IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 6, 2020, accepted November 16, 2020, date of publication November 26, 2020,
date of current version December 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3040798

TensorFlow Quantum: Impacts of Quantum State Preparation on Quantum Machine Learning Performance

DANIEL SIERRA-SOSA, MICHAEL TELAHUN, (Graduate Student Member, IEEE), AND ADEL ELMAGHRABY, (Life Senior Member, IEEE)
Department of Computer Science and Engineering, University of Louisville, Louisville, KY 40292, USA
Corresponding author: Michael Telahun (michael.telahun@louisville.edu)

ABSTRACT Learning methodologies on quantum devices have shown that there are advantages in utilizing quantum properties. A requirement for using quantum computing in machine learning techniques is the data representation as quantum states. In Quantum Machine Learning, quantum state preparation is paramount to attain a functional pipeline in a model. One state preparation method, amplitude encoding, allows a dataset to be mapped or encoded more robustly and enhances the learning of quantum models. Albeit more densely represented, a dataset which has been prepared by amplitude encoding provides a more learnable input to a model. The two main advantages from using amplitude encoding are an increase in classification accuracy and reduced variability of learning epoch to epoch. In this paper, we compare the basic implementations of TensorFlow Quantum's Quantum Convolutional Neural Network and a hybrid quantum-classical network using angle encoding, with a third network of our design that utilizes amplitude encoding for enriched state preparation. Our results show there is a direct benefit in performing amplitude encoding before training a TensorFlow Quantum hybrid quantum-classical model. In the best case scenario, amplitude encoding made classifying the samples 8.9% more accurate.

INDEX TERMS Amplitude encoding, machine learning, quantum computing, quantum information, and state preparation.

I. INTRODUCTION

Quantum Machine Learning (QML) is an interdisciplinary field where Quantum Computing (QC) and Machine Learning (ML) converge. Interest in QML over the last couple of years has grown largely due to the advances in hardware implementations of quantum devices known as Noisy Intermediate Scale Quantum (NISQ) devices [1], [2]. The goal of this rising field is to describe learning models that apply the benefits of computing on quantum devices so that operations in machine learning can be performed [3] and potentially improved. It should be noted that we are still not able to show QML on NISQ devices can surpass the abilities of classical techniques [4]–[6]. However, the constantly improving capabilities of quantum information processing is promising [7], [8], and NISQ era devices are getting closer to one day outperforming their classical counterparts.

State preparation is a fundamental component of data pre-processing for QML. One of these methods is amplitude encoding [9], [10]. This method maps the classical data into the amplitude of fundamental quantum computing unit, the qubit. The qubit is the quantum dual of the binary bit and is represented as $|\psi\rangle$ (read state psi or ket psi). An example of a qubit is defined in (1), where α_n corresponds to the probability amplitudes constrained by (2), $|0\ldots00\rangle, |0\ldots01\rangle, \ldots, |1\ldots11\rangle$ are the basis states, and n is the number of basis states. Unlike a bit, the qubit can be in a linear combination of several states (superposition), meaning that in such a system two or more basis states can coexist [11].

$$|\psi\rangle = \alpha_0|0\ldots00\rangle + \alpha_1|0\ldots01\rangle + \cdots + \alpha_n|1\ldots1\rangle \tag{1}$$

$$\sum_{i=0}^{n} |\alpha_i|^2 = 1 \tag{2}$$

Amplitude encoding can be likened to one-hot encoding, where information is losslessly encoded and decoded. One-hot encoding takes a sample from a dense representation to one that is sparse. For example, if a dataset is

The associate editor coordinating the review of this manuscript and approving it for publication was Szidónia Lefkovits.

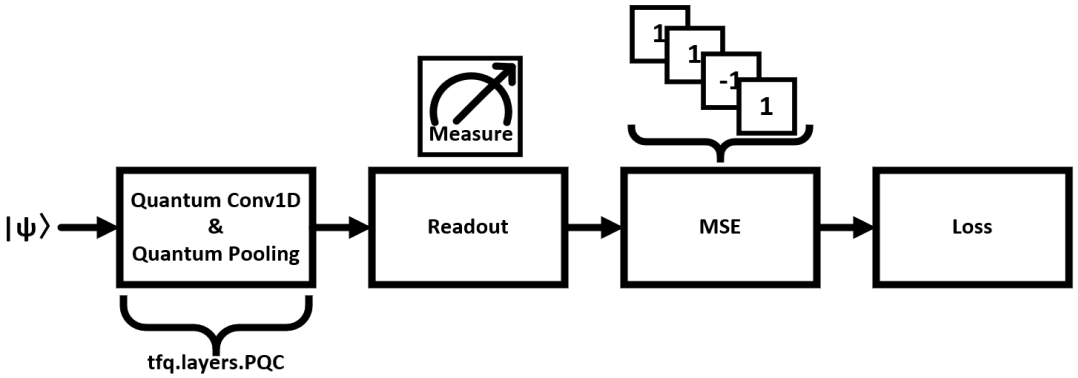

FIGURE 1. Base architecture implemented in each model for quantum convolution and quantum pooling. For the QCNN model this is the whole architecture. MSE and Loss of the Angle-Hybrid and Amplitude-Hybrid follow after a classical MLP is added.

densely represented as $(\langle 1\rangle, \langle 2\rangle, \langle 3\rangle)$ with one-hot it would be encoded into $(\langle 1, 0, 0\rangle, \langle 0, 1, 0\rangle, \langle 0, 0, 1\rangle)$. The application of encoding techniques, in both the classical and quantum sense, may determine whether a dataset can be learned from [12]. Nonethless, in amplitude encoding, a sample is encoded from a classical representation to a quantum representation, changing the computational basis and allowing superposition of states among other properties; whereas in one-hot encoding, a sample only changes its *form* from dense to sparse. In both cases, the encoding method's performance is dependent on the applied architecture.

Google recently announced a new quantum computing library named TensorFlow Quantum (TFQ) for Python [13]. TFQ is a promising library with recently published findings on Google's Sycamore device [14]. TFQ is meant to combine the abilities of the TensorFlow (TF) library with a quantum computing frame of mind. This library is a machine learning library for rapid prototyping of hybrid quantum-classical ML models. TFQ works in conjunction with two other Python libraries, Sympy and Cirq, which leverage symbolic mathematics and quantum logic circuit design respectively [15], [16]. When utilized together, TFQ allows a user to easily develop a QML model, Sympy provides a user the ability to interject trainable quantum parameters, and Cirq allows for the application of logical quantum operators or gates necessary for fundamental quantum computing. These three libraries are the software stack for developing QML techniques in this work. We perform tests and show results of amplitude encoding's current capabilities on a simulated quantum device since TFQ does not mention a publicly available physical quantum device. These results are based on the available models in the TFQ library documentation. The application of amplitude encoding with the TFQ library shows a considerable improvement over a short number of epochs and a measurable decrease in sporadic training behavior. These improvements are in line with what recent work has indicated [17], [18].

The rest of this paper is organized as follows. Architectural design, data sets, and implementation of methods used in this work are given in Section II. In Section III, experimentation results and the analysis of the work done here are given. Finally, the conclusion and discussion of future work is provided in Section IV.

II. METHODS

Two of the three models implemented in this work follow directly from the TFQ documentation, Quantum Convolutional Neural Network (QCNN) and Hybrid Quantum/Classical Neural Network [13]. For the third model we propose a modification of the TFQ Hybrid Quantum/Classical Neural Network which we call Amplitude-Hybrid Quantum/Classical Neural Network as it utilizes amplitude encoding for the preparation method of quantum states. Subsequent descriptions of these three *models* are referred to as ''QCNN'', ''Angle-Hybrid'', and ''Amplitude-Hybrid''. Throughout this work comparisons are focused between the Angle-Hybrid and Amplitude-Hybrid models but QCNN is included for completeness.

A. ARCHITECTURE & DESIGN

Each architecture at its core implements the QCNN architecture shown in Fig. 1. This model consists of two layers of quantum convolutions (QConv) with a quantum pooling layer following each of them. The circuit gate implementations for QConv and quantum pooling are depicted in Fig. 2. The Angle-Hybrid architecture consists of this model in Fig. 2 followed by a small multi-layer perceptron (MLP). This architecture combines a quantum learning technique with a classical learning technique, hence the name Hybrid. Our model, Amplitude-Hybrid, uses a pair of QConvs and quantum pooling layers and the same MLP used in the Angle-Hybrid model but a different implementation for how data is fed to the model is used.

Each model shares the same hyperparameters within the QCNN and MLP components. Architecturally, the only difference between the Angle-Hybrid and Amplitude-Hybrid models is our input layer. The input layer for the Amplitude-Hybrid model was modified to allow both quantum encoded samples and classical samples. This was done by first inputting the classical samples to the model at the

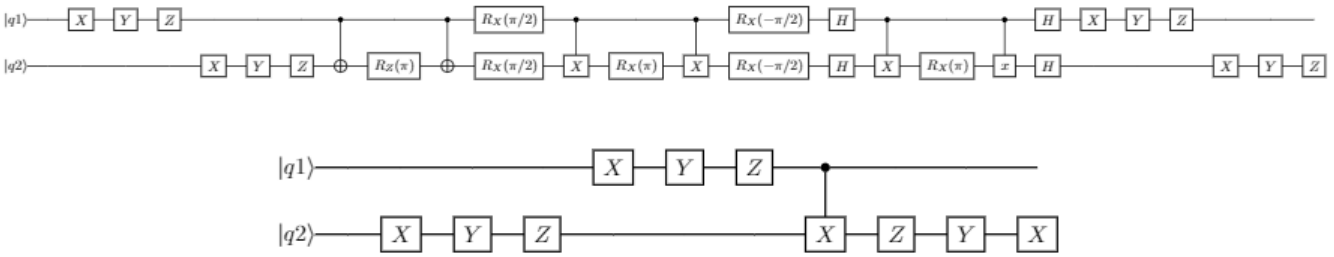

FIGURE 2. Base quantum model circuits implemented in each architecture for quantum convolution (top circuit) and quantum pooling (bottom circuit).

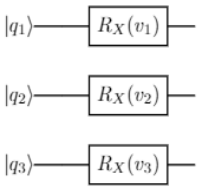

FIGURE 3. Angle preparation of classical data into the state $|\psi\rangle$ for a 3-Dimensional classical sample as used in the TFQ documentation.

beginning, but within the QCNN the quantum encoded samples are used. For the Angle-Hybrid model, angle encoding is performed by using either a rotation about the x-axis or y-axis via an R_X or R_Y gate where the angle is given as the feature for each sample. This method of preparation is shown in Fig. 3.

Where the three dimensional sample's vector $V = \langle v_1, v_2, v_3\rangle$ will be encoded into the states (ψ_1, ψ_2, ψ_3), onto qubits (q_1, q_2, q_3) respectively. As depicted in Fig. 3, this encoding will work for n number of dimensions of a given sample provided there are n number of qubits to generate the encoded set of states. A sample prepared by this method has the same dimensionality as its classical counterpart, is relatively straightforward to prepare, and does not need as much unpacking as an amplitude encoded representation.

In our Amplitude-Hybrid model the difference in model architecture is in the input, our input layer provides some added training influence onto the classical data via the amplitude encoded states. In particular, the Amplitude-Hybrid model additionally includes classical samples as the initial input to the model to influence the encoded states. This change is necessary for this model to encode the features in the amplitude of a given state $|\psi\rangle$. Before the amplitude encoded states are created, the data must be converted to their angle representations with multi-controlled rotations [12] which are performed using (3), where the angle θ is created via a vector, v^i represents the *ith* classical sample, and β is the angle based on the arcsin of the number of dimensions in the sample space. The complexity of a represented state depends on the number of dimensions in a sample, and the angles which map a sample's features is dependent on the contents of a sample [12], [18].

$$|\psi\rangle = R(v^i, \beta)|q_1 \ldots q_{s-1}\rangle|q_s\rangle \tag{3}$$

The state $|\psi\rangle$ is prepared as a circuit of "cascading" R_y rotations such that n R_y applications are performed where n represents the power in binary for encoding a feature vector v^i [12]. Therefore, if a sample has dimensions equal to ten, n would equal four, i.e. since n equal to three at most encodes an eight dimension sample. The circuit in Fig. 4 is used to perform the amplitude encoding for four dimensional samples in the Amplitude-Hybrid model. The complexity of amplitude encoding can be seen visually just by comparing the number of gates in Fig. 4 and Fig. 3. With four dimensions we can also see the limitations of applying amplitude encoding for large datasets in NISQ era devices which are known for the presence of noise and limited quantum volume [7], [19]. In this work, although amplitude encoding only needs three qubits, the number of gates, or circuit depth, applied to create the state $|\psi\rangle$ grew roughly ten fold when compared to the angle encoding method. In a NISQ device this leads to issues such as dechoerence, or a loss of information. This means that on a NISQ device datasets with a large number of features have a hard time utilizing amplitude encoding. Similarly, angle encoding requires a qubit per dimension which again raises the concern of dechoerence but also issues with device topology and total number of qubits.

A sample of data is encoded following a set of just two steps which is expressed via (3) and Fig. 4. First a sample must be converted feature-wise to a set of angles (3). This is so that the information can be used for state preparation [10], [12], [18]. Second the state preparation of the circuit must be created using a combination of *Controlled Y gates* R_y. The "cascade" of *Controlled Y gates* R_y depends on the number of dimensions or features in a sample. Recently, Araujo *et al.* have shown a very through generalization of this method for both conversion of the dataset and state preparation using the classical algorithm for divide and conquer [18].

B. DATASETS

To evaluate performance we trained each model using different synthetic datasets from the Python library Scikit-Learn [20]. The ability to control the classes' distance and feature distribution in the tests was needed in gathering consistent results. We create these datasets by specifying a center box per class *(0, 1)*. The center box or centroid of each class is given as the positive and negative value for each of the classes, i.e. if the center box is 1.6 the centriods for both classes are *(-1.6, 1.6)*. When the centriods are closer to zero the two classes share more overlap and less when the centriods are more spread apart. The two classes were converted from

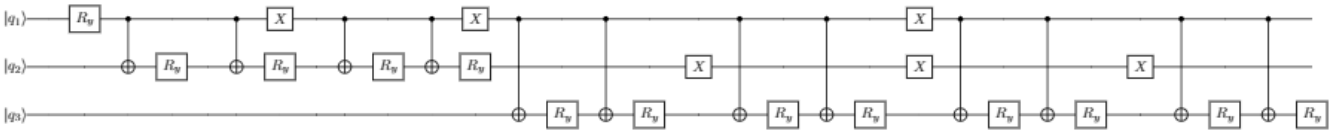

FIGURE 4. Circuit creating the amplitude encoded state $|\psi\rangle$ with three qubits for a classical sample.

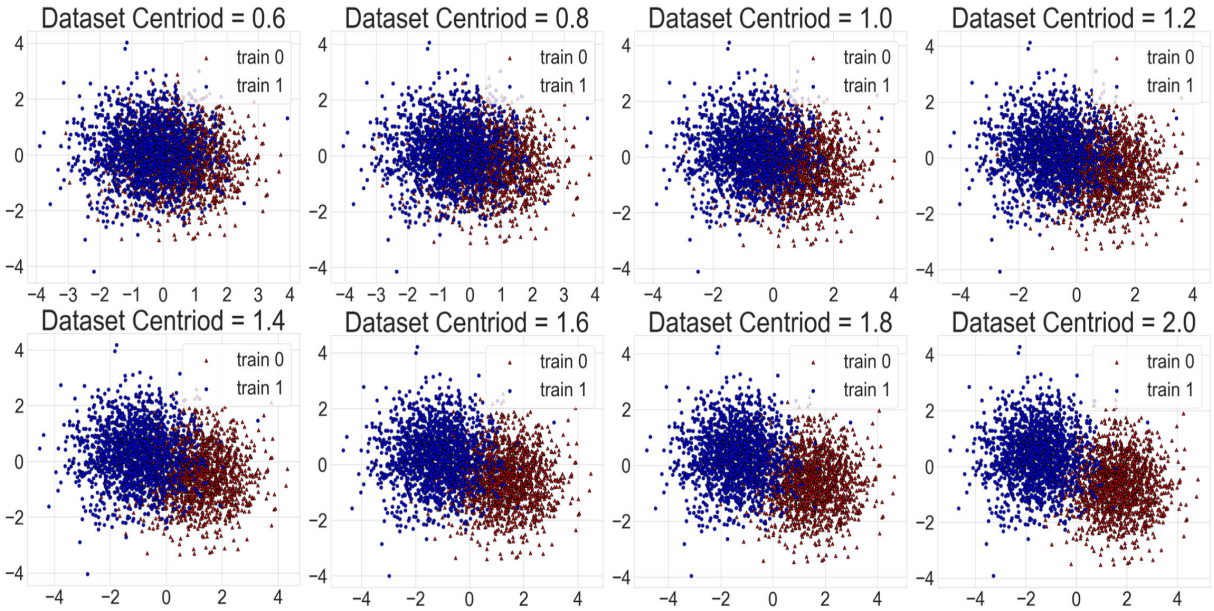

FIGURE 5. Two-dimensional plots of each dataset used to test each model. Noting the separation between classes as the center box values grows from 0.6 to 2.0.

(0, 1) to *(-1, 1)* to keep consistent with TFQ's documentation. Each dataset was created with four features.

With our datasets we conduct two sets of tests to evaluate: (I) the ability of the Amplitude-Hybrid model to converge at a higher accuracy sooner and (II) the ability to learn less erratically and therefore more effectively over time. To test these hypotheses, we train the datasets with eight different centroid distances for both (I) and (II). The datasets are shown in Fig. 5 using a two-dimensional projection plot of the four features. Each training dataset was split into train/validation subsets with 80% and 20% of samples respectively. The train set contained 2,048 training samples, validation contained 512 samples, and test or evaluation was done on an additional 512 samples after training was completed.

III. EXPERIMENTS & RESULTS

The experiments in this work were performed in a classical device that simulates a physical quantum device. This was done primarily because there is currently no quantum computer publicly available that is tied to TFQ. The two tests we will further describe here were based on the number of training epochs, (I) eight-epochs and (II) fifty-epochs. The task for each model was to classify the two classes *(-1, 1)*. The models use the accuracy metric from TFQ documentation, which computes the mean over the set of predicted labels when equal to their true labels. We calculate and show the results for loss, accuracy, precision, recall, and F1-score.

$$accuracy = \frac{1}{n}\sum_{i=1}^{n}(y_i = \tilde{y}_i) \tag{4}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 \tag{5}$$

For both *accuracy* and *MSE* y_i is the observed value and $\tilde{y}_i$ is the predicted value. We use mean square error as our loss function, seen in (5). The optimizer used in the MLP was Adaptive Moment Estimation (Adam) [21] optimizer, described in (6), with the learning rate η set to 0.02, following the TFQ documentation. In (6), θ_{t+1} is the current gradient of the stochastic gradient descent (SGD) based on the previous gradient θ_t,

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t, \tag{6}$$

where the weight $\hat{v}_t$ and momentum $\hat{m}_t$ are defined as:

$$\hat{v}_t = \frac{\beta_2 v_{t-1} + (1-\beta_2)g_t^2}{1-\beta_2^t}$$

$$\hat{m}_t = \frac{\beta_1 v_{t-1} + (1-\beta_1)g_t}{1-\beta_1^t}. \tag{7}$$

with m_t and v_t are estimates of the gradients' mean and variance respectively, and β_1 and β_2 are forgetting factors.

TABLE 1. Testing results on holdout set of 512 additional samples for each model (QCNN, Angle-Hybrid, and Amplitude-Hybrid) after eight epochs of training.

Eight-Epoch Model Metrics									
Model	Centriod	Loss	Accuracy	Precision (-1)	Recall (-1)	F1-Score (-1)	Precision (1)	Recall (1)	F1-Score (1)
QCNN	0.6	0.829	0.683	0.755	0.564	0.646	0.639	0.808	0.713
Angle-Hybrid	0.6	0.965	0.687	0.729	0.618	0.669	0.655	0.760	0.703
Amplitude-Hybrid	0.6	0.889	0.708	0.767	0.618	0.684	0.667	0.804	0.729
QCNN	0.8	0.739	0.726	0.796	0.625	0.700	0.679	0.832	0.748
Angle-Hybrid	0.8	0.808	0.740	0.779	0.687	0.730	0.708	0.796	0.749
Amplitude-Hybrid	0.8	0.702	0.763	0.819	0.690	0.749	0.721	0.840	0.776
QCNN	1.0	0.660	0.767	0.832	0.683	0.750	0.720	0.856	0.782
Angle-Hybrid	1.0	0.672	0.781	0.807	0.751	0.778	0.757	0.812	0.783
Amplitude-Hybrid	1.0	0.533	0.826	0.864	0.782	0.821	0.792	0.872	0.830
QCNN	1.2	0.595	0.792	0.848	0.725	0.781	0.750	0.864	0.802
Angle-Hybrid	1.2	0.563	0.800	0.812	0.793	0.803	0.789	0.808	0.798
Amplitude-Hybrid	1.2	0.396	0.865	0.897	0.832	0.863	0.836	0.900	0.867
QCNN	1.4	0.544	0.826	0.864	0.782	0.821	0.792	0.872	0.830
Angle-Hybrid	1.4	0.475	0.833	0.836	0.839	0.838	0.831	0.828	0.829
Amplitude-Hybrid	1.4	0.281	0.902	0.930	0.874	0.901	0.875	0.932	0.903
QCNN	1.6	0.507	0.859	0.874	0.847	0.860	0.844	0.872	0.858
Angle-Hybrid	1.6	0.406	0.853	0.845	0.874	0.859	0.863	0.832	0.847
Amplitude-Hybrid	1.6	0.183	0.939	0.949	0.931	0.940	0.929	0.948	0.938
QCNN	1.8	0.484	0.871	0.862	0.889	0.875	0.880	0.852	0.865
Angle-Hybrid	1.8	0.357	0.876	0.861	0.904	0.882	0.894	0.848	0.870
Amplitude-Hybrid	1.8	0.113	0.966	0.965	0.969	0.967	0.967	0.964	0.965
QCNN	2.0	0.473	0.892	0.881	0.912	0.896	0.904	0.872	0.887
Angle-Hybrid	2.0	0.286	0.902	0.878	0.938	0.907	0.931	0.864	0.896
Amplitude-Hybrid	2.0	0.101	0.974	0.966	0.984	0.975	0.983	0.964	0.973

The final dense layer of each model (QCNN, Angle-Hybrid, Amplitude-Hybrid) uses tanh as the activation function since the two classes aim to classify *(-1, 1)*. The tanh function or hyperbolic tangent is defined in (8) where x is the current sample weight,

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{8}$$

The *tanh* function is also referred to as a scaled sigmoid function in some literature [22]. Precision is calculated as a ratio of true positive (TP) and false positive (FP)

$$Precision = \frac{TP}{TP + FP}, \tag{9}$$

whereas recall is calculated as a ratio of TP and false negative (FN).

$$Recall = \frac{TP}{TP + FN}, \tag{10}$$

F1-score is calculated as the relationship (harmonic mean) between precision and recall,

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}. \tag{11}$$

Equations (9), (10), and (11) were used to gather additional statistics about each model's behavior on the range of centroid distances after training was completed.

A. EIGHT-EPOCHS

In the first set of experiments, we aimed to show that amplitude encoding can help a model converge faster. Specifically, this would mean fewer training epochs are needed. With that in mind, we train and check the performance of each model for each dataset after just eight epochs and evaluate the results on the testing dataset. Table 1 shows that in every dataset the Amplitude-Hybrid model was the top overall performer for each evaluation metric. Where *(-1)* and *(1)* in the metrics represent the class labels.

We consider each cluster's centroid distance as our metric for classification difficulty. For the first two centroids, *0.6 and 0.8*, the most difficult datasets in this work, the Amplitude-Hybrid model achieved roughly 2% improvement over the Angle-Hybrid model. At *1.4* centroid distance, the Amplitude-Hybrid model in Table 1 is the first occurrence of a model that achieves an accuracy of 90% or higher. Additionally, we see that the Angle-Hybrid model never reaches an accuracy of 90% until the final dataset of 2.0. Each model increases its performance as the cluster overlapping in the datasets decreases, i.e. as we go from centroid *0.6* to centroid *2.0*. By the time the distance is *1.4*, the Amplitude-Hybrid model attains higher than 90% accuracy, overall precision, overall recall, and overall F1-score. The training validation accuracy of each dataset is shown in Fig. 6 along with the validation loss for each model. Several of the Amplitude-Hybrid accuracy plots also show the models began to behave linearly after roughly four epochs, meaning the models overcame the nonlinear approximation of the heuristic. From Table 1 we can see that in almost every difficultly Amplitude-Hybrid outperforms Angle-Hybrid in per class Precision, Recall, and F1-Score. That being said, when we average the two classes together these metrics are always better for Amplitude-Hybrid. When comparing the Fig. 6 we can see that for the first two centroid distances 0.6 and 0.8 in the Amplitude-Hybrid model learn rather well after just training for the eight epochs. In fact overall these two learn better than any other in QCNN or Angle-Hybrid. The thing we must consider here is by also taking into account the results from Table 1 for these three models. In terms of learning better we describe this in terms of increase in accuracy over the eight epochs and decrease in loss. However, the results in every

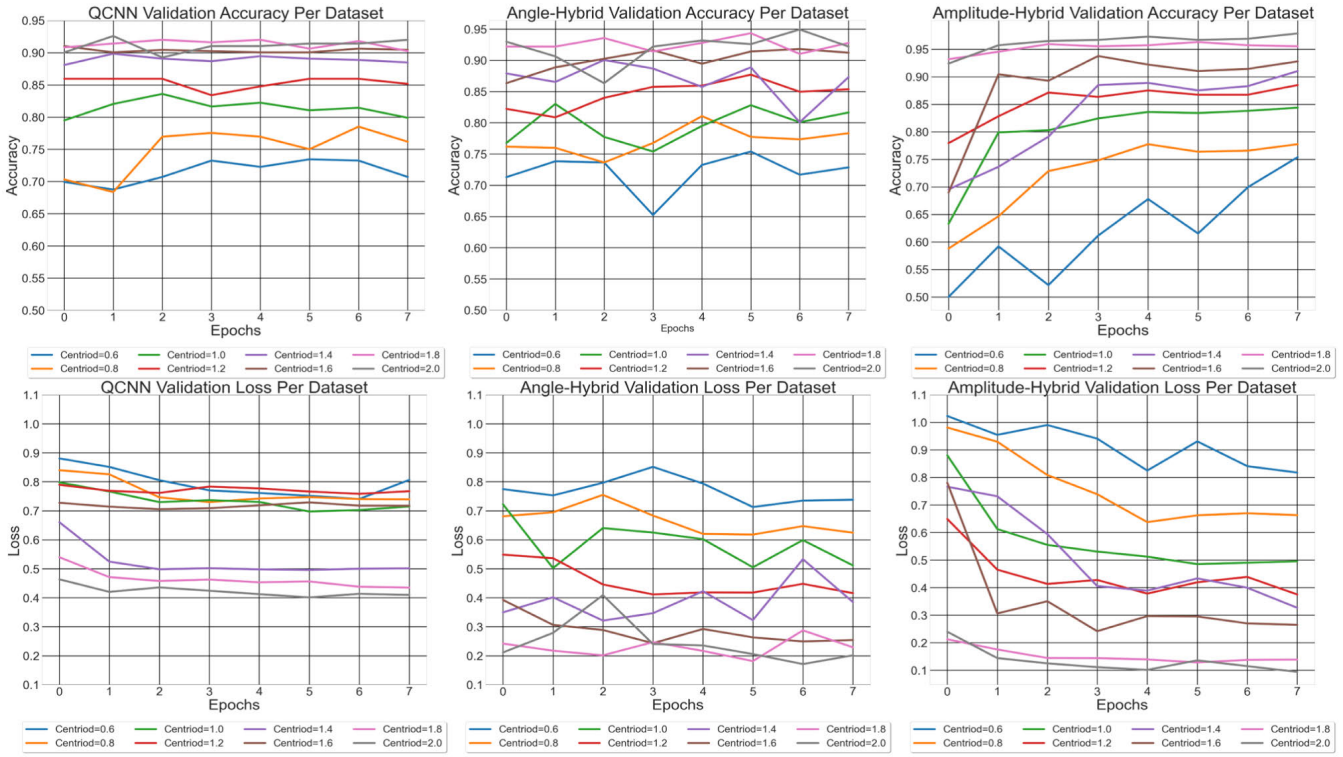

FIGURE 6. **Training histories of each model over eight epochs. Left to right shows the models QCNN, Angle-Hybrid and Amplitude-Hybrid. Top to bottom are the respective training validation accuracy and training validation loss (best seen in color).**

metric (per class and averaged) are very close to the results in both QCNN and Angle-Hybrid, as we have stated above they are still better. But we also know that the first two models utilize a simplified encoding method, as we have described this is angle encoding, and because of this see tightly correlated results in every difficultly. The results in the first two difficulties even the most obvious results. We see that with or without the hybrid quantum/classical addition to the network the QCNN and Angle-Hybrid models perform similarly in that there is not much in terms of accuracy improvement over the eight epochs. In the best case, for QCNN and Angle-Hybrid, Angle-Hybrid was only two percent better than the QCNN model. In the majority of cases it is obvious that from Table 1 that Angle-Hybrid is within 0.5% accuracy of the QCNN model. In contrast the 1.0, 1.2, 1.4, and 1.6 difficulties all improve at least by ten percent from the first epoch and in a few cases even twenty percent for the Amplitude-Hybrid model. When looking at the results in Table 1 these difficulties for QCNN and Angle-Hybrid models are all very similar. It follows that the models utilizing angle encoding limit the ability of overall learning, from Table 1 it is clear that the additional components of the Angle-Hybrid model do not improve the performance. Although the following sections discusses the topic of (II) the same behavior and similarity of results for angle encoding can be gleamed in Table 2.

B. FIFTY-EPOCHS

With the second set of experiments, we sought to show the consistency of learning over time. Specifically, this would mean Amplitude-Hybrid is better at learning the datasets. We consider the eight-epoch scenario presented in Fig. 6 first in our evaluation. Over the eight epochs of training histories, we see that each model is learning with a generally increasing accuracy and decreasing loss, model to model. These trends additionally appear for each centroid distance. With this observation, we consider the results in Table 2 and the training graphs in Fig. 7. It is apparent from the results in Table 2 that the overall results of both the QCNN and Angle-Hybrid model are higher than several of the results presented in Table 1. Additionally in the experiments reported in Table 2, in most cases, the Amplitude-Hybrid model performed a few percent worse than those presented in Table 1. This can be seen in several of the metrics. At first glance, we view these results as counter-intuitive. Arbitrarily training for a larger number of epochs is not always an effective means of achieving increased accuracy or any of the metrics presented in this work. We include this Table 2 for completeness and discuss it further in our conclusions.

To evaluate the variability in training, we consider the training history graphs for fifty-epochs in Fig. 7. It is noted that looking at the Table 2 for variability is not a substantial source as it evaluates the results based on the final epoch on the testing set. The second objective is to consider the variance that arises from training in quantum models [23], [24]. In the case of the Amplitude-Hybrid model, it is apparent that after roughly 10-20 epochs the model has fit to the data as best as it can, considering the peaks in accuracy,

TABLE 2. Testing results on holdout set of 512 additional samples for each model (QCNN, Angle-Hybrid, and Amplitude-Hybrid) after 50 epochs of training.

Fifty-Epoch Model Metrics

Model	Centriod	Loss	Accuracy	Precision (-1)	Recall (-1)	F1-Score (-1)	Precision (1)	Recall (1)	F1-Score (1)
QCNN	0.6	0.798	0.687	0.729	0.618	0.669	0.655	0.760	0.703
Angle-Hybrid	0.6	1.013	0.683	0.771	0.541	0.636	0.634	0.832	0.719
Amplitude-Hybrid	0.6	1.171	0.685	0.798	0.515	0.626	0.629	0.864	0.728
QCNN	0.8	0.708	0.753	0.785	0.713	0.748	0.726	0.796	0.759
Angle-Hybrid	0.8	0.818	0.742	0.821	0.633	0.715	0.690	0.856	0.764
Amplitude-Hybrid	0.8	0.953	0.718	0.835	0.561	0.671	0.657	0.884	0.754
QCNN	1.0	0.631	0.787	0.814	0.755	0.784	0.762	0.820	0.789
Angle-Hybrid	1.0	0.641	0.796	0.859	0.721	0.784	0.750	0.876	0.808
Amplitude-Hybrid	1.0	0.727	0.796	0.898	0.679	0.773	0.732	0.920	0.815
QCNN	1.2	0.568	0.830	0.832	0.835	0.834	0.827	0.824	0.825
Angle-Hybrid	1.2	0.497	0.839	0.891	0.782	0.833	0.797	0.900	0.845
Amplitude-Hybrid	1.2	0.522	0.849	0.930	0.763	0.838	0.791	0.940	0.859
QCNN	1.4	0.520	0.849	0.843	0.866	0.854	0.855	0.832	0.843
Angle-Hybrid	1.4	0.384	0.865	0.903	0.824	0.862	0.831	0.908	0.868
Amplitude-Hybrid	1.4	0.367	0.896	0.948	0.843	0.892	0.853	0.952	0.899
QCNN	1.6	0.486	0.876	0.864	0.900	0.882	0.891	0.852	0.871
Angle-Hybrid	1.6	0.300	0.896	0.916	0.877	0.896	0.877	0.916	0.896
Amplitude-Hybrid	1.6	0.260	0.925	0.970	0.881	0.924	0.886	0.972	0.927
QCNN	1.8	0.467	0.888	0.875	0.912	0.893	0.903	0.864	0.883
Angle-Hybrid	1.8	0.242	0.917	0.926	0.912	0.919	0.909	0.924	0.916
Amplitude-Hybrid	1.8	0.160	0.949	0.975	0.923	0.949	0.924	0.976	0.949
QCNN	2.0	0.347	0.902	0.889	0.923	0.906	0.916	0.880	0.897
Angle-Hybrid	2.0	0.300	0.894	0.906	0.885	0.895	0.882	0.904	0.893
Amplitude-Hybrid	2.0	0.115	0.970	0.976	0.965	0.971	0.964	0.976	0.970

drops in loss, and gradual overfitting in the rest of the history. A gradual or sudden but increasing increase to loss appears when the model starts to overfit, i.e. it retains too much influence of the training data while failing to successfully evaluate the validation data. This consistency and nominal change in variance between training epochs is what we hoped to find in the Amplitude-Hybrid model. The reason we expected this conclusion was due to the amplitude encoding state preparation. The information was encoded from the classical representation with much richer representation than the trivial angle encoding approach described in section II-A Fig. 3. This in turn implies that better learning behavior can be achieved simply by involving a better encoding method.

With respect to the QCNN and Angle-Hybrid models, we can see a difference in the historical outcomes from epoch to epoch in Fig. 7. In both the QCNN and Angle-Hybrid models, we see that almost none of the datasets showed a best fit to the data. In several cases, these models made improvements to accuracy and loss within the first twenty epochs. In a few cases, small improvements can be seen within another twenty epochs. Consider again the 10-20 epoch range mentioned above for the Amplitude-Hybrid model. The Angle-Hybrid's historical behavior is the opposite of the Amplitude-Hybrid model, where the heuristic is consistently improving and then consistently overfitting. In the Angle-Hybrid model, the histories show that the model erratically jumps from a ''high'' accuracy to one that is 10% or even 15% less within just a few epochs. This type of behavior generally implies that learning is failing as the heuristic tries to ''guess'' where the local maximum will be at the end of an iteration (per batch), and shown here at the end of an epoch. Looking at the results in Table 2 we can not see the real behavior of the model, only whether or not the results for any dataset are good. Looking at Table 2 consider the results for Angle-Hybrid and Amplitude Hybrid for 1.0, 1.2, and 1.4 difficulties. The difference in accuracy between these three difficulties is 0.0%, 0.976%, and 3.125% in favor of Amplitude-Hybrid in the latter two. Just looking at the accuracy, even other metrics for that matter such as F1-Score, the former two show there is not much difference between these models. The other metrics such as recall and precision are even higher for the first two with Angle-Hybrid. This information in Table 2 are misleading to imply that the two models perform roughly the same for these difficulties or centriods. They also hardly express what can be seen in the Fig. 7. The training behavior for all three difficulties in Angle-Hybrid appears random when considering the fact that over the fifty epochs each of these models bounced around 5-10% epoch to epoch. In contrast you can see the three models for Amplitude-Hybrid were much tighter over the entire training period while displaying less random behavior epoch to epoch.

The results in both Tables 1 and 2 can be misleading when considered without their training histories in Figs. 6 and 7. We consider the case where the dataset centroid distance is set to *1.2* and look at the results for Angle-Hybrid and Amplitude-Hybrid models. Angle-Hybrid shows that it is on a positive slope that comes back down after a few epochs. Considering the same dataset for the Amplitude-Hybrid model, we see that during this entire training history the model has been roughly 3-5% higher than at the 50^{th} epoch. We can see that at 14 epochs the model performance has peaked and stays there for some time before beginning to overfit. Due to this behavior, we conclude that with amplitude encoding the models that trained for fifty-epochs began to overfit, and therefore the overall drop in metric evaluations is naturally expected.

FIGURE 7. Training histories of each model over fifty epochs. Left to right shows the models QCNN, Angle-Hybrid and Amplitude-Hybrid. Top to bottom are the respective training validation accuracy and training validation loss (best seen in color).

The same cannot be said about the Angle-Hybrid model. In the case of the Angle-Hybrid model, we recognize that training was so erratic that the model performance cannot be ascertained. The results improving after fifty-epochs may not be more than coincidence.

IV. CONCLUSION

From Tables 1 and 2 presented in the results, we consider the most obvious difference, which is that the collected results for eight-epochs surpass those of the fifty-epoch model. We also see that only the Amplitude-Hybrid model consistently over performed the fifty-epoch experiments in the eight-epoch experiments. Although it appears the QCNN and Angle-Hybrid models both show some cases in the fifty-epoch experiments with higher accuracy, among other metrics, than in the eight-epoch experiments, we believe this to be coincidence. In the subsection III-B, we describe that the unpredictability of Amplitude-Hybrid's training over long periods was due to the simplified preparation of states. The variability between epochs in the Angle-Hybrid model make its results inconclusive. This means we cannot find an epoch that would point to the conclusion of learning and the beginning of overfitting. From Fig. 7, it is apparent that after the first dozen or so epochs the Amplitude-Hybrid model does not continue to steadily improve, but it is still better Angle-Hybrid on every dataset. We conclude that roughly 10-20 epochs is an appropriate amount of time to train for most of the datasets.

Amplitude encoding should not be considered the only performance enhancement required to improve a quantum-classical model. Further, state preparation is only one component that machine learning techniques stand to benefit from when applied in quantum or quantum-classical architectures. An additional method of improvement could be how readout is performed when the data passes from the quantum pooling layer to the classical MLP shown in Fig. 2. Ultimately, it is up to the individual to decide what processing, pre-processing, and/or design methods to add to their architecture when trying to enrich quantum-classical models.

Moving forward, we seek to find a representative model and dataset combination that coincides with the results here. With much anticipation, we expect the abilities of amplitude encoding could further benefit specific datasets. Specifically, investigating properties of features found within a dataset, such as distribution, or overall distance between dimensions. We are currently testing our application of amplitude encoding on a physical quantum device with non-synthetic datasets for truly empirical results on a different platform.

REFERENCES

[1] S. Y.-C. Chen, C.-H.-H. Yang, J. Qi, P.-Y. Chen, X. Ma, and H.-S. Goan, "Variational quantum circuits for deep reinforcement learning," *IEEE Access*, vol. 8, pp. 141007–141024, 2020.

[2] W. Liu, P. Gao, Y. Wang, W. Yu, and M. Zhang, "A unitary weights based one-iteration quantum perceptron algorithm for non-ideal training sets," *IEEE Access*, vol. 7, pp. 36854–36865, 2019.

[3] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, pp. 195–202, Sep. 2017.

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

[5] S. S. Tannu and M. K. Qureshi, "Not all qubits are created equal: A case for variability-aware policies for NISQ-era quantum computers," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Apr. 2019, pp. 987–999.

[6] D. Bhattacharjee, A. A. Saki, M. Alam, A. Chattopadhyay, and S. Ghosh, "MUQUT: Multi-constraint quantum circuit mapping on NISQ computers: Invited paper," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2019, Art. no. 8942132.

[7] M. Alam, A. Ash-Saki, and S. Ghosh, "Addressing temporal variations in qubit quality metrics for parameterized quantum circuits," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2019, pp. 1–6.

[8] D. Sierra-Sosa, J. Arcila-Moreno, B. Garcia-Zapirain, C. Castillo-Olea, and A. Elmaghraby, "Dementia prediction applying variational quantum classifier," 2020, *arXiv:2007.08653*. [Online]. Available: http://arxiv.org/abs/2007.08653

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

[10] M. Schuld, I. Sinayskiy, and F. Petruccione, "Prediction by linear regression on a quantum computer," *Phys. Rev. A, Gen. Phys.*, vol. 94, no. 2, Aug. 2016, Art. no. 022342.

[11] M. A. Nielsen and I. Chuang, "Quantum computation and quantum information," Tech. Rep., 2002.

[12] M. Schuld, *Supervised Learning With Quantum Computers*. Springer, 2018.

[13] M. Broughton, G. Verdon, T. McCourt, A. J. Martinez, J. H. Yoo, S. V. Isakov, P. Massey, M. Y. Niu, R. Halavati, E. Peters, M. Leib, A. Skolik, M. Streif, D. Von Dollen, J. R. McClean, S. Boixo, D. Bacon, A. K. Ho, H. Neven, and M. Mohseni, "TensorFlow quantum: A software framework for quantum machine learning," 2020, *arXiv:2003.02989*. [Online]. Available: http://arxiv.org/abs/2003.02989

[14] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, and D. A. Buell, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[15] A. Meurer *et al.*, "SymPy: Symbolic computing in Python," *PeerJ Comput. Sci.*, vol. 3, p. e103, Jan. 2017.

[16] A. Ho and D. Bacon, *Announcing Cirq: An Open Source Framework for NISQ Algorithms*. Google AI Blog, 2018

[17] R. LaRose, A. Tikku, É. O'Neel-Judy, L. Cincio, and P. J. Coles, "Variational quantum state diagonalization," *NPJ Quantum Inf.*, vol. 5, no. 1, pp. 1–10, Dec. 2019.

[18] I. F. Araujo, D. K. Park, F. Petruccione, and A. J. da Silva, "A divide-and-conquer algorithm for quantum state preparation," 2020, *arXiv:2008.01511*. [Online]. Available: http://arxiv.org/abs/2008.01511

[19] P. Das, S. S. Tannu, P. J. Nair, and M. Qureshi, "A case for multi-programming quantum computers," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2019, pp. 291–303.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[22] Y. Ito, "Representation of functions by superpositions of a step or sigmoid function and their applications to neural network theory," *Neural Netw.*, vol. 4, no. 3, pp. 385–394, Jan. 1991.

[23] S. J. Nawaz, S. K. Sharma, S. Wyne, M. N. Patwary, and M. Asaduzzaman, "Quantum machine learning for 6G communication networks: State-of-the-Art and vision for the future," *IEEE Access*, vol. 7, pp. 46317–46350, 2019.

[24] Y. Li, M. Tian, G. Liu, C. Peng, and L. Jiao, "Quantum optimization and quantum learning: A survey," *IEEE Access*, vol. 8, pp. 23568–23593, 2020.

DANIEL SIERRA-SOSA is currently an Assistant Professor with the Computer Science and Engineering Department, University of Louisville, with expertise in mathematical modeling, data analytics, artificial intelligence, machine learning, and quantum computing, his research results have been published in well-known journals. He worked the development of an application for the assessment of patients in health care facilities, a predictive model for patient's outcome, in addition to participating in the development of mobile applications. He is an IBM's Qiskit Advocate and IBM's skills academy instructor, he teaches courses in quantum computing deep learning techniques, programming, signal, and image processing at the undergraduate and graduate level and been an advisor for advanced project courses.

MICHAEL TELAHUN (Graduate Student Member, IEEE) received the B.S. degree in computer science and computer engineering from the University of Louisville, Louisville, KY, USA, in 2019, where he is currently pursuing the M.Eng. degree in computer science and engineering. Since 2018, he has been a Data Scientist with Kindred Healthcare, USA, where he has formerly worked as a Software Developer. Since 2019, he has been a Research Assistant with the Innovative and Emerging Technologies Laboratory, University of Louisville. His research interests include hybrid quantum-classical machine learning, information encoding for quantum devices, image processing, detection and assessment of patient wounds, and applications of deep learning for smart solutions.

ADEL ELMAGHRABY (Life Senior Member, IEEE) has held appointments with the Software Engineering Institute, Carnegie-Mellon University, and the University of Wisconsin–Madison. He is currently the Director of Industrial Research and Innovation and a Winnia Professor of computer science and engineering with the Speed School of Engineering, University of Louisville. He continued collaborations, mentoring, and scientific contributions have resulted in research funding, international collaboration. He has published articles in many prestigious journals, such as the IEEE TRANSACTIONS ON MEDICAL IMAGING, *Medical Physics*, the *Journal of Neuroscience Methods*, and *Protein Engineering*. His research interests include smart cities, data analytics, medical imaging, bioinformatics, and computer-aided diagnostics. He was recognized for his achievements by several professional organizations, including a Golden Core Membership Award by the IEEE Computer Society at the 50th anniversary celebration.