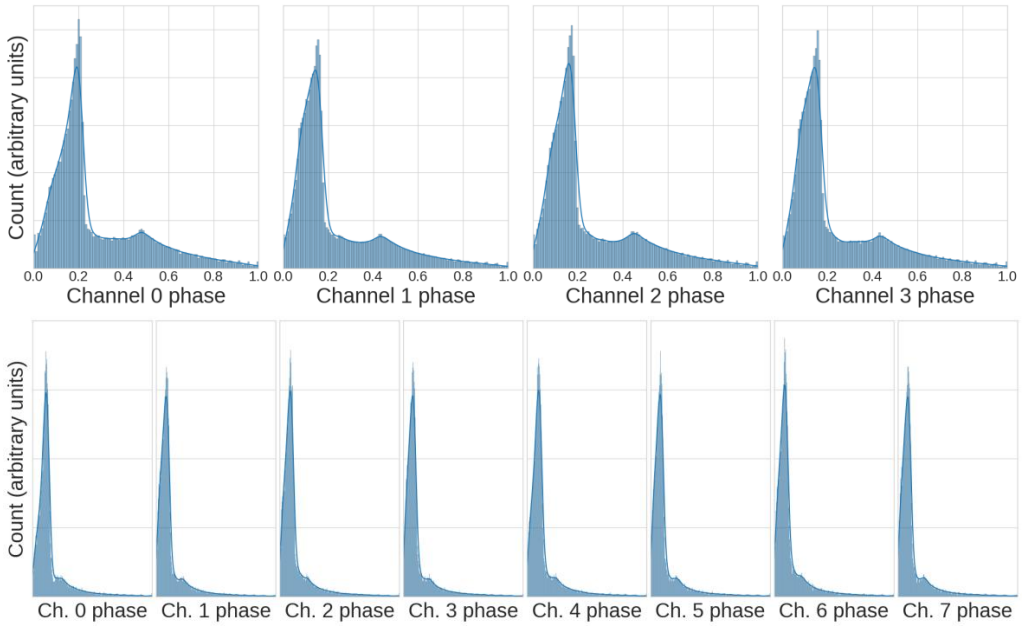

Figure 4. Histogram with kernel density estimate overlay for the four channels of the 2-qubit dataset (**top** row) and for the 3-qubit dataset (**bottom** row).

1.2. Literature

In the literature, researchers have explored measurement interpretation as an alternative to the highest-peak method [9] and using machine learning for noise prediction [10]. The present work is the first instance of machine learning being applied to the problem of interpreting quantum phase estimation measurement output, other than preliminary work performed by the author with a less sophisticated noise model [11]. The goal of the present work is to find a method for QPE that improves upon the traditional, highest-peak method of estimation.

2. Materials and Methods

The code for the models was prepared and executed within a Python 3.10.12 environment, including sklearn 1.2.2 and keras 2.13.1. Each of the 2/3/4/5-qubit datasets were divided into an 80% training set and a 20% test/holdout set. For neural network algorithms, the training set was further subdivided into 80% training and 20% validation sets. The methods described in this section include the quantum circuit modeling, generation of the dataset, noise generation, details on each of the algorithms and a speed-of-prediction analysis. The methods are summarized in the pseudocode below:

- Define the quantum circuit
  - Select phase values uniformly between [0, 2π) for generating circuits.
  - Create qiskit QPE circuits with qubits = [2,3,4,5] in the first register and 1 qubit in the second register.
  - Specify the unitary operator and eigenvector.
- Define noise and create datasets
  - Introduce depolarizing noise with probability p using qiskit NoiseModel.
  - Simulate circuits with 21 noise levels p = [0.00, 0.01, 0.02, . . ., 0.20].
- For dataset in qubits = [2,3,4,5]:
  - Create and split datasets with phase, noise level and features for each quantum circuit simulated.
  - For algorithm = [linear regression, random forest, XGBoost ensemble, neural network]:
    - Evaluate overall performance of traditional phase estimation
    - For noise level p = [0.00, 0.01, 0.02, . . ., 0.20]
      - Evaluate each algorithm to determine performance variation with noise
    - Training parameters are summarized in Tables 1 and 3
- Analyze and compare model performance
  - Compare metrics = [MSE, prediction speed, overfitting, variation of accuracy with p] for each algorithm.
  - Select the optimal algorithm as the best tradeoff of these metrics.
- Validation
  - Initialize IBMQ Perth quantum computer using qiskit_ibm_runtime.
  - Send circuit output to trained models.
  - Compare predictions from models to the analytic results.

2.1. Quantum Circuit Modeling

In order to construct the datasets, a Python package performing high-level quantum computing tasks was used. The package qiskit developed by IBM was used to generate QPE circuits with either 2, 3, 4 or 5 qubits in the first register and 1 qubit in the second register. Aside from the number of qubits in the first register, the distinguishing features of a standard QPE circuit are the specifications of U and $|u\rangle$. In our case, regardless of the number of qubits in the first register, the unitary operator whose phase was being estimated

was the phase gate $P(\phi)$ from Equation (6). The phase gate denoted in matrix form, and its eigenvectors $|0\rangle$ and $|1\rangle$ are shown in Equations (7)–(9). The corresponding eigenvalues are 1 and $e^{i\phi}$.

$$P(\phi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix} \tag{7}$$

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{8}$$

$$|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{9}$$

The second register is initially in the state $|0\rangle$, but an X gate is applied to the qubit transforming it into the $|1\rangle$ state. Thus, when the unitary operations are applied, $U|u\rangle = e^{2\pi i\theta}|u\rangle$ becomes $P(\phi)|1\rangle = e^{i\phi}|1\rangle$, so we have $\theta = \phi/2\pi$. The values of θ estimated by the QPE circuits were chosen uniformly from the interval $[0, 1)$, and thus the values of ϕ in each Phase gate were chosen uniformly from the interval $[0, 2\pi)$. For each value of θ, 21 circuits were generated with different levels of noise, the specifications of which are detailed in Section 2.2.

2.2. Noise Modeling

Noise was introduced via the qiskit NoiseModel, which contains options for adding many different types of noise to the circuit. In this case, in order to create a tunable parameter of noise, the type of noise added was the depolarizing error of probability p for both 1 and 2 qubit gates since the circuits in this work have only 1 and 2 qubit gates. The circuit having noise parameter p essentially means that there is probability p that, when a 1 or 2 qubit gate is applied, the qubits involved will depolarize, going from their initial state (needed for the algorithm to work) to a mixed state. The further details of quantum circuit noise and depolarizing noise have been explored in the literature; however, they are beyond the scope of this paper [4,12,13]. Figure 5 shows the feature counts when 100,000 shots are simulated within a 4-qubit system with $\theta = 1/3$. The left/center/right panels show the features with noise levels of $p = 0$, 0.05 and 0.2. The result of the absence of noise (left) was a strong, noticeable peak near the best approximation of $\theta = 1/3$, and a less prominent peak when noise was increased (right) when $p = 0.2$.

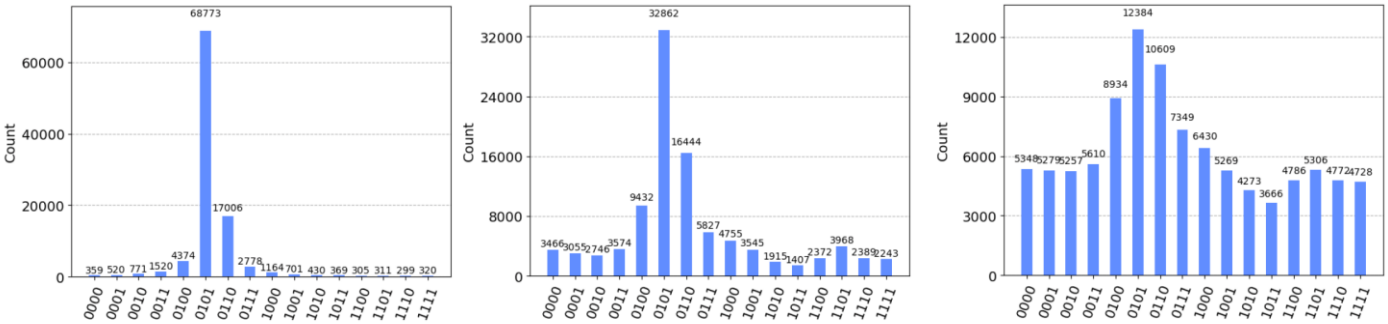

Figure 5. Feature distribution for a 4-qubit system with a quantum phase of $\theta = 1/3$ and varying levels of noise. The noise levels are $p = 0$, 0.05 and 0.2 for the left, center and right panels, respectively [14]. Note that the y-axis is not fixed.

The result of increasing the noise created a decaying signal in the ultimate distribution. Each time a phase was selected for an n-qubit circuit, a circuit estimating that phase was created in qiskit 21 times, each with a noise parameter p ranging from 0 to 0.2 in steps of 0.01. This resulted in 21 data points for each of the phases that was used to train the models. With the exception of $n = 2$, the circuits were simulated with $1000 \cdot 2^n$ shots; essentially, samples from the probability distribution, where n is the number of qubits (ranging from

2 to 5). The 2-qubit circuits sampled 10,000 phases, the 3-qubit circuits sampled 2500 phases, the 4-qubit circuits sampled 5000 phases, and the 5-qubit circuits sampled 10,000 phases.

2.3. Metrics and Algorithms

The modeling effort analyzed the potential of five algorithms to predict the quantum phase of a series of qubits: the traditional method of estimating the phase of the eigenvalue of a unitary operator, linear regression, random forest, the ensemble XGBoost algorithm and neural networks. Additionally, the performance of a trivial model was measured by calculating the performance of a model that predicted the mean of the dataset.

The mean squared error (MSE) was selected as the primary performance metric due to its widespread usage in regression problems. Overfitting is another metric used in this work, which is defined as the MSE measured on the training dataset divided by the MSE measured on a holdout/test dataset. A value of overfitting greater than 1 indicates the model does not generalize well on unseen data. A small level of overfitting is tolerated but a high level is undesirable. The performance for all algorithms is summarized in Section 3.

2.3.1. Traditional Phase Estimation

Recall that the QPE algorithm estimates the phase of the eigenvalue of a unitary operator (i.e., the $\theta \in [0, 1)$ in the characteristic equation $\hat{U}|u\rangle = e^{2\pi i\theta}\left|u\right\rangle$). The traditional method of estimation only approximates the phase to a precision limited by the number of qubits in the quantum computer. This is performed by taking the most occurring output (in the numerical basis ranging from 0 to 2^{n-1}, where n is the number of qubits in the first register) and dividing it by 2^n. Thus, the traditional estimator for the phase based on a given dataset is the mode of the dataset divided by 2^n. This is the traditional method since it can be shown that the mode of the dataset occurs with probability at least $4/\pi^2 \approx 0.405$, and that this is the best approximation of θ. As an example, if the most common sequence of bits in a 4-qubit first register was 0110, the estimation for θ would be $10/16 = 0.625$. This traditional method leaves information about the phase behind in the resulting distribution since values of θ in the range $(9.5/16, 10.5/16)$ will all produce distinct distributions, all with mode 0110. That is, given n bits used in the first register, the traditional, mode estimator can only estimate θ to one part in 2^{-n} bits of accuracy. Note that, due to the periodic nature of the complex exponential, θ near 1 can cause the mode to occur at 0 instead of $2^n - 1$. For large n, this method can produce a very accurate result, but large numbers of highly accurate qubits are not available on today's computers. We are seeking to compare how well the traditional method compares to machine learning methods to predict the phase of a unitary operator. Efforts to use machine learning to estimate the phase given the output of the algorithm would be considered successful if they achieved a level of precision greater than that afforded by the traditional and trivial method for the given number of qubits.

The model performance in traditional estimation was calculated on the entire dataset, as opposed to using the test/holdout datasets that the other algorithms used. This is due to the traditional method relying on a calculation based on the features for each row, instead of an algorithm that learns and generalizes on the training data.

2.3.2. Linear Regression

Within Python, the *sklearn* ordinary least squares (OLS) linear regression algorithm was applied as it is a common algorithm for regression problems. OLS linear regression is a statistical method used to model the relationship between the predictors (channels) and the dependent variable (quantum phase) by fitting a linear equation to the data. The algorithm minimizes the sum of squared differences between the observed and predicted values of the dependent variable. Other forms of linear regression, such as ridge regression or lasso regression, were not used as the dataset did not suffer from multi-collinearity or have need for feature selection [15]. Within the *sklearn* OLS algorithm, the default Moore–Penrose pseudo-inverse solver was used, and all features were used without interaction terms.

2.3.3. Tree-Based Methods

Two tree-based methods were evaluated in this work: the *sklearn* RandomForestRegressor algorithm was evaluated as it performed well in a quantum phase transition problem [16], and the extreme gradient boosting algorithm XGBoost was selected to investigate the performance of an ensemble algorithm, which can often rival the performance of neural networks. Tree-based algorithms are known to overfit if the maximum depth hyperparameter is too high [17,18], and the example in Figure 6 highlights this issue. The right side of the figure shows the training and test dataset performance for max_depth = 30, which overfits the training dataset by 85%. The inferior performance of the green test dataset curve shows that the model has memorized the training data and does not perform well on the test data. The left side of the figure shows the same chart for max_depth = 10, where overfitting is limited to 15%. Even though the "entire dataset" metric makes it seem the max_depth = 30 model is better, it is noteworthy that the mean "test dataset" performance is very similar to the overfit max_depth = 30 model. The max_depth = 10 model will generalize better on unseen data, and its performance is equivalent on unseen data.

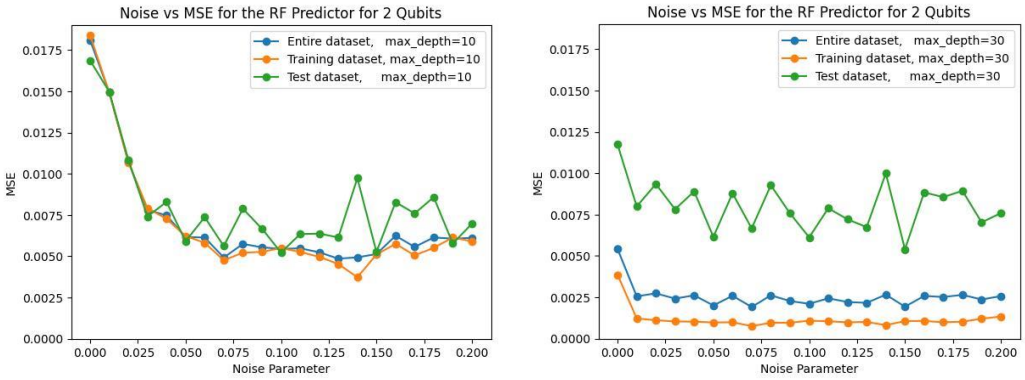

Figure 6. Overfitting example for the random forest algorithm and the 2-qubit datasets for a maximum depth parameter of 10 (**left**) and 30 (**right**). For these examples, the MSE of the training dataset (orange), test dataset (green) and entire dataset (blue) are presented for varying levels of noise.

The random forest algorithm used 200 trees, a squared error criterion, and the maximum depth parameter was swept to determine the best performance that can be obtained with <15% overfitting. In this analysis, the maximum depth was varied from 1 to 11 for each of the 2/3/4/5-qubit models to determine this parameter.

Figure 7 shows the evaluation process for the 2-qubit model, where a model was created for each value of maximum depth ranging from 1–11. In Figure 7, the *x*-axis is the maximum depth parameter, and the blue line shows that model MSE improves with a higher depth parameter. The green line plots overfitting (defined in Section 2.3), showing that the algorithm quickly overfits as the maximum depth parameter is increased. The dashed green line shows a 15% overfitting threshold, which, in this case, is exceeded for a value of maximum depth > 10. The maximum depth parameter where overfitting is limited to <15% is provided in Table 1 for the 2/3/4/5-qubit models, for both the random forest and XGBoost algorithms.

Similar to the random forest algorithm, the XGBoost maximum depth parameter was swept to determine the highest depth parameter that can be used with <15% overfitting. The results of the sweep are presented in Table 1 for the 2/3/4/5-qubit models, and it can be seen that the XGBoost algorithm overfit the data at lower values of maximum depth.

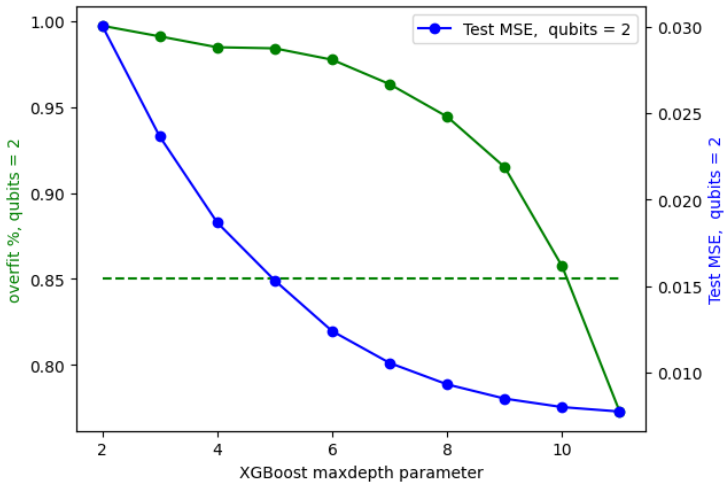

Figure 7. For the 2-qubit dataset, model MSE as measured on the test dataset (blue line, right axis), overfitting ratio (green line, left axis) and 15% overfitting threshold (dashed green line), for varying levels of the maximum depth parameter.

Table 1. Threshold for maximum depth to achieve <15% overfitting on the test/holdout dataset.

Algorithm	2 Qubit	3 Qubit	4 Qubit	5 Qubit
Random Forest	10	7	6	7
XGBoost	4	2	2	3

2.3.4. Neural Network

The Python *keras* framework was used to create neural network models using a sequential architecture, with a baseline model and hyperparameter-tuned model created for each dataset. The baseline model consisted of four layers: a TensorFlow *preprocessing* normalization layer to standardize the inputs, two hidden dense layers of ReLU neurons with L2 = 0.001 regularization, and an output dense layer consisting of a single linear neuron signifying a regression problem. An adaptive moment estimation (Adam) optimizer was selected as the optimizer, and the model was trained for 240 epochs at an initial learning rate of 0.2. To aid in training, a callback halved the learning rate every 30 epochs. A 5–100 neurons sweep was conducted to establish a starting point for the multi-dimensional hyperparameter optimization.

The baseline neural network size was informed by Widrow's rule of thumb, which relates the number of neurons to the number of data points P, the number of weights (neurons $\times$ (inputs + 1)), and the desired error level according to Equation (10) [19].

$$P = \frac{neurons \times (inputs + 1)}{error} \tag{10}$$

The maximum recommended neuron count for the 2/3/4/5-qubit models was then calculated as 1250/175/185/190 neurons (respectively) by solving Equation (10) for *neurons*. The inputs to the equation were the average 3% error level in this work, 4/8/16/32 inputs, and the 210,000/52,500/105,000/210,000 row dataset size. The 2-qubit recommendation was much larger than the others due to possessing only four inputs and having 210,000 rows of data. Maximum neuron counts in the hyperparameter sweeps were limited to within 3x of the Widrow recommendations.

Within Python, the neural network model's hyperparameters were tuned using the Adaptive Experimentation Platform (Ax) library. Ax uses Bayesian optimization for numeric hyperparameters, which include the initial learning rate, number of hidden layers, neurons per layer and batch size. Bandit optimization was used to tune the categorical optimizer hyperparameter. The training dataset was split into training and validation, and

the hyperparameters were tuned using the validation dataset. After tuning to find the optimal set of hyperparameters, the final metrics were calculated using the test/holdout dataset. The optimal hyperparameters are presented in Section 3.5 for each model.

2.4. Speed Analysis and Validation on an IBM Device

If a research application were to require a fast prediction time, the time to execute *model.predict()* on each dataset was recorded within Python for each algorithm in order to provide this for a researcher's consideration. A 2-qubit datapoint was obtained from an IBMQ quantum computer by running *qiskit_ibm_runtime* [20]. The quantum computer was *IBMQ_perth*, whose qubit layout is shown in Figure 8. This quantum computer is no longer available for use but the following specifications have been reported for this device in the literature [21,22] and information about the job is retrievable [23]. It is a 7 superconducting transmon qubit quantum computer with a quantum volume of 32. The basis gates are the CNOT, I, RX, SX, and X gates. The T1 time = 168.85 msec and the T2 time = 132.51 msec. The median CNOT error rate = 8.690×10^{-3} and the median SX error rate = 2.8060×10^{-4}. The median readout error = 2.930×10^{-2}.

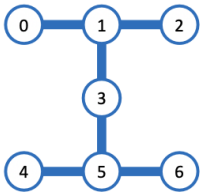

Figure 8. Device topology for the IBM Perth quantum computer [21]. The numbers 0–6 refer to individual transmon qubits.

There was no use of any error mitigation from *qiskit_ibm_runtime* for these data. The phase and channels were 1/3 and [0.092, 0.58175, 0.181, 0.14525], respectively. This data point used for validation was directly extracted from the IBM Quantum jobs page on the date of the experiment. The best 2-qubit model was then used to predict this datapoint, and the prediction was compared to the actual phase and the traditional method of phase estimation.

3. Results

In this section, the model accuracy, prediction speed, overfitting level and variation in accuracy with noise level for five quantum phase estimation algorithms are presented in Table 2. Additionally, the performance of the trivial mean-predicting model is shown in the top row. Notable aspects of modeling using each of the algorithms are discussed below the table.

Table 2. Test/holdout MSE for each algorithm and dataset. The speed column contains the prediction speed as measured on a large batch of the dataset.

Algorithm	2 Qubit MSE	3 Qubit MSE	4 Qubit MSE	5 Qubit MSE	Average Execution Speed (Records/s)	Overfit	Notes
Trivial	0.0839	0.0816	0.0843	0.0828	N/A	--	Mean-predicting model
Traditional	0.1200	0.0160	0.0190	0.0180	10,000	--	Mostly level noise profile
Linear Regression	0.0500	0.0240	0.0330	0.0330	51,800,000	<1%	No interaction terms used. Highly varying noise profile
Random Forest	0.0080	0.0070	0.0060	0.0035	228,000	<15%	Moderately varying noise profile
XGBoost	0.0090	0.0065	0.0035	0.0015	798,000	<15%	Low varying noise profile
Ax-tuned NN	0.0069	0.0027	0.0011	0.0005	13,100	<10%	Low varying noise profile

3.1. Traditional Phase Estimation

The performance of the traditional, highest-peak estimator is shown in Figure 9 for all four datasets and the 21 levels of noise present in each dataset. The 2-qubit dataset is shown in blue and has notably worse performance than the other datasets. This is due to the fact that the error of this method goes as 2^{-n}, where n is the number of qubits. Thus, the jump in error from 2 to 3 qubits is larger than the jump from 3 to 4 and from 4 to 5. The similarity in error for the 3/4/5 qubit datasets is due to the dominant contribution of phases near 0 and 1, where phases in the interval $\left(2^{-n-1}, 1\right)$ will generate a prediction of 0 using the traditional method. This is because, as was covered in Section 2.3.1, the highest peak of the distribution occurs at the integer closest to $2^n\theta$, though this is periodic. For $\theta \in \left(2^{-n-1}, 1\right)$, this integer is 0 and not $2^n - 1$, causing a peak at 0 and a large contribution to MSE. In fact, the 3/4/5-qubit datasets all have similar performance near MSE = 0.02. The traditional method exhibits mostly level performance across various noise levels, which is desirable. However, this method had the poorest performance of all algorithms tested, and on the 2-qubit dataset it performed worse than the trivial mean-predicting model.

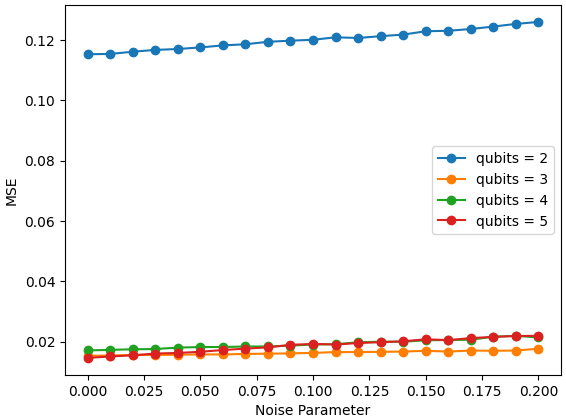

Figure 9. MSE from the traditional phase estimation algorithm for the 2/3/4/5-qubit datasets vs. noise.

3.2. Linear Regression

The performance of the 3-qubit and 5-qubit linear regression models is shown in Figure 10, and the 2-qubit and 4-qubit datasets possessed similar attributes. In each figure panel, the MSE is calculated for the training dataset (orange) test dataset (green) and the entire dataset (blue) for the 21 values of noise. As is visible in the figure, the linear regression algorithm had a very low level of overfitting, calculated as <1% by comparing the performance of the training and test datasets. An undesirable characteristic of the linear regression algorithm is that the model performance on all datasets had a high degree of MSE variation with noise level.

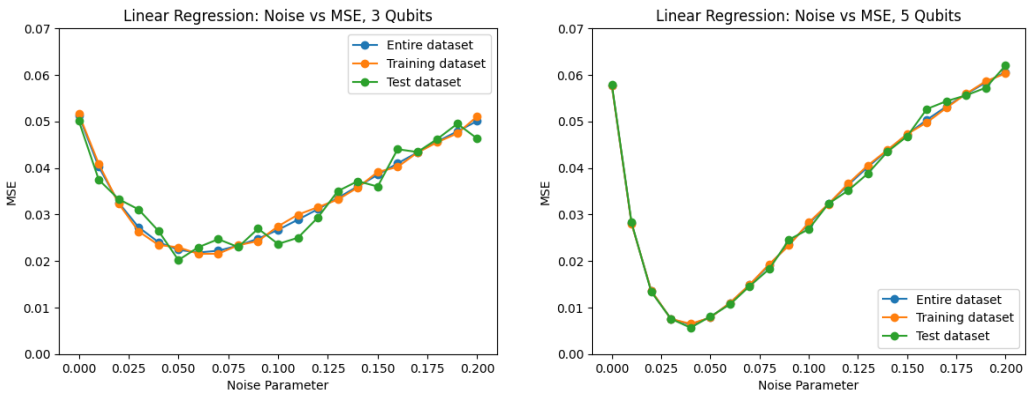

Figure 10. For the 3-qubit dataset (**left**) and 5-qubit dataset (**right**), the MSE of the training dataset (orange), test dataset (green), and entire dataset (blue) are presented for varying levels of noise.

3.3. Tree-Based Algorithms

Figure 11 presents the model performance on the test/holdout split for the random forest (left) and XGBoost (right) tree-based algorithms for all datasets. The unique value of max_depth from Table 1 was used to limit overfitting to <15% for each dataset and algorithm, and that value of max_depth is visible in the legend of each panel. The two algorithms possessed substantially improved performance when compared to linear regression, and similar performance when compared to each other. The figure shows that the variation in MSE with noise was less for the XGBoost algorithm.

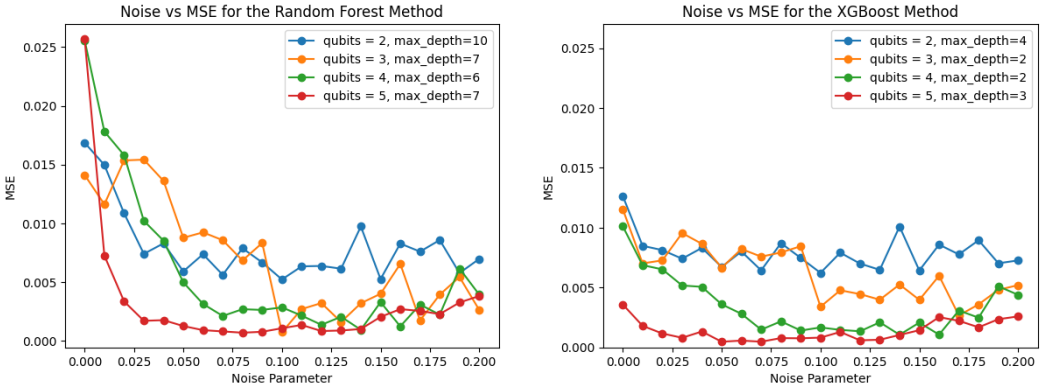

Figure 11. MSE vs. noise for all datasets when using the random forest (**left**) and XGBoost (**right**) tree-based algorithms. The legend of each panel identifies the dataset and value of max_depth from Table 1 used to generate each curve.

3.4. Neural Networks

Two approaches to NN were studied: a baseline 1D hyperparameter sweep using neuron count, and a multi-dimension hyperparameter sweep using Ax. For the 1D sweep, the optimal neuron count for the 2/3/4/5-qubit models was found to be two hidden layers of 200/50/35/50 neurons, respectively.

The Adaptive Experimentation Platform (Ax) library was used to tune the neural network model's hyperparameters, and Ax converged on a solution within 30 optimization loops. Table 3 presents a summary of the multi-dimensional hyperparameter tuning using Ax, including the hyperparameters tuned, their search range, the best set of hyperparameters for each dataset, and the performance achieved on the test/holdout potion of each dataset.

Table 3. Ax-optimized neural network hyperparameter sweep range, and the optimal set of hyperparameter selected for each model. The performance is calculated on the test/holdout dataset. SGD: stochastic gradient descent; RMSprop: root mean square propagation.

Hyperparameter	Range	2 Qubit	3 Qubit	4 Qubit	5 Qubit
Initial learning rate	0.1–0.3	0.3	0.25	0.28	0.19
Number of hidden layers	1–4	3	3	4	3
Neurons/layer	5–100	88	74	94	47
Batch size	128–4096	128	522	128	307
Optimizer	Adam, SGD, RMSprop	SGD	SGD	SGD	SGD
Test MSE	--	0.0069	0.0027	0.0011	0.0005
Overfit %	--	−2.1%	10%	1.6%	5.6%

For the 2-qubit model the NN training curves and model performance on varying levels of noise are presented in Figure 12. The left side of the figure shows the impact of the decreasing learning rate, which enables small improvements in performance every

30 epochs of training. The right side of the figure shows relatively level performance across a range of noise values, which is desirable. For the 3/4/5-qubit datasets, the training curves and noise-varying model performance of the Ax-optimized NNs showed similar attributes to those in Figure 12.

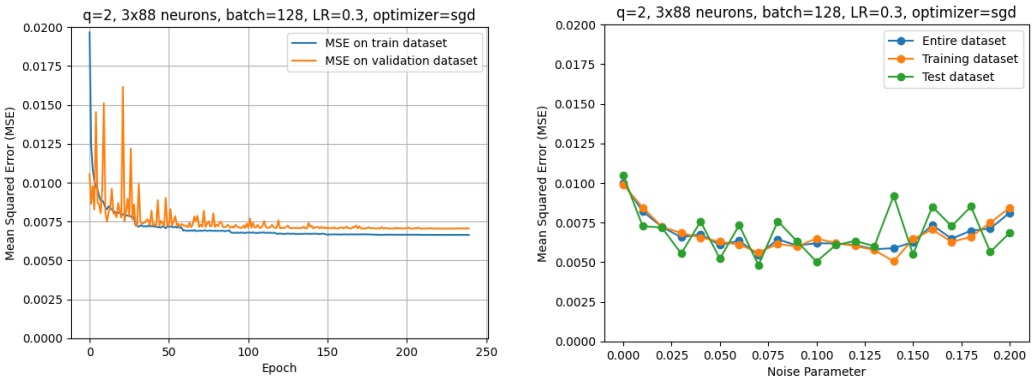

Figure 12. For the 2-qubit dataset, performance of the final NN model using the Ax-optimized set of hyperparameters as shown in Table 3. The training curves on the train dataset are shown in blue, and the validation split is shown in orange (**left**), and the model performance on the training/test/entire datasets are presented for varying levels of noise (**right**).

3.5. Quantum Experiment

The [0.092, 0.58175, 0.181, 0.14525] 2-qubit datapoint from the IBMQ quantum computer Perth possessed a phase of 1/3 [20]. The traditional highest-peak method output a phase of 0.25, which had a 25% error when compared to the actual phase. The best 2-qubit model predicted a phase of 0.3394, yielding an error of 0.006.

4. Discussion

An ideal QPE post-processing algorithm would predict the phase accurately in the presence of noise, have non-varying performance across a variety of noise levels, possess minimal overfitting, and have a rapid prediction time so that lag is not induced into any quantum computing applications where QPE is an intermediate step. No algorithm was a clear winner when considering these four criteria, as the lowest-error model (Ax-tuned NN) was also the slowest predictor; the algorithm with the lowest overfitting (linear regression) and fastest prediction time had the highest error level and a high degree of variation of error with noise. The XGBoost ensemble method was judged to be the best tradeoff between these criteria, as it had the second-best error level, second-best prediction time and low variation of error with noise. It is worth noting that this is not necessarily a negative result, as not all applications require speed or maximum accuracy.

A residual analysis for the 5-qubit XGBoost algorithm is shown in Figure 13, and it can be seen that the model struggles to predict the boundary phases near 0 and near 1 since the phase is an angular measurement that passes continuously between 0 and 1.

The second-best algorithm was the Ax-tuned NN models as they had the lowest error level and variation of error with noise. Of the three optimizers compared, the stochastic gradient descent (SGD) optimizer was best for all datasets. Other generalizations noted across all datasets were that the best performance was achieved with high initial learning rates, three or four hidden layers, and batch sizes (128–522) that were low in comparison to the 52,500–210,000 row dataset size.

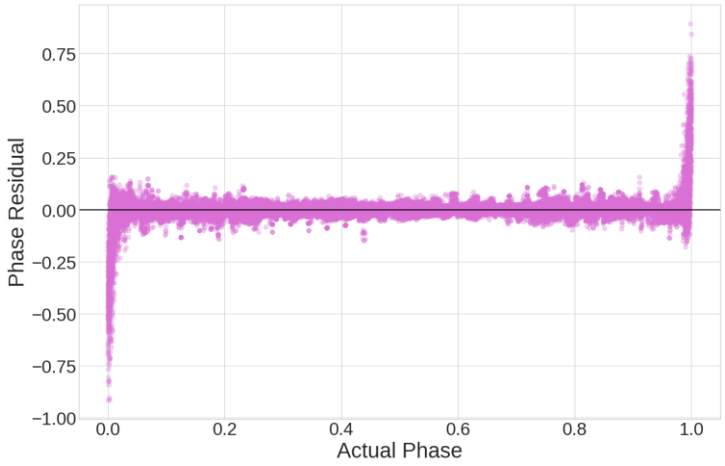

Figure 13. Residual analysis for the XGBoost algorithm predicting the 5-qubit dataset. All levels of noise are present in this sample.

Limitations and Drawbacks

There are a few limitations of this method of enhancing QPE that are important to consider. First, this method requires the simulation of large datasets, which is computationally expensive for small qubit systems (like those considered here) and nearly impossible for many qubit systems since the computational difficulty is exponential. Since the accuracy of the traditional method increases exponentially with the number of qubits as well, a QPE system of several qubits free of noise would be better than the best performance of our method. Therefore, the useful application for this type of improvement to the standard QPE approach lies in circumstances where only a few qubits are available in the first register. This is, however, the circumstance in which the field of quantum computing finds itself now.

More specific limitations in this experiment include the fact that only one type of error was examined in the simulated circuits. For a more complete picture, Pauli measurement, depolarizing, thermal relaxation, and reset errors (among many others) would need to be inserted into the simulations, though this would require more simulation for every type of noise that is modeled. Furthermore, only one phase was present in the distributions. For general applications, the state of the second register may not be an eigenstate of the unitary operator, which leads to multiple phases affecting the probability distribution from which we sample. Our models are not applicable to these scenarios. Simulations of more general situations with multiple types of error and multiple phases are underway in order to train more advanced machine learning models. Finally, this experiment only measured performance with absolute error in phase, not the periodic error for the phase, $\widetilde{\theta}$, given by $\min\left\{\left|\widetilde{\theta}-\theta\right|, 1-\left|\widetilde{\theta}-\theta\right|\right\}$. It is possible to train models and perform analysis with this periodic error, and having both sets of models may be desired by some researchers. This is, again, an opportunity for further work.

5. Conclusions

In this work, QPE using machine learning was shown to offer significant potential to improve phase estimation in the presence of depolarizing noise. In this work, simulated quantum datasets were generated for 2/3/4/5-qubit systems, and each row of each dataset contained the quantum register, the phase to be predicted, and 21 different levels of depolarizing noise that were added to the phase. The phase was transformed to range from 0–1, and the noise level ranged from 0–0.2. The mean squared error was used as the primary model performance metric, and a 6x–36x improvement in model performance was noted, depending on the dataset, when comparing the Ax-tuned neural network to the highest-peak estimator.

The model prediction speed, overfitting level and variation in accuracy with noise level was determined for five machine learning algorithms. The prediction speed ranged from 10,000–51,800,000 records per second, which could constrain the applications of the slowest algorithms. Overfitting ranged from 1–15%, and the MSE for the neural networks ranged from 0.0069–0.0005. While the XGBoost ensemble algorithm did not possess the lowest error level, it was judged to be the best tradeoff between the four criteria due to its error level, prediction time and low variation of error with noise. The lowest-error model (neural network) was also the slowest predictor; the algorithm with the lowest overfitting and fastest prediction time (linear regression) had the highest error level and a high degree of variation of error with noise. A machine learning prediction was made on a 2-qubit datapoint obtained from an IBMQ 2-qubit quantum computer, and it demonstrated a significant improvement over the traditional method. The models and experiment possess the potential to increase the QPE accuracy for emerging quantum computers.

Author Contributions: Conceptualization, C.W., D.W. and T.W.; methodology, T.W. and D.W.; software, C.W. and T.W.; validation, C.W. and T.W.; formal analysis, C.W. and T.W.; investigation, C.W. and T.W.; data curation, C.W.; writing—original draft preparation, C.W. and T.W.; writing—review and editing, C.W., D.W. and T.W.; visualization, T.W. All authors have read and agreed to the published version of the manuscript.

Funding: The authors gratefully acknowledge support for this work from the Air Force Research Laboratory, including access to IBM Quantum resources.

Data Availability Statement: The original contributions presented in the study are included in the article; further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest. The views expressed are those of the authors and do not reflect the official guidance or position of the United States Government, the Department of Defense, the United States Air Force, the United States Space Force or any agency thereof. Reference to specific commercial products does not constitute or imply its endorsement, recommendation, or favoring by the U.S. Government. The authors declare this is a work of the U.S. Government and is not subject to copyright protections in the United States. This article has been cleared with case number 88ABW-2024-0139.

References

1. Shankar, R. *Principles of Quantum Mechanics*, 2nd ed.; Plenum Press: New York, NY, USA, 2011.
2. Nielsen, M.; Chuang, I. *Quantum Computation and Quantum Information*, 10th ed.; Cambridge University Press: Cambridge, UK, 2010.
3. Cleve, R.; Ekert, A.; Macchiavello, C.; Mosca, M. Quantum Algorithms Revisited. *Proc. R. Soc. London. Ser. A Math. Phys. Eng. Sci.* **1998**, *454*, 339–354. [CrossRef]
4. Kitaev, A.Y. Quantum measurements and the Abelian Stabilizer Problem. *arXiv* **1995**, arXiv:quant-ph/9511026.
5. Chapeau-Blondeau, F. Modeling and Simulation of a Quantum Thermal Noise on the Qubit. *Fluct. Noise Lett.* **2022**, *21*, 2250060. [CrossRef]
6. Harper, R.; Flammia, S.; Wallman, J. Efficient Learning of Quantum Noise. *Nat. Phys.* **2020**, *16*, 1184–1188. [CrossRef]
7. Shaib, A.; Naim, M.H.; Fouda, M.E.; Kanj, R.; Kurdahi, F. Efficient noise mitigation technique for quantum computing. *Sci. Rep.* **2023**, *13*, 3912. [CrossRef] [PubMed]
8. IBM Corporation. *IBM SPSS Modeler CRISP-DM Guide*; IBM Corporation: Armonk, NY, USA, 2011.
9. Cruz, P.; Catarina, G.; Gautier, R.; Fernández-Rossier, J. Optimizing quantum phase estimation for the simulation of Hamiltonian eigenstates. *Quantum Sci. Technol.* **2020**, *5*, 044005. [CrossRef]
10. Zlokapa, A.; Gheorghiu, A. A deep learning model for noise prediction on near-term quantum devices. *arXiv* **2020**, arXiv:2005.10811.
11. Woodrum, C.; Weeks, D. Machine Learning Approaches to Evaluating Quantum Phase Estimation Algorithm Output. In Proceedings of the NAECON 2023-IEEE National Aerospace and Electronics Conference, Dayton, OH, USA, 28–31 August 2023.
12. Qiskit. Building Noise Models (Qiskit Aer 0.13.1). 2023. Available online: https://qiskit.org/ecosystem/aer/tutorials/3_building_noise_models.html (accessed on 1 September 2023).
13. Qiskit. Applying Noise to Custom Unitary Gates (Qiskit Aer 0.13.1). 2023. Available online: https://qiskit.org/ecosystem/aer/tutorials/4_custom_gate_noise.html (accessed on 3 February 2024).
14. Woodrum, C. Methods of Evaluating Quantum Phase Estimation Circuit Output. Master's Thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH, USA, 2023.
15. Géron, A. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*; O'Riley: Newton, MA, USA, 2019.

16. Canabarro, A.; Fanchini, F.F.; Malvezzi, A.L.; Pereira, R.; Chaves, R. Unveiling phase transitions with machine learning. *Phys. Rev. B* **2019**, *100*, 045129. [CrossRef]
17. Saul, J.; Wagner, T.; Mbonimpa, E.; Langhals, B. Atmospheric Meteorological Effects on Forecasting Daily Lightning Occurrence at Cape Canaveral Space Force Station. In Proceedings of the 2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE), Las Vegas, NV, USA, 24–27 July 2023.
18. Tucker, T.; Wagner, T.; Auclair, P.; Langhals, B. Machine Learning Prediction of DoD Personal Property Shipment Costs. In Proceedings of the 2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE), Las Vegas, NV, USA, 24–27 July 2023.
19. Widrow, B. ADALINE and MADALINE. In Proceedings of the 1st International Conference on Neural Networks, San Diego, CA, USA, 21 June 1987.
20. IBM Quantum Team. IBMQ_Perth Quantum Computer, Job ID: 4vm4mnq2dtrqsq566g. 2024. Available online: https://quantum-computing.ibm.com (accessed on 2 February 2024).
21. AbuGhanem, M. Full Quantum Process Tomography of a Universal Entangling Gate on an IBM's Quantum Computer. *arXiv* **2024**, arXiv:2402.06946.
22. Trochatos, T.; Xu, C.; Deshpande, S.; Lu, Y.; Ding, Y.; Szefer, J. Hardware Architecture for a Quantum Computer Trusted Execution Environment. *arXiv* **2023**, arXiv:2308.03897.
23. IBM Quantum Documentation. Retired Systems. Available online: https://docs.quantum.ibm.com/run/retired-systems#retrieve (accessed on 9 May 2024).