Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

On Depth, Robustness and Performance Using the Data Re-Uploading Single-Qubit Classifier

P. Easom-McCaldin[1], A. Bouridane[1], A. Belatreche[1] and R. Jiang[2]
[1]Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, UK
[2]Department of Computing and Communications, Lancaster University, Lancaster, UK

Corresponding author: A. Bouridane (ahmed.bouridane@northumbria.ac.uk)
This work was partially supported by the QNRF under Grant NPRP No.8–140-2–065 and the EPSRC under Grant EP/P009727/1.

ABSTRACT Quantum machine learning (QML) is a new field in its' infancy, promising performance enhancements over many classical machine learning (ML) algorithms. Data reuploading is a QML algorithm with a focus on utilizing the power of a singular qubit as an individually capable classifier. Recently, there have been studies set out to explore the concept of data re-uploading in a classification setting, however, important aspects are often not considered in experiments, which may hinder our understanding of the methodology's performance. In this work, we conduct an analysis of the single-qubit data re-uploading methodology, in relation to the effect that system depth has on classification performance, and robustness against the influence of environmental noise during training. We do this in an effort to bridge together previous works, solidify the concepts of the methodology, and provide reasonable insight into how transferable the methodology is when applied to non-synthetic data. To further demonstrate the findings, we also analyse results of a case study using a subset of MNIST data. From this work, our experimental results support that an increase in system depth can lead to higher classification performance, as well as improved stability during training in noisy environments, with the sharpest performance improvements seemingly occurring between 1-3 uploading layer repetitions. Leading on from our experimental results, we also suggest areas that for further exploration, to ensure we can maximize classification performance when using the data re-uploading methodology.

INDEX TERMS Machine Learning, Quantum Computing, Quantum Machine Learning.

I. INTRODUCTION

Quantum machine learning is a rapidly expanding domain, bringing promising performance enhancements through complex feature space representations [1-5] and lowering computational complexity of equivalent classical algorithms by exponential factors in cases [6-11]. Variational quantum circuits (VQCs) are currently an area of large interest in the field [12-20], and provide a natural progression point for developing quantum algorithms due to their optimization capability.

VQCs often appear to be initialised using circuit structures and designs which are seemingly chosen at random, or have very little justification. Whilst this may work fine in certain scenarios, we need to look at what aspects of these circuits improve our performance, and whether certain features, such as the depth to our circuits, are most beneficial. Two measures of circuit capability referred to as 'expressability' and 'entangling capability' were explored initially in [21]. This was furthered in [22], where the performance of these circuits were compared in a classification setting. These studies suggest that expressability and performance of VQCs will start to plateau at a point, however this point may change dependent on the circuit used.

An encoding and classification strategy that has shown to be promising for an individual qubit is the concept of data re-uploading, introduced in [23]. Here, layers of parameterized gates are repeated to embed classical input data into Hilbert space. As a minimum, only a single qubit is required for classification, which makes this a promising methodology to pursue.

A critical aspect that should be explored when using data re-uploading is the correlation between circuit parameters and performance. These parameters could be considered as the number of qubits, entanglement usage and depth (i.e. the number of uploading layers used). The original proposal of data re-uploading partially explored these parameters, where added depth to the circuit did show performance increasing, before beginning to saturate. However, arguably there was not enough evidence to support that increasing depth, qubits or the use of entangling layers is always necessary to consistently improve upon performance.

Many QML algorithms are designed and tested with simulations. Whilst simulations can be effective in determining optimal performance, they leave an important factor of how the results of the proposed system may translate across to a real-world task through a quantum processing unit (QPU). An analysis in [24] took this into account, showcasing results processed using a QPU. However, little insight was provided into showing any correlation between circuit parameters and performance.

Ultimately, effective use of each qubit is especially important at the current NISQ era of quantum computation, as we are fairly limited by qubit cohesion and connectivity in QPUs. Many quantum algorithms rely on a moderate to large number of cohesive qubits to compute or encode inputs, which is not necessarily practical to use at the current time.

Also, it is especially important that we can maximise the working potential of each qubit used during computation, so that when the approach is extended to multiple qubits, the efficiency of the architecture is not affected. Doing so will not only allow us to understand the computational power that a single qubit possesses, but also provide an insight into effective VQC design, where each qubit can be maximally used. Because of the reasons outlined here, this work will focus on the use of a single qubit only.

Overall, recent works that explore data re-uploading described previously lack an important aspect which should be examined, such as correlations between system parameters and classification performance, or how the influence of noise affects classification performance. These are aspects which should be examined together in order to gain a full understanding of the methodology, and how this may translate to the wider field. Therefore, the aims of this work are to determine any correlations present between circuit parameters and performance, and to determine how this may translate to use in noisy environments, using a single qubit only.

Ultimately, the contribution of this work will be through an analysis of classification performance using the data re-uploading single qubit classifier. Through our experiments, we aim to identify key trends within system design, which can not only aid classification performance, but improve robustness of training in noisy environments. The work presented here will not only aid in our understanding of performance using the data re-uploading methodology, but how we can adapt our VQC design, to maximise the effectiveness of each available qubit dependent to the environment.

In order to achieve these aims, previous work will be bridged through an analysis of classification performance with varied circuit depths, using artificially generated datasets of incrementing difficulty. The resulting embeddings will be examined, where necessary, to give indications of how they change dependent to the input and design of the VQC. This will aid our search in determining effective embeddings of data, which are capable of producing higher-performing standards of classification. In addition, this will determine whether the methodology remains viable as the dimensionality of the task increases.

Alongside this, a case study will be conducted using MNIST data to provide a realistic indication of how the methodology may translate across to a scenario with non-artificial data. The inclusion of this will help to negate any biases that may have occurred due to the inclusion of artificially generated data.

In addition to the previous points, the methodology will be tested using a simulated noisy quantum environment. Doing so will help us to identify any design considerations that may assist convergence during training, and reach higher levels of performance.

The contribution of this work is through an in-depth analysis of the data re-uploading methodology. In this work, we identify general correlations between increased system depth supporting improved classification performance. Our experimental results also support that increasing system depth

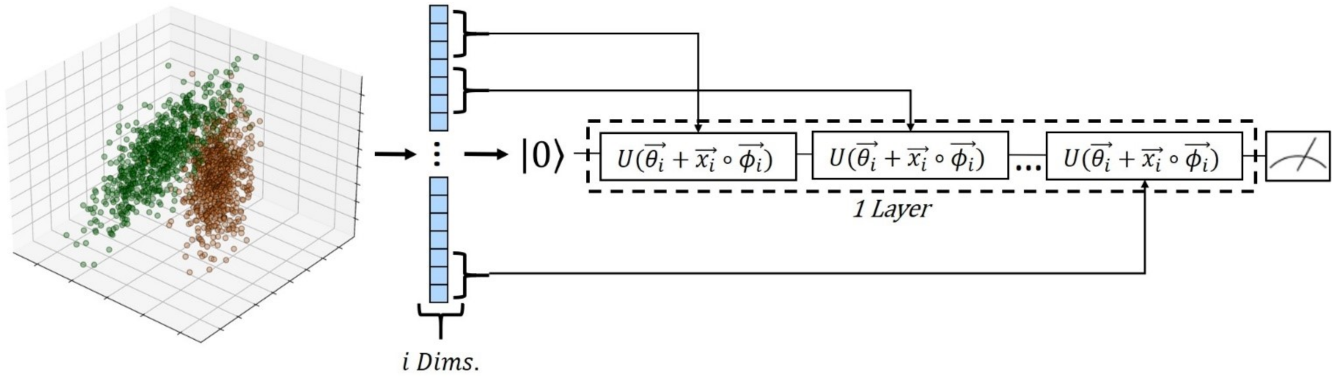
$|0\rangle$
$U(\vec{\theta_i} + \vec{x_i} \circ \vec{\phi_i})$
$U(\vec{\theta_i} + \vec{x_i} \circ \vec{\phi_i})$
...
$U(\vec{\theta_i} + \vec{x_i} \circ \vec{\phi_i})$
1 Layer
i Dims.

FIGURE 1. **Overview of the data re-uploading process from i-dimensional data-point to measurement. Firstly, the input data-point is recognized as a single column vector. Then, each input dimension is 'uploaded' by an arbitrary unitary gate, using a weighted sum of 2 rotational parameters per input. This process is repeated until each datapoint dimension has been encoded, where the qubit is finally measured with respect to a target state.**

may boost stability during training in noisy quantum environments, leading to better overall performance.

The structure of this paper is as follows. Firstly, we will briefly introduce the methodology of data re-uploading, to provide some background knowledge required. Then, an outline of our experimental setup and produced results will be described. Afterwards, an analysis of the produced results will be conducted, where we can identify key aspects in order to draw any conclusions.

II. METHODOLOGY

Within machine learning, we are often presented with data that is in the form of a column vector. Data re-uploading is a methodology in which we can encode these vectors into a feature Hilbert space using successive unitary operations, acting on each dimension of the input. For any $SU(2)$ operation U, we are able to decompose the operation into the following [36]:

$$U = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta) \tag{1}$$

Where $\alpha \in \mathbb{R}$ is the global phase factor, and $\beta, \gamma, \delta \in \mathbb{R}$ are the Euler angles that define each rotation. Here, we can then define the Euler angles as:

$$\begin{aligned} \beta &= \theta_i + x_i \cdot \phi_i \\ \gamma &= \theta_{i+1} + x_{i+1} \cdot \phi_{i+1} \\ \delta &= \theta_{i+2} + x_{i+2} \cdot \phi_{i+2} \end{aligned} \tag{2}$$

With θ and ϕ being weight parameters fed into our optimization loop, and x_i being the value of our input x at dimension i. These trainable weights define the extent to which the state of the qubit is rotated, with respect to the value of the input. From these parameter definitions, we can utilize a maximum of three input dimensions per unitary operation.

From here, we can cycle through our input vector, encoding a set of three data dimension values at a time, until the input vector has been fully encoded. In the proposed methodology, this is referred to as a full 'upload layer' of the data. By repeating this embedding of input data and adding successive uploading layers, a highly-complex feature Hilbert space can be created in an attempt to improve the learning capacity of the algorithm.

Once the input vector has been uploaded to the specified number of times, then the fidelity of the encoded quantum state is measured with respect to a target state. For each task, we pick a set of target states that are maximally distanced from each other, e.g., for a binary classification task, we could configure the target states for each datapoint of class 0 to be state $|0\rangle$, and each datapoint of class 1 to be state $|1\rangle$.

The loss function that is used throughout this work is based on the weighted fidelity loss function defined in [23], however we exclude the individual class weightings. Defined in eq. 3, the loss function aims to minimize the fidelity of datapoints between their current state and respective target states, where θ and ϕ are parameters to be optimized, x is the input data, D is the size of the training/validation dataset and C is the number of classes.

$$\frac{1}{2}\sum_{d=1}^{D}\sum_{c=1}^{C}\left(F(\vec{\theta}, \vec{x}_d, \vec{\phi}) - F_c(\vec{x}_d)\right)^2 \tag{3}$$

A full, detailed description of this method can be found at its' proposal in [23], however a brief example is detailed in figure 1 for simplicity.

III. RESULTS

A. EXPERIMENT SETUP

1) BARREN PLATEAU PROBLEM

As a preface before describing the following experiments, it is relevant to address the barren plateau problem that is largely present when training VQCs and the effect that this has if not considered. Barren plateaus are areas of near-zero gradient within the loss landscape that, if not considered, can substantially affect the training of VQCs and not allow for stable convergence to a minimum in sufficient time.

This problem was addressed in [25], where several approaches have been considered since to avoid this problem, such as local cost functions [26, 27], evaluating initialized weights [28], or the use of quantum natural gradient [29, 30].

For the following experiments, the problem of barren plateaus is considered by initializing 10 randomized weight sets, where all weight sets are initialized using a Gaussian distribution, with a mean of 0 and standard deviation of 0.1. We use these values as within preliminary experiments, they produced more consistency between training samples than other weight initialization values.

For each weight set, a single epoch was conducted on the test dataset. The parameter set that produced the lowest test loss value initially was then used thereafter throughout training. Whilst this helped to avoid the problem of barren plateaus in our experiments, it should be pointed out that this is a temporary solution to the problem only and alternative measures should be analyzed for a better solution to avoiding the barren plateau problem.

2) EXPERIMENTATION PLAN

To outline the following results, the concepts targeted by this work will be addressed in order. Firstly, to consider how depth affects performance of the data re-uploading scheme, the number of layers used within the system will be incremented from 1 to 10. Alongside this, data dimensionality will be increased from 3 up to 15 in increments of 3 to determine the effect of increased classification difficulty. This will be extended from binary to multi-class classification tasks, in order to provide reasonable assumptions on how this performance may translate to other tasks.

Secondly, within QPUs, quantum noise from external factors (e.g. environmental) can negatively impact the output

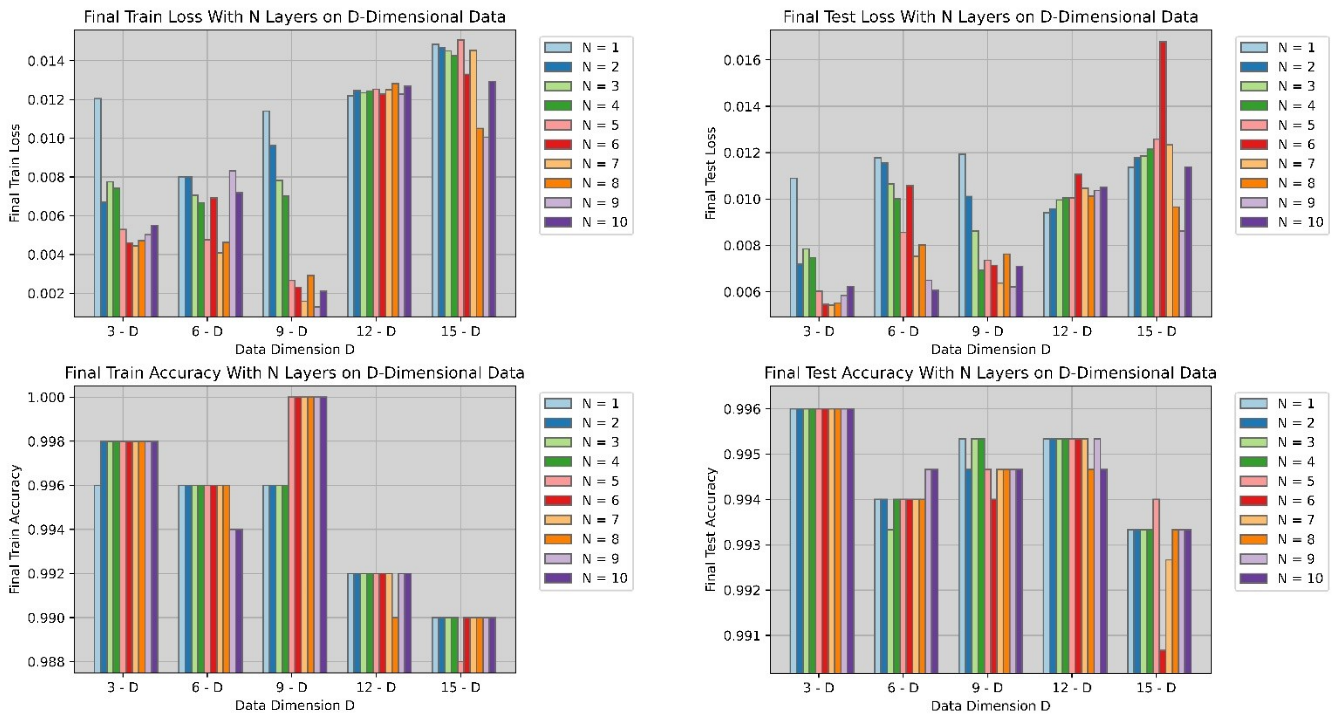

FIGURE 2. Result charts of a binary classification task, using layer depths N of 1-10 per each dataset dimension D. The chart displays training set loss (top-left), test set loss (top-right), training set accuracy (bottom-left) and test set accuracy (bottom-right).

of the quantum system executed. To account for this, simulated noise of varied strength will be included, in order to determine the robustness per system depth used.

Thirdly, a case study of non-synthetic data (MNIST) will be included to give a realistic indication of the performance that the methodology may bring to a real-world task.

For all of the following experiments, the PennyLane library [31] was used, alongside the PyTorch interface. For non-noisy environments, the Qulacs [32] qubit simulator plugin was used within PennyLane, and for noisy environments, PennyLanes' mixed state simulator was used. For reproducibility, all

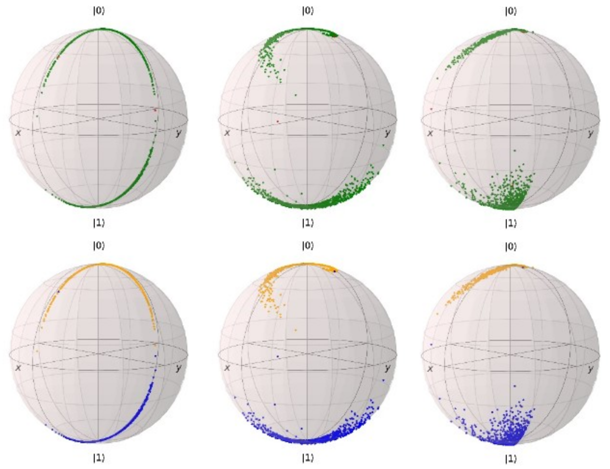

FIGURE 3. Bloch sphere visualizations of test set embeddings at epoch 30 with varied system depth. The top row represents correctly classified points (green) versus incorrectly classified points (red). The bottom row represents the distribution of classes (different colour per different target class value). Left to right on both rows is the system depths of 1 layer, 7 layers and 10 layers respectively.

relevant randomization seeds were set to zero, unless stated otherwise.

All artificially-generated data was natively handled using scikit-learn [33], as this allowed for much greater flexibility in defining the data and features used appropriately. For each dataset, unless otherwise specified, the parameters were set to use 1 redundant feature, 2 informative features, 1 cluster per class, a class separation of 1 and a random generation seed of 1234.

B. LAYER CORRELATIONS

For the following results, artificial datasets were generated, consisting of 500 train and 1500 test images split evenly between the number of classes used. 30 epochs of training were used per experiment, using SGD optimizer and a learning rate of 10^{-2}, unless otherwise stated. We use these hyperparameters as from preliminary experiments, this produced much more stable convergence on average than higher learning rates, whilst reducing the computation time per experiment in comparison to lower learning rates with additional epochs.

Figure 2 shows results of a binary classification task, where the depth of system (i.e. number of layers) is varied from 1 to 10 and the dimensionality of the dataset increased from 3 to 15 dimensions, in intervals of 3 dimensions. The generally expected behaviour here would be for the overall trend of performance per layer over each dimension to worsen, due to the scaling of difficulty of the task, with each individual dimensional groups' performance improving as layers are added, giving increased learning capacity to the system.

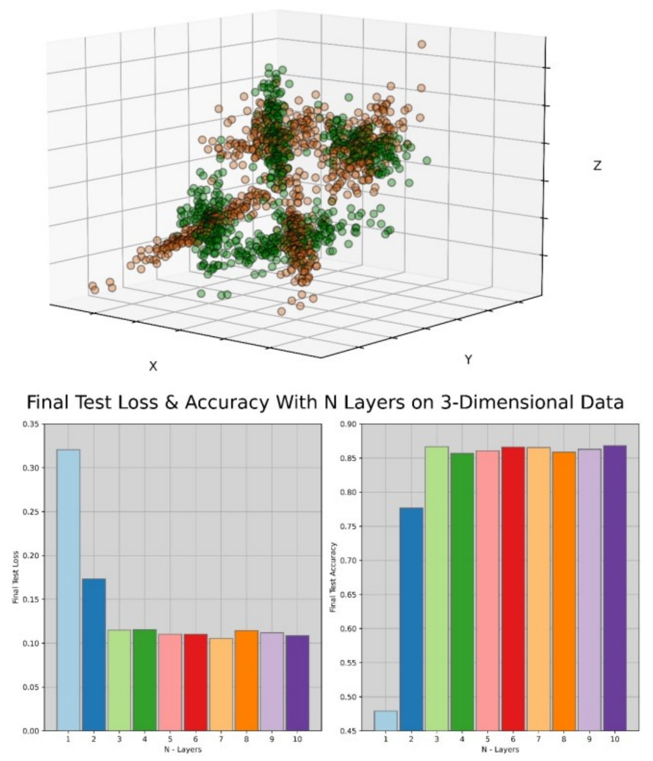

FIGURE 4. Top – Euclidean space view of the 3-dimensional dataset used, with considerably more overlap between class clusters. Bottom – Charts displaying test set loss (left) and test set accuracy (right) at epoch 10, with varied system depth between 1-10 layers.

The chart displaying test loss in figure 2 is fairly consistent with this behaviour until 12 dimensions are reached. From here, the behaviour almost reverses, where the performance of the system does not improve with additional layers until after 6 layers, where it appears to plateau.

Focusing on 3-dimensional loss results, fairly consistent performance increases can be seen with added depth until 7 layers, where performance starts to regress and worsen thereafter. This behaviour is not unique and happens on more than one occasion. Regardless of the slight regression within the loss value, the system still classifies the vast majority of the test set correctly, and does not change throughout the different depth values implemented.

Figure 3 displays embeddings of test data using Bloch sphere visualizations, with layer depths of 1, 7 and 10 at epoch 30. Regardless of the number of layers, it can be seen that the system still classifies each point correctly, minus the outliers which are heavily nested inside the opposing classes cluster.

However, figure 3 shows the advantage brought by an increased system depth, that of allowing for more complex mappings of data. A depth of 1 layer produced a fairly linear embedding in this case, where the distribution of points is quite narrow along that particular rotational line, whereas 10 layers allowed for rotations to occur in the embedding, and thus form a more complex feature space.

In the case of the data used here, perhaps only a simpler complexity of embedding was needed to separate the clusters and classify them to a high degree of accuracy. However, for data that is not so separable with an intensified overlap between clusters of datapoints, the advantage of increased depth may become more apparent.

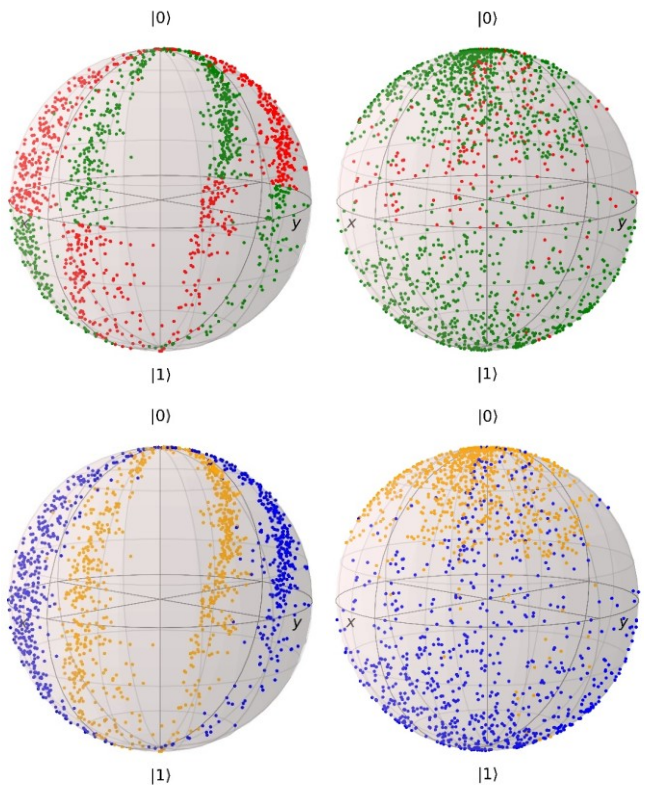

FIGURE 5. Bloch sphere visualizations displaying embeddings of the test dataset shown in figure 4. The top row represents correctly classified points (green) and incorrectly classified points (red). The bottom row show the distribution of classes (different colour per different target class value). Left to right on both rows is a system depth of 1 and 10 layers, respectively.

Figure 4 shows results of a binary classification task using a dataset with an extreme overlap between class clusters, more so than the data used previously. This dataset was generated using 3 informative features, 4 clusters and a class separation of 2. The dataset consisted of 500 train and 1500 test images split evenly between the 2 classes. Each experiment was trained for 10 epochs, using stochastic gradient descent optimization and a learning rate of 10^{-3} to avoid overfitting in this case.

Again, it can be seen that improvements in performance occur with an increased system depth, however these improvements begin to saturate after approximately 3 layers. In comparison to previous results, these performance increases are much more stable, and any regressions in performance with additional depth are at a smaller scale. In addition, there are a much higher proportion of misclassified points using a lower number of layers, when compared to the previous results gathered and displayed in figure 2.

If we look to the embeddings of the test set data displayed in figure 5, the embedding capability of a single layer is much more rigid and restricted in comparison to 10 layers, which allows for a greater degree of flexibility in its' mapping of

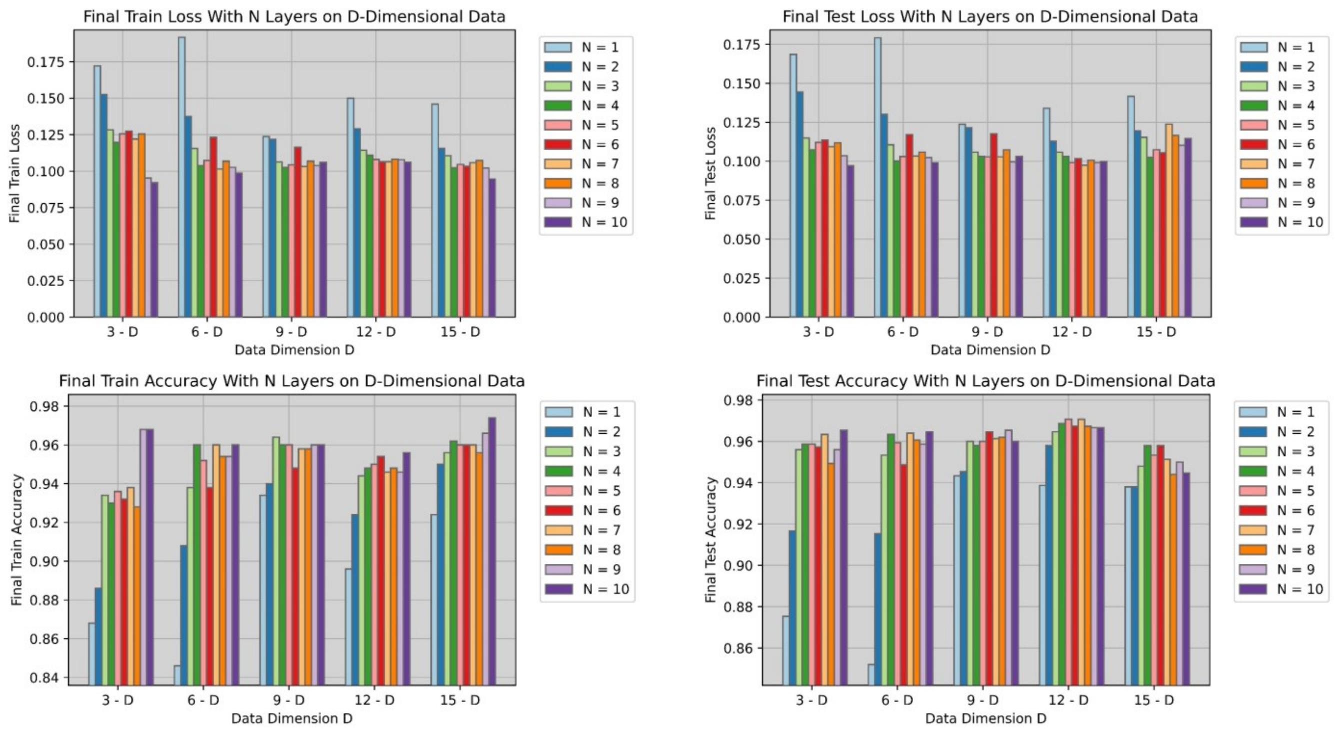

FIGURE 6. **Result charts of a 3-class classification task, using layer depths N of 1-10 per each dataset dimension D. The chart displays training set loss (top-left), test set loss (top-right), training set accuracy (bottom-left) and test set accuracy (bottom-right).**

data. This flexibility results in the system being able to separate each class more effectively.

In the case of the first experiment, a system depth of a single layer, requiring 6 parameters in total, was sufficient to perform to a high standard. Whilst increasing depth generally improved the confidence of these scores up until 7 layers, these additional layers were unnecessary to determine much better performance and just increased complexity.

For the secondary dataset with a severe overlapping between data clusters, a much more complex level of embedding was needed to classify the dataset to a good standard, which was not sufficiently found until 3 layers and onwards. The performance increases between layers 1, 2 and 3 here in figure 3 are much bigger in proportion to those shown previously in figure 2.

However, as we increase the number of classes within the classification task, the boundaries for each class region on the Bloch sphere will become smaller when using a single qubit. For a multi-class task, higher levels of embedding flexibility than that of a binary task may be required to effectively map each datapoint to their respective class region.

Figure 6 displays 4 result charts gathered from a multi-class classification task, consisting of the default data generation scheme described earlier, with 3 datapoint classes. Looking at the general behaviour between depth and data dimensionality, straight away it can be seen that there is a correlation between depth and performance. The correlation shown here is arguably much stronger than that of the initial set of results displayed previously in figure 2.

Whilst performance improvements can be seen with added system depth, these improvements do saturate and begin to plateau at a point. On average, the sharpest increases to performance occur between 1-3 layers, and quickly plateau thereafter. As before, there are cases where performance starts to regress, such as test set performance using 15-dimensional dataset. However, as there is a spike at the corresponding train set performance, it is unclear whether aspects of these performance regressions are due to slight overfits towards the training data.

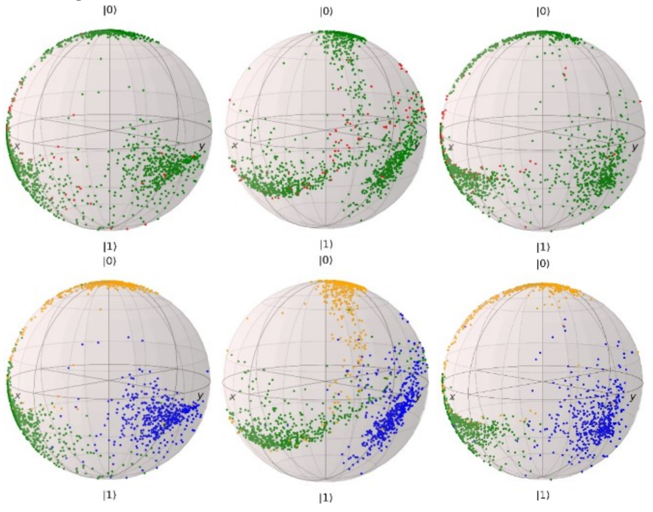

FIGURE 7. **Bloch sphere visualizations displaying embeddings of the test dataset used for results shown in figure 6. The top row represents correctly classified points (green) and incorrectly classified points (red) at epoch 30. The bottom row displays the distribution of classes at epoch 30, with each colour representing a different class. From left to right on both rows is a system depth of 5, 6 and 7 layers, respectively.**

In figure 2, the classification accuracy using a single layer was much higher in proportion to successive layers than in the case of results shown in figure 6. This suggests that the data here was much harder to classify to a high standard, where embedding complexity is a key feature in determining classification performance.

Upon a closer review, there are cases where test set performance starts to regress. This is apparent with 6-dimensional and 9-dimensional data results, where a spike occurs when 6 layers are used. In an inspection of the corresponding embeddings displayed in figure 7, we can see that the embedding of layers 5 and 7 are very similar. However, the data distribution formed from 6 layers is closer together.

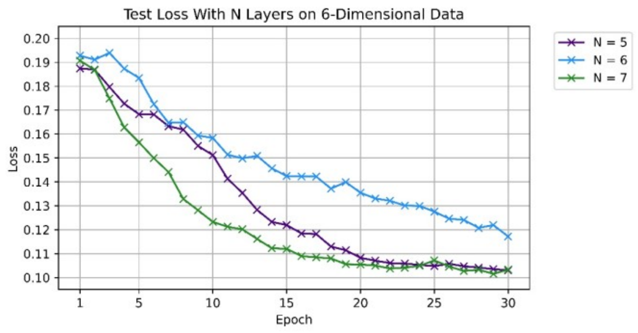

FIGURE 8. Plot of test set loss result per epoch using 6-dimensional data and a system depth of 5, 6 and 7 layers.

Figure 8 displays test set loss values over training for a system depth of 5, 6, and 7 layers using 6-dimensional data. Here, it can be seen that layer 6 has a much slower convergence rate in comparison to layers 5 and 7 in this scenario. However, where the curves for 5 and 7 layers appear to plateau, the curve for 6 layers is still steadily decreasing. This implies that the initial weights for layer 6 may have been initialized in a region of lower gradient than the weights of layer 5 and 7. This would cause a delay in convergence, similar to the behaviour in figure 8. If training were to be continued, then it is likely that the curves would meet at roughly the same boundary between loss values of 0.10-0.11.

Whilst weight initialization may a factor in the drop of performance in this case, it is difficult to state that this factor caused performance drops in other cases throughout this work. For example, there are cases of performance regression occurring, i.e., test set performance using 15-dimensional data at 10 layers displayed in figure 6. However, there is also a drop in the corresponding train set performance, which suggests that this may have been a slight overfit causing the drop in performance and not related to the mapping of data or initialization of weights.

Overall, the previous results support that increasing system depth does generally improve classification performance on average, with the biggest improvements usually occurring between depth increments of 1, 2 and 3 layers. From visualizing the differences between depth increments, a clear advantage that increased depth has is by being able to produce much more complex mappings of data.

However, in cases, increased depth does not necessarily relate equally to improved performance. This implies that an optimal depth is data dependent, where depending on the complexity of the task more layers are needed to effectively separate each class cluster towards their respective target states. Due to the innate randomness of weight initialization, it is hard to justify the impact this this or other reasons had on performance, i.e., whether any drops in performance were related to suboptimal initial weights, slight overfitting to training data or purely from the depth specified at the time.

C. INTRODUCTION OF NOISE

To simulate the effect of noise during training in a QPU environment, amplitude damping channels are implemented within the system after each unitary gate. Whilst there are many quantum noise channels which could be used to simulate noise (e.g. bit-flips, de-phasing and depolarizing channels), amplitude damping was implemented as it provides a realistic noise model, and is frequently used to model noise within other works [5, 34]. Amplitude damping is a model of qubit energy relaxation through interactions with the environment over time. The result Q with decay probability $\gamma \in [0,1]$ of Kraus operators K acting on the density matrix ρ is:

$$Q_\gamma(\rho) = K_0 \rho K_0^\dagger + K_1 \rho K_1^\dagger \tag{4}$$

Where:

$$K_0 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{bmatrix}, \quad K_1 = \begin{bmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{bmatrix} \tag{5}$$

The effect that amplitude damping has on the qubits' density matrix can be defined as:

$$Q_\gamma(\rho) = \begin{pmatrix} \rho_{00} + \gamma\rho_{11} & \sqrt{1-\gamma}\rho_{01} \\ \sqrt{1-\gamma}\rho_{10} & (1-\gamma)\rho_{11} \end{pmatrix} \tag{6}$$

For all experiments using simulated noise, amplitude damping is implemented within the mixed state simulator available through the PennyLane library. More information on the amplitude damping channel can be found at [35].

For the following experiments, 3 and 15-dimensional artificially-generated data was initialized, with a train to test image split of 50 to 150 datapoints per class. For each dataset dimensionality, the noise magnitude was incremented from 0 to 1, in intervals of 0.1 and the circuit depth was also increased. Each training session consisted of 30 epochs of training, using stochastic gradient descent for optimization and a learning rate of 10^{-2}.. In each experiment, final loss values at epoch 30 were taken, and the change between these values per noise magnitude λ was recorded.

As we implement simulated noise after each parametrized gate, as the defined number of layers and task dimensionality increases, naturally the occurrences of noise will increase proportionally. Therefore, we measure the change in loss in proportion to the occurrences of noise within that particular

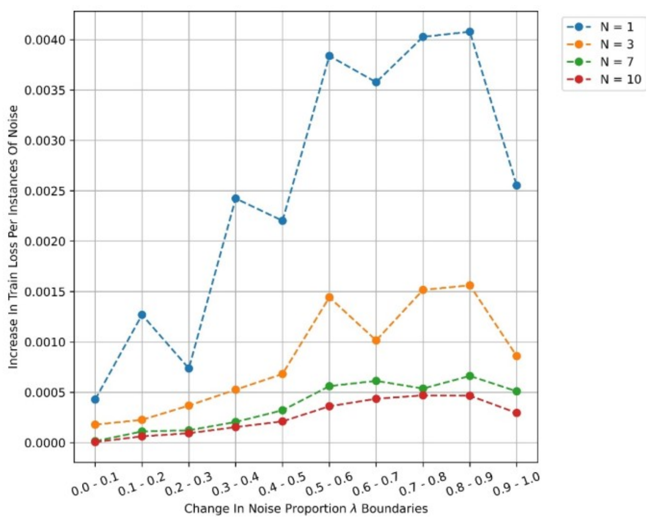

FIGURE 9. Plot displaying the proportional change in final training loss values between 0.1 intervals of noise strength values λ for N layers. The results shown are for a binary classification task, using 3-dimensional data.

circuit. This avoids any unfair advantage that a lesser depth circuit may possess, since noise would naturally be implemented less than a circuit with a higher depth.

Within figure 9, it can be seen that for lower circuit depths of 1 and 3, the increases within the final training loss value are much bigger in comparison to a larger circuit depth, as noise magnitude increases. However, as the number of layers is increased, this rate of change does begin to saturate. In this case, the visible drop-off in training loss to the right-hand side can be justified. For a binary classification task, the associated target states will be located on opposing points on the Bloch sphere, or states $|0\rangle$ and $|1\rangle$ for simplicity. As noise magnitude increases, the distribution of datapoints will be drawn closer towards the $|0\rangle$ state. If the noise is extremely strong, then the datapoints with target state $|1\rangle$ will be very far away, unable to be drawn further away. As loss is calculated using the measure of fidelity between states, this explains the corresponding drop off.

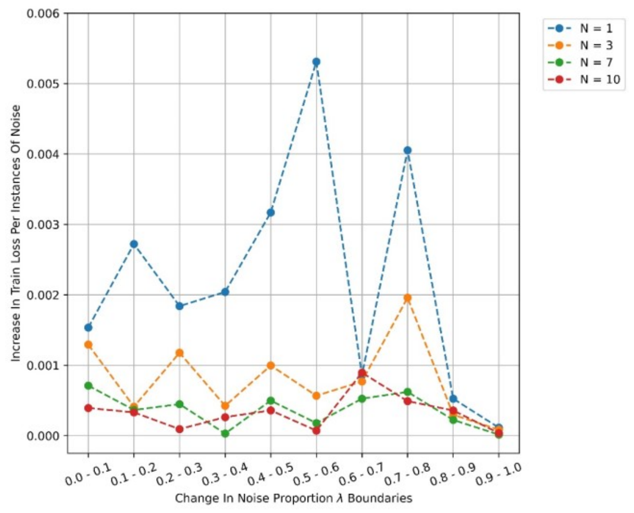

FIGURE 10. Plot displaying the proportional change in final training loss values between 0.1 intervals of noise strength values λ for N layers. The results shown are for a binary classification task, using 15-dimensional data.

Again, as shown in figure 10, the rate of increase is much larger for lower numbers of layers used, even with a larger dataset dimensionality of 15. In this case, a drop can again be seen to the right-hand side of the chart, as a result of the decrease in fidelity between datapoints and their target states slowing down.

From figures 9 and 10, these results suggest that using a circuit of larger depth may perhaps bring an advantage of robustness against the influence of noise during training. Whilst the benefits of this did appear to saturate as we got closer to a layer depth of 10, these results do suggest that using additional layers may allow for a better quality of training, by resisting the influence of environmental noise, ensuring that the training can converge more stably.

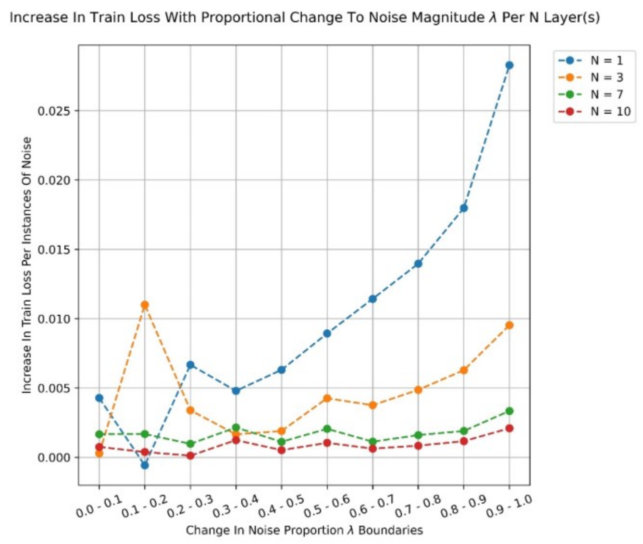

FIGURE 11. Plot displaying the proportional change in final training loss values between 0.1 intervals of noise strength values λ for N layers. The results shown are for a 3-class classification task, using 3-dimensional data.

Figure 11 displays the proportional change to training loss for a 3-class classification task using 3-dimensional data. Here, a much sharper performance regression can be seen for 1 layer as noise magnitude increases. Regardless of noise increments, 7 and 10 layers show a fairly stable level of increase in loss, only showing signs of divergence from $\sim\lambda$=0.8.

In contrast to previous results, there is a diverging behaviour to the right-hand side of the charts. As the results in figure 11 were produced from a 3-class classification task, the maximally-distanced target class states are distributed more heavily away from the $|0\rangle$ state in our setup. Because datapoints are drawn closer towards the $|0\rangle$ state as noise magnitude is increased, the associated loss value will increase at a higher rate than that of a binary classification task due to a larger cumulative distance between each datapoint and its' target class state.

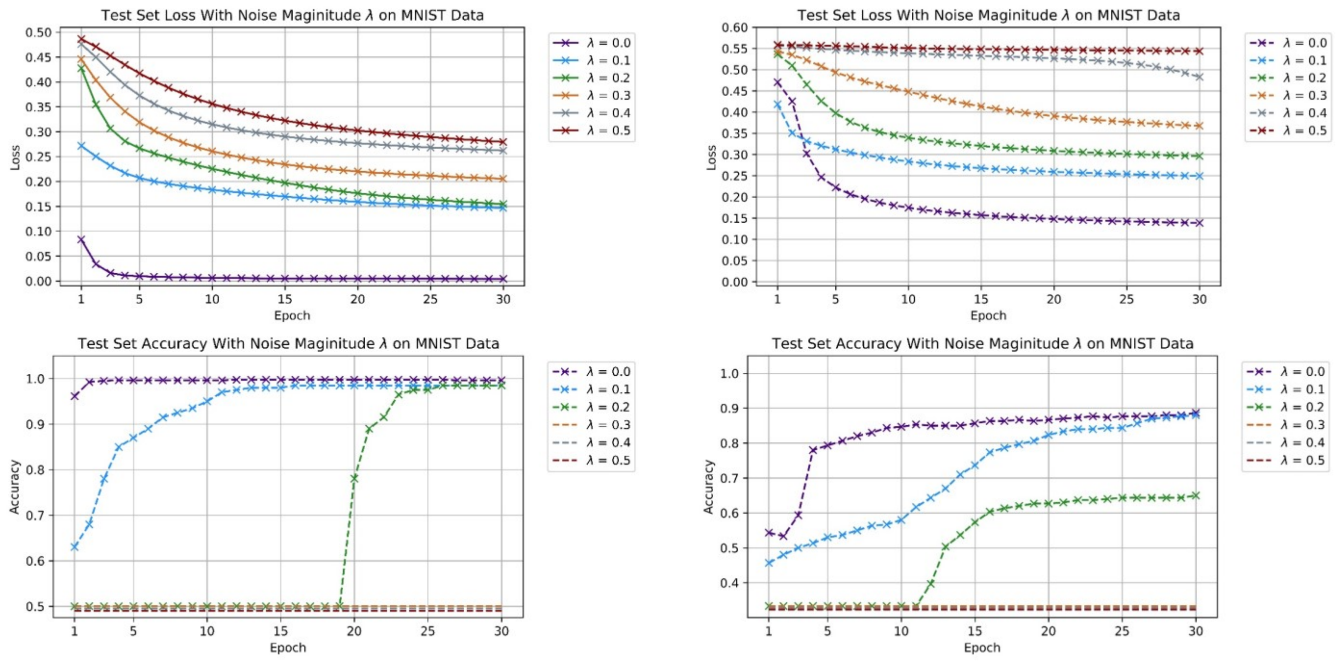

FIGURE 13. **Figure displaying results from binary and multi-class experiments using downsampled MNIST data. Left – Test set loss (top) and test set accuracy (bottom) results from a binary classification task (classes 0 and 1), with varied noise magnitude λ and a system depth of 1 layer. Right - Test set loss (top) and test set accuracy (bottom) results from a 3-class classification task (classes 0, 1 and 2), with varied noise magnitude λ and a system depth of 1 layer.**

Similar behaviour can be seen in figure 12, showing results from a 3-class classification task, using 15-dimensional data. Here, layers 7 and 10 show consistently lower changes in comparison to lower layers of 1 and 3. However, again there is a much larger difference between 1 layer and 3 layers than 7 layers and 10 layers. This supports that whilst additional layers may provide added robustness against the effects of noise during training, this benefit does saturate as more and more layers are implemented to the system.

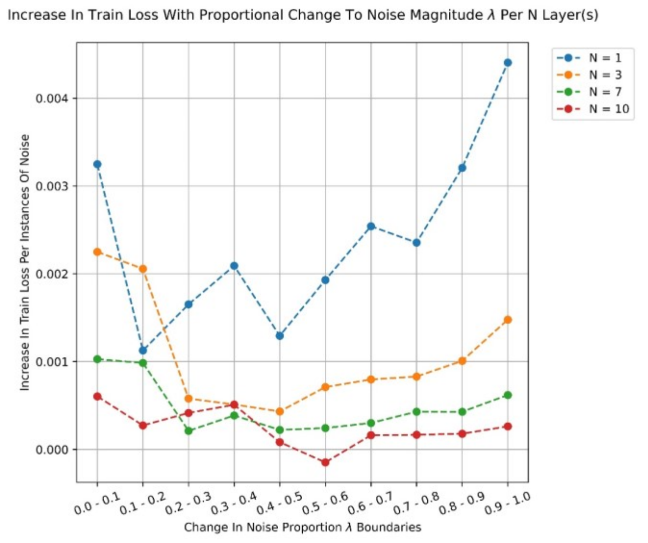

FIGURE 12. **Plot displaying the proportional change in final training loss values between 0.1 intervals of noise strength values λ for N layers. The results shown are for a 3-class classification task, using 15-dimensional data.**

Overall, the results displayed within this section support the possibility of an advantage of robustness against noise during training, when additional layers are implemented to the system. Whilst plots showed a higher variance of training loss between noise intervals using 15-dimensional data (shown in figures 10 and 12) over 3-dimensional data (shown in figures 9 and 11), a higher depth consistently showed more stability throughout experiments, rather than the diverging behaviour seen in lower layer depths.

However, similar to results seen previously throughout section 3B, any advantage of robustness during training appeared to plateau as depth increased. The sharpest of improvements could be seen between 1 layer used and 3 layers used. The differences between 7 layers and 10 layers used were minimal, and arguably not worth the increase in complexity that additional layers would bring.

D. MNIST CASE STUDY

Previously, experiments have been conducted using artificially-generated datasets. Whilst this is acceptable for examining specific details surrounding performance, it does not always give a realistic representation of how the algorithm may perform on a non-artificial dataset.

For this reason, the following results will be from experiments using MNIST data. For these experiments, a subset of the MNIST dataset was used, consisting of 200 images per class within the training set, and 100 images per class within the testing set. Each image used was normalized and downsampled to a size of 9x9, in order to reduce the processing time required.

For the hyperparameter choice, a system depth of a single layer was used, again to reduce processing required. For optimization, Adam [37] was implemented with a learning rate of 10^{-4}. Each experiment was trained for 30 epochs, with test set results taken at the end of each training epoch.

To give a representation of noise impact on non-artificial data, the same schematic for noise introduction was used as outlined in previous results. 6 experiments were conducted per dataset, with magnitudes of noise increasing from no noise to a value of 0.5 in intervals of 0.1.

By looking at the results of a binary classification task (using digit classes 0 and 1) displayed on the left-hand side in figure 13, it can easily be seen that the experiment with zero noise performed to an excellent level within 5 epochs. As the noise magnitude is increased, the system is still able to classify the test dataset to an excellent standard between noise values of $\lambda = 0.1$ and $\lambda = 0.2$. However, the loss value begins to converge to approximately 0.15, making this a more realistic level than the loss result with zero noise influence.

For results with $\lambda = 0.2$, a delay in accuracy increase can be seen, where improvements do not occur until approximately epoch 19. As the loss value is decreasing at a satisfying rate, this delay can be justified from the distribution of all datapoints residing in a single state region only until this time. Once a noise magnitude of 0.4 is reached, the system is unable to produce any effective encoding of data and performance is at a minimum throughout.

By looking at results using 3-classes (digits 0, 1 and 2), displayed on the right-hand side of figure 13, the performance is further decreased than when classifying 2 classes only, with a max test set accuracy of 88.7% with no noise influence. As noise in increased to a value of $\lambda = 0.1$, similar levels of performance are reached, with a final test set accuracy value of 88%, however convergence is much slower than when no noise is present.

As noise becomes stronger to $\lambda = 0.2$, performance starts to plateau after epoch 15 despite a sharp improvement between 11-15 epochs. Similar to binary classification results, once the level of noise reaches a magnitude of $\lambda = 0.3$, the system is unable to perform at a level better than random, since all datapoint will be located in a single target region.

Overall, the results shown in in this section using MNIST data are very good within the binary classification task, for noise levels up to $\lambda = 0.2$, even with the increase in convergence time. When extended to a 3-class task, performance did have a substantial drop, however, the target state boundaries are also lesser as additional classes are used.

For both tasks, as noise reaches higher levels, then the system is unable to cope with the influence brought and is unable to converge to a level different than random guesses. These results were with a single uploading layer only, where previous results show adding extra depth may provide the robustness and complexity of embedding needed to classify the datapoints to a higher accuracy.

IV. DISCUSSION

The aim of this work is to bridge knowledge between previous works, determine any correlation between system depth and performance using the data re-uploading methodology, test robustness of the system when using different depths, and finally provide an indication of how this methodology may perform on non-artificially generated data.

From the results gathered within this work, we have identified a general trend where increasing depth does tend to improve upon previous performance. However, the sharpest performance benefits seem to occur between 1-3 uploading layers. After approximately 3 layers were introduced, any performance increases were often not as distinctive, where the increased depth just added complexity to the system with little reward in performance.

In cases, (e.g. the case of 10 layers trained on 15-dimensional data, displayed in figure 6), it could be seen that increasing depth did not relate to improved performance. In some of these cases, this could be justified from factors such as slight overfitting to the training data used, or perhaps from other factors, such as the initial weight selection. In these cases, it is hard to determine whether any regression in performance occurred solely from the selected depth, or from the influence of other factors.

A large advantage shown with an increased depth is the allowance for more complex embeddings of data. Lower circuit depths with lower total parameters were fairly rigid in their embedding capability, therefore restricting the freedom of movement needed in order to effectively separate the overlapping data clusters.

When examining from a perspective of noise, our results support that a higher system depth could be linked to robustness of noise during training. In comparison to lower system depths, higher depths had consistently smaller proportional changes as noise magnitude increased, therefore providing a more stable platform to train from. However, as with general classification performance, this advantage of robustness did saturate as additional layers were implemented, with the sharpest robustness improvements generally occurring between 1-3 layers.

In the case of experiment results using MNIST data, the data re-uploading methodology showed promising binary classification results, using only a single layer of 162 total parameters. As the levels of noise increased here, good performance was still achieved with lower noise levels, but was unable to converge after a noise magnitude of $\lambda = 0.3$.

Whilst the results gathered on MNIST data may not be state-of-the-art, it should be considered that this performance was achieved using a single layer and a single qubit only. Therefore, these results are fairly promising in relation to the early state that QML is in. As this methodology expands to multiple qubits of larger depth, then this performance can only hope to be improved upon, and extended to much higher numbers of classes.

In the wider field of QML, we are able to link insights gathered from our experimental results to other relevant works directed towards VQC design and implementation. In [21], expressability of a qubit was determined by its ability to navigate the Bloch sphere, which was also analysed in [22]. Our results support the idea that increased embedding complexity, which relates to the expressability of a qubit, can allow for the complex feature spaces needed to separate entangled clusters of datapoints.

Therefore, by improving our embedding complexity, or expressability in a VQC, we can have a much higher capability in classifying difficult, overlapping datapoint clusters to a good level of performance, compared to if we did not consider this in our design. However, once we have reached a sufficient level of embedding complexity, or expressability, then adding additional depth may just increase computational complexity for little performance improvement in return.

From this work, there are some limitations and areas for future exploration that should be addressed alongside the described contributions. Whilst our experimental results showed trends appearing, in cases it was hard to justify whether performance differences were influenced by other factors such as initial weight selections. Whilst our weight initialization strategy was kept constant throughout, it was not necessarily an optimal choice. Currently, there have been some efforts to address weight initialization strategies, in association with avoiding barren plateaus [28]. However, it still remains an open question of whether there are any optimal initialization strategies that may benefit the training of VQCs, and specifically when using the data re-uploading methodology.

When using the data re-uploading methodology with a single qubit only, as we increase the number of classes used, then the corresponding class regions within the Bloch sphere also become reduced in area. In the original proposal of the methodology [23], the authors presented the use of multiple qubits, which naturally introduce larger state boundaries per class than when using a single qubit. In order to do this, entangling layers of CZ gates were introduced to give the dependency needed between qubits. However, this also leaves room to explore the effects from different implementations of entanglement measures, and whether there is an optimal setup for introducing multiple qubits.

V. CONCLUSION

In this work, we have conducted an analysis of the data re-uploading methodology, using a single qubit only. Multiple values of depth were used throughout this work, in order to give an indication of how this parameter affects classification performance. We also introduced simulated noise to determine any key features that are beneficial in providing robustness and stability during training.

Here, our experimental results support that increasing depth does improve classification performance, with the sharpest improvements occurring between approximately 1-3 layers used. A clear advantage displayed is that the complexity of data embedding improved alongside increased depth, which allowed for highly overlapping datapoint clusters to separate more effectively. However, our results also suggest that once a sufficient level of embedding complexity is found, then additional depth may just increase complexity with little performance benefit rewarded in return.

In the case of our noise simulations, our results suggest that higher depth values may allow for improved stability during training. However, extreme levels of noise will continue to have extreme consequences, due to the nature of the algorithm and how predictions are measured.

Considering limitations and directions of future work, we suggest that studies should be conducted into favourable weight initializing strategies for avoiding barren plateaus, to assist in stable training convergence. Alongside this, as we extend to using multiple qubits, how we introduce measures of entanglement may provide a substantial role in determining overall classification performance and should be explored further.

VI. REFERENCES

[1] M. Schuld and N. Killoran, "Quantum Machine Learning in Feature Hilbert Spaces," 2019. doi: 10.1103/PhysRevLett.122.040504.
[2] V. Havlíček *et al.*, "Supervised learning with quantum-enhanced feature spaces," 2019. doi: 10.1038/s41586-019-0980-2.
[3] C. M. Wilson *et al.*, "Quantum Kitchen Sinks: An algorithm for machine learning on near-term quantum computers," 2018. Accessed: May 13, 2020. [Online]. Available: http://arxiv.org/abs/1806.08321.
[4] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, "Quantum embeddings for machine learning," 2020. Accessed: Jun. 02, 2020. [Online]. Available: http://arxiv.org/abs/2001.03622.
[5] R. LaRose and B. Coyle, "Robust data encodings for quantum classifiers," 2020. Accessed: Sep. 28, 2020. [Online]. Available: http://arxiv.org/abs/2003.01695.
[6] Y. Dang, N. Jiang, H. Hu, Z. Ji, and W. Zhang, "Image classification based on quantum K-Nearest-Neighbor algorithm," 2018. doi: 10.1007/s11128-018-2004-9.
[7] V. Dunjko, J. M. Taylor, and H. J. Briegel, "Quantum-Enhanced Machine Learning," *Phys. Rev. Lett.*, vol. 117, no. 13, 2016, doi: 10.1103/PhysRevLett.117.130501.
[8] C. Ding, T.-Y. Bao, and H.-L. Huang, "Quantum-Inspired Support Vector Machine," 2019. Accessed: Jun. 01, 2020. [Online]. Available: http://arxiv.org/abs/1906.08902.
[9] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum principal component analysis," *Nat. Phys.*, vol. 10, no. 9, pp. 631–633, Jul. 2014, doi: 10.1038/NPHYS3029.
[10] P. Rebentrost, A. Steffens, I. Marvian, and S. Lloyd, "Quantum singular-value decomposition of nonsparse low-rank matrices," 2018. doi: 10.1103/PhysRevA.97.012327.
[11] P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big data classification," 2014. doi: 10.1103/PhysRevLett.113.130503.
[12] E. Farhi and H. Neven, "Classification with Quantum Neural Networks on Near Term Processors," 2018. Accessed: May 30, 2020. [Online]. Available: http://arxiv.org/abs/1802.06002.
[13] A. Mari, T. R. Bromley, J. Izaac, M. Schuld, and N. Killoran, "Transfer learning in hybrid classical-quantum neural networks," Dec. 2019,

Accessed: Apr. 28, 2020. [Online]. Available: http://arxiv.org/abs/1912.08278.

[14] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, "Circuit-centric quantum classifiers," *Phys. Rev. A*, vol. 101, no. 3, Apr. 2020, doi: 10.1103/PhysRevA.101.032308.

[15] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," 2019. doi: 10.1038/s41567-019-0648-8.

[16] 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," 2019. Accessed: May 21, 2020. [Online]. Available: http://arxiv.org/abs/1907.00397.

[17] M. Henderson, S. Shakya, S. Pradhan, and T. Cook, "Quanvolutional neural networks: powering image recognition with quantum circuits," 2020. doi: 10.1007/s42484-020-00012-y.

[18] J. M. Liang, S. Q. Shen, M. Li, and L. Li, "Variational quantum algorithms for dimensionality reduction and classification," 2020. doi: 10.1103/PhysRevA.101.032323.

[19] Y. Liu *et al.*, "Variational quantum circuits for quantum state tomography," 2020. doi: 10.1103/PhysRevA.101.052316.

[20] J. Bausch, "Recurrent Quantum Neural Networks," 2020.

[21] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, "Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms," 2019. doi: 10.1002/qute.201900070.

[22] T. Hubregtsen, P. Josef, J. Pichlmeier, · Patrick Stecher, and · Koen Bertels, "Evaluation of Parameterized Quantum Circuits: on the relation between classification accuracy, expressibility and entangling capability." Accessed: Sep. 28, 2020. [Online]. Available: https://www.researchgate.net/publication/340115185.

[23] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, "Data re-uploading for a universal quantum classifier," *Quantum*, vol. 4, p. 226, Jul. 2020, doi: 10.22331/q-2020-02-06-226.

[24] W. Cappelletti, R. Erbanni, and J. Keller, "Polyadic Quantum Classifier," 2020. Accessed: Sep. 28, 2020. [Online]. Available: http://arxiv.org/abs/2007.14044.

[25] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nat. Commun.*, vol. 9, no. 1, 2018, doi: 10.1038/s41467-018-07090-4.

[26] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, "Cost-Function-Dependent Barren Plateaus in Shallow Quantum Neural Networks."

[27] A. Skolik, J. R. Mcclean, M. Mohseni, P. Van Der Smagt, and M. Leib, "Layerwise learning for quantum neural networks."

[28] E. Grant, L. Wossnig, M. Ostaszewski, and M. Benedetti, "AN INITIALIZATION STRATEGY FOR ADDRESSING BARREN PLATEAUS IN PARAMETRIZED QUANTUM CIRCUITS TECHNICAL NOTE," 2019.

[29] J. Stokes, J. Izaac, N. Killoran, and G. Carleo, "Quantum Natural Gradient."

[30] N. Yamamoto, "On the natural gradient for variational quantum eigensolver."

[31] V. Bergholm *et al.*, "PennyLane: Automatic differentiation of hybrid quantum-classical computations." Accessed: Jul. 21, 2020. [Online]. Available: https://pennylane.ai.

[32] Y. Suzuki *et al.*, "Qulacs: a fast and versatile quantum circuit simulator for research purpose," 2020. Accessed: Jan. 04, 2021. [Online]. Available: http://arxiv.org/abs/2011.13524.

[33] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J., 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, *12*, pp.2825-2830.

[34] C. Ciliberto *et al.*, "Quantum machine learning: A classical perspective," *Proc. R. Soc. A Math. Phys. Eng. Sci.*, vol. 474, no. 2209, 2018, doi: 10.1098/rspa.2017.0551.

[35] Nielsen, M.A. and Chuang, I.L., 2001. "Quantum computation and quantum information". *Phys. Today, 54(2), p.60.*

[36] A. Barenco *et al.*, "Elementary gates for quantum computation," 1995. doi: 10.1103/PhysRevA.52.3457.

[37] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Dec. 2015, Accessed: Jan. 27, 2021. [Online]. Available: https://arxiv.org/abs/1412.6980v9.