Applied Quantum Algorithms Project

Raghav Juyal

May 2025

Part I

Theory Questions

1 Strongly Correlated Systems

Question 1

a

Fermionic creation and annihilation operators obey the following commutation relations:

$$\{a_i, a_j\} = 0$$
$$\{a_i^\dagger, a_j^\dagger\} = 0$$
$$\{a_i, a_j^\dagger\} = \delta_{ij} I$$

b

The transformation of fermionic creation and annihilation operators into Majorana operators is given below:

$$c_{i,0} = a_i + a_i^\dagger$$
$$c_{i,1} = \mathbf{i}(a_i - a_i^\dagger)\, where\, \mathbf{i}^2 + 1 = 0$$

c

Majorana operators obey the following commutation relations:

$$\{c_{i,\alpha}, c_{j,\beta}\} = 2\delta_{ij}\delta_{\alpha\beta} I$$

Question 2

The Jordan-Wigner transformation maps the creation and annihilation operators to the following qubit operators:

$$a_k = (\otimes_{i=1}^{k-1} Z_i) \otimes (\frac{X_k + \mathbf{i}Y_k}{2}) \otimes (\otimes_{i=k+1}^{n} I_i)$$
$$a_k^\dagger = (\otimes_{i=1}^{k-1} Z_i) \otimes (\frac{X_k - \mathbf{i}Y_k}{2}) \otimes (\otimes_{i=k+1}^{n} I_i)$$

From this we can see that since we have to apply gates to the first k qubits for a_k or $a_k^\dagger$, the locality for them is k. Thus, average locality for n qubits is given as,

$$\frac{1}{n}\sum_{i=1}^{n} I = \frac{n(n+1)}{2n} = \frac{n+1}{2}$$

Question 3

The Hamiltonian for the Fermionic Hubbard model on a 1-dimensional chain is given as

$$H = -t\sum_{i,\sigma}(a^\dagger_{i,\sigma}a_{i+1,\sigma} + a^\dagger_{i+1,\sigma}a_{i,\sigma}) + U\sum_i n_{i,u}n_{i,d}$$

Here, t is the hopping amplitude, $a_{i,\sigma}$ and $a^\dagger_{i,\sigma}$ are fermionic annihilation and creation operators at site i and spin $\sigma \in \{u(\uparrow), d(\downarrow)\}$, $n_{i,\sigma} = c^\dagger_{i,\sigma}c_{i,\sigma}$ is the number operator and U is interaction energy between electrons.

Question 4

Using the results of the Jordan-Wigner transformation in Question 2, we see the following results

$$\begin{aligned}
a^\dagger_{k,\sigma}a_{k+1,\sigma} &= (\otimes_{i=1}^{k-1}Z_{i,\sigma}) \otimes (\frac{X_{k,\sigma} - \mathbf{i}Y_{k,\sigma}}{2}) \otimes (\otimes_{i=k+1}^{n}I_{i,\sigma}) \times (\otimes_{i=1}^{k}Z_{i,\sigma}) \otimes (\frac{X_{k+1,\sigma} + \mathbf{i}Y_{k+1,\sigma}}{2}) \otimes (\otimes_{i=k+2}^{n}I_{i,\sigma}) \\
&= (\otimes_{i=1}^{k-1}I_{i,\sigma}) \otimes (\frac{X_{k,\sigma} - \mathbf{i}Y_{k,\sigma}}{2})Z_{k,\sigma} \otimes (\frac{X_{k+1,\sigma} + \mathbf{i}Y_{k+1,\sigma}}{2}) \otimes (\otimes_{i=1}^{k+2}I_{i,\sigma}) \\
&= (\otimes_{i=1}^{k-1}I_{i,\sigma}) \otimes (\frac{X_{k,\sigma} - \mathbf{i}Y_{k,\sigma}}{2}) \otimes (\frac{X_{k+1,\sigma} + \mathbf{i}Y_{k+1,\sigma}}{2}) \otimes (\otimes_{i=1}^{k+2}I_{i,\sigma}) \\
&= (\frac{X_{k,\sigma} - \mathbf{i}Y_{k,\sigma}}{2}) \otimes (\frac{X_{k+1,\sigma} + \mathbf{i}Y_{k+1,\sigma}}{2}) \quad \text{(ignoring the identity matrices)} \qquad (1)
\end{aligned}$$

Similarly we get

$$a^\dagger_{k+1,\sigma}a_{k,\sigma} = (\frac{X_{k,\sigma} + \mathbf{i}Y_{k,\sigma}}{2}) \otimes (\frac{X_{k+1,\sigma} - \mathbf{i}Y_{k+1,\sigma}}{2}) \qquad (2)$$

and

$$a^\dagger_{k,\sigma}a_{k,\sigma} = (\frac{I - Z_{k,\sigma}}{2}) \qquad (3)$$

Using the above results to apply the Jordan-Wigner transformation to the Fermionic Hubbard Model given in Question 3 gives us:

$$\begin{aligned}
H &= -t\sum_{i,\sigma}(a^\dagger_{i,\sigma}a_{i+1,\sigma} + a^\dagger_{i+1,\sigma}a_{i,\sigma}) + U\sum_i n_{i,u}n_{i,d} \\
&= -t\sum_{i,\sigma}((\frac{X_{i,\sigma} - \mathbf{i}Y_{i,\sigma}}{2}) \otimes (\frac{X_{i+1,\sigma} + \mathbf{i}Y_{i+1,\sigma}}{2}) + (\frac{X_{i,\sigma} + \mathbf{i}Y_{i,\sigma}}{2}) \otimes (\frac{X_{i+1,\sigma} - \mathbf{i}Y_{i+1,\sigma}}{2})) + U\sum_i(\frac{I - Z_{i,u}}{2})(\frac{I - Z_{i,d}}{2}) \\
&= \frac{-t}{2}\sum_{i,\sigma}(X_{i,\sigma}X_{i+1,\sigma} + Y_{i,\sigma}Y_{i+1,\sigma}) + \frac{U}{4}\sum_i(\frac{I - Z_{i,u}}{2})(\frac{I - Z_{i,d}}{2})
\end{aligned}$$

The two Hamiltonians have the same eigenspectrum since the Jordan-Wigner transformation it is a unitary transformation which preserves eigenvalues. It also preserves the anticommutation relations of the operators.

Question 5

a

For the Hubbard Model, using the result of Question 4, we can see that we only have interactions between neighbouring terms, which gives us $\mathcal{O}(n)$ where n is the number of qubits.

b

For the Heisenberg Model, the Hamiltonian is given as

$$H = -\frac{1}{2}\sum_{i=1}^{n}(J_x X_i X_{i+1} + J_y Y_i Y_{i+1} + J_z Z_i Z_{i+1})$$

From the Hamiltonian, we can see that we only have interactions between neighbouring terms, which gives us $\mathcal{O}(n)$ where n is the number of qubits.

c

For the Electronic Structure Problem, the Hamiltonian is given as

$$H = \sum_{i,j} h_{ij} a_i^\dagger a_j + \frac{1}{2}\sum_{i,j,k,l} g_{ijkl} a_i^\dagger a_j^\dagger a_k a_l$$

where i, j, k, l vary from 1 to n. From the Hamiltonian, we can see that due to the two-body terms(varying i, j, k, l) we get $\mathcal{O}(n^4)$ where n is the number of qubits.

2 Quantum Machine Learning

Question 1

We can implement Quantum Support Vector Machines(QSVMs) via two main methods: Implicit and Explicit Models.

Implicit Models

Implicit models, like the Quantum Kernel Estimator, implement a fixed quantum feature map and use the quantum computer to compute the inner products. The quantum circuit has no trainable parameters and is used to evaluate kernels, which are then used in the classical support vector machine(SVM) model.

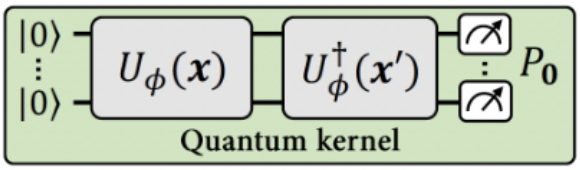


Figure 1: Implicit Model

Explicit Models

Explicit models, like Quantum Variational Classifiers, that first encodes classical data via a quantum feature map(which is usually fixed), and then has variational layers, i.e., layers with parameterized gates whose parameters are trainable. Here the measurement outcomes can be used as the classification value or sent for further classical processing.

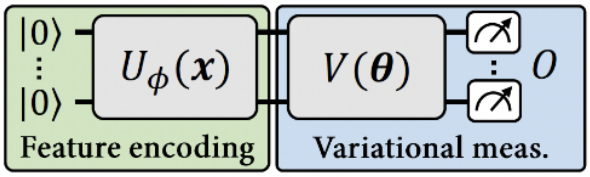


Figure 2: Explicit Model

Question 2

a

Feature maps are functions that map input data from its original space to another space(normally of higher dimension) called the feature space. The aim is that data that is not linearly separable in the original space may be linearly separable in the feature space.

Kernels are functions that compute the inner product in the feature space without explicitly using the feature map to transform the data. This way of avoiding the explicit transformation is known as the kernel trick.

b

The kernel is an inner product in the feature space, so the relation is given as

$$K(x, x') = \langle \phi(x), \phi(x') \rangle$$

c

According to Mercer's theorem, a kernel has a feature map if and only if it is a positive semidefinite kernel, aka a Mercer kernel.

Question 3

No, there do not exist any learning problems that quantum learners can learn but classical learners cannot, simply because we can then simulate the quantum learner on a classical computer. That being said, there do exist some learning problems that have an exponential speed up when using quantum learners compared to classical learners(assuming the widely believed conjecture that Shor's algorithm cannot be done classically in polynomial time). One example is the discrete logarithm problem

Question 4

To measure the probability of the first qubit in state $|0\rangle$, given that the d depth circuit doesn't depend on n(number of qubits) if we have local gates of constant width(i.e. 2 qubit gates, etc), we can draw a lightcone[1], i.e., lines connecting correlated qubits, starting at the qubit we want to measure and see that this grows exponentially in depth(for 2 qubit gates it is of $\mathcal{O}(2^d)$(similar concept to receptive field in convolutional neural networks). This does not depend on n and thus we can simulate efficiently in classical computer since we only need to focus on the qubits within the lightcone. That being said, we assume that the locality of the gates is independent of n. If it does depend on n, then the lightcone spreads very wide, and the number of qubits to be considered will depend on n and thus not be efficient on a classical computer.

Part II

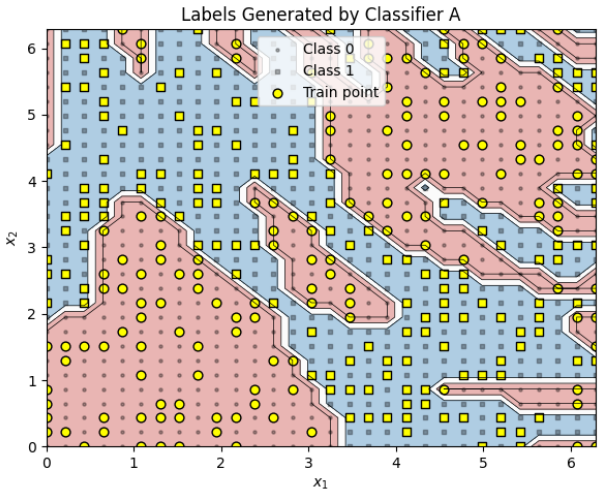
2.D. Expressivity of the generating set

1 Introduction

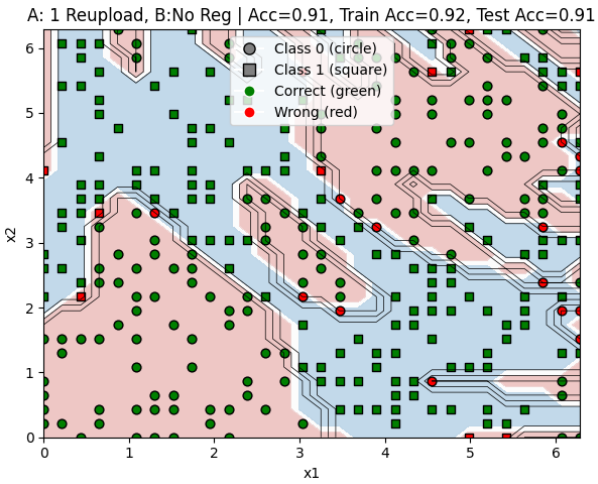
With the growing fields of machine learning and quantum computing, there is a lot of interest in whether quantum computing could answer some of the problems we face in classical machine learning, which led to what we now call quantum machine learning [2], [3]. One particular way to solve supervised learning tasks is to use variational quantum classifiers, where we use parameterized quantum circuits(PQCs) to map input to a prediction. PQCs often contain an encoding layer to encode the input into the quantum circuit, variational layers where we have parameterized gates whose parameters are trainable and a cost operator which tells us which basis to measure our quantum circuit in. In this work, I explore how different hyperparameters affect the expressivity, i.e., what functions or decision boundaries can the model learn, and the generalizability, i.e., how is the performance of the model on unseen data. Related works include [4], [5], where they go more into the theoretical bounds.

2 Methods

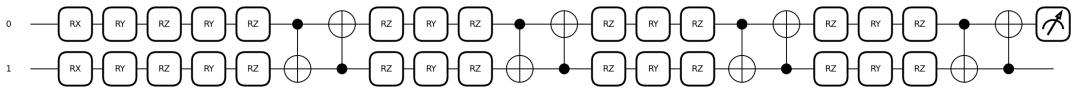
To look into the expressivity of different PQCs, I used two models A and B. I generated data with model A and used model B to learn it, and vice versa. In this way, we can see how different parameters can affect the expressivity of the PQCs.



(a) Generated Data



(b) Trained Decision Boundary



(c) Circuit to Generate Data

Figure 3: Visualisation of the data generation and learning process. 3a shows the dataset generated using a parameterised quantum circuit. 3b shows the decision boundary learned by the same circuit model but without regularisation in training. 3c shows the circuit architecture used to generate the training data.

2.1 Data Generation

To generate data, I take model A, randomly choose its parameters and generate labels for 900 points in a grid where $0 \leq x, y \leq 2\pi$. To label, if the measurement outcome is greater than 0, it is in class 1, else in class 0. Example of generated data can be seen in Fig. 3a and was generated by the circuit given in Fig. 3c.

2.2 Encoding

In most cases, the encoding of the models has remained fixed and is given in Fig. 4. Unless specified, this is the encoding used.

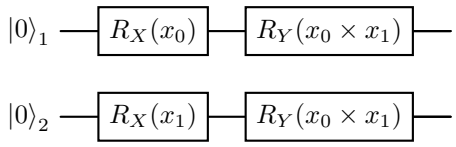


Figure 4: Encoding used in most models

2.3 Variational Layers

In most cases, the variational ansatz remains the same and is given in Fig. 5. It only gets changed when number of qubits is changed when varying width. In that case, for single qubit there is no entanglement and for more than 2 qubits, the entangling gates are applied in a circular fashion.

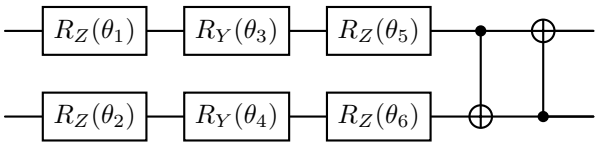


Figure 5: Single variational layer used in most models

2.4 Cost Operator

When measuring to get the output, we have to decide on what basis to measure. In most of the models, I have measured the first qubit in the Z basis.

2.5 Training

After generating 900 points of labelled data, I then split it into train and test sets. In most cases, 300 data points were used for training. While it is on the lower side, the results did not change much when I increased it to 600 data points. I then trained model B on the training data with Binary Cross Entropy loss. In most cases, the epochs were 100, and the learning rate was 0.1. Even when changing the learning rate to 0.01 or 0.001 or epochs to 50 or 200 or 500, the results didn't change much. An example of a decision boundary learnt by the model is given in Fig. 3b.

2.6 Outputs

When generating the data, the architecture, circuit diagram for PQCs like in Fig. 3c and layer information for classical neural networks, is shown alongside the decision boundary image similar to Fig. 3a. After the model has been trained, I display the learned decision boundary (similar to Fig. 3b),

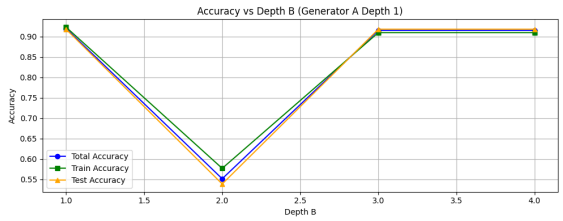
the train, test and total accuracy, precision, recall, F1 score and time taken to train the model. For comparing results, I have plotted the accuracies vs the hyperparameter being changed.

3 Results

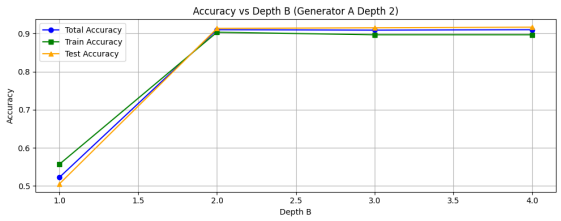
In this section, I discuss the outputs for the various hyperparameters varied. Most architectures are the same as mention in Section 2 but to see more details you can refer to the code attached to this report or at `https://github.com/RaghavJuyal/Expressivity-of-Generating-Sets`

3.1 Depth

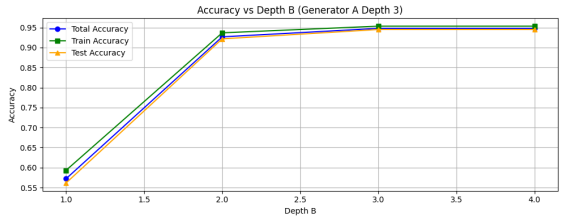
Here, I kept everything the same but only varied the number of variational layers. As can be seen in Fig. 6, in general, as we increase the depth, the accuracies(train and test) increase for all generators considered, with the exception of when depth 2 is used to learn data generated by depth 1(Fig. 6a). This suggests that as we increase the number of variational layers, the expressivity and generalizability of the PQC increase, which follows from the deeper circuits having more parameters.



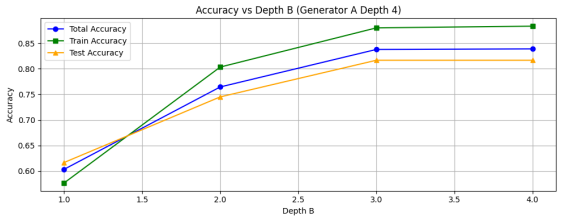
(a) Depth 1



(b) Depth 2



(c) Depth 3



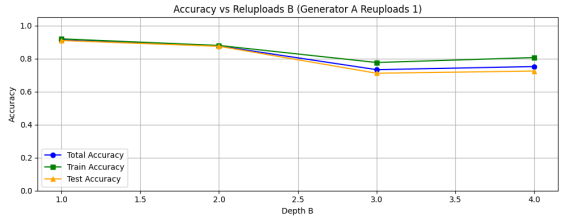
(d) Depth 4

Figure 6: Comparison of outputs for different circuit depths.

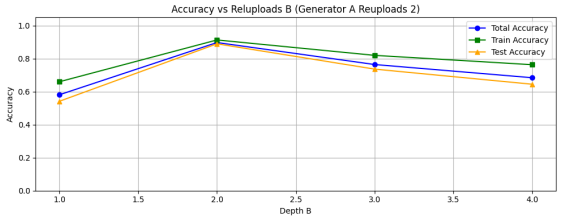
3.2 Reuploading

3.2.1 Depth

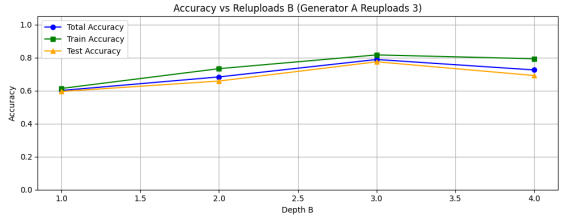
Here we see the effect of reuploading data. Each case has four variational layers. What is varied is how often I have done data encoding. Reupload depth 1 means I have only encoded data in the start before the variational layers. Reupload depth 2 means I have encoded data both just before the first variational layer and just before the second variational layer. Similarly, we can define Reupload depths 3 and 4. From Fig. 7, we see that whatever model generated the data performed the best in both the train and test data. Thus, we can't really conclude any effect due to a larger number of encoding layers. This experiment was repeated for more training data, more epochs and a different learning rate; however, similar results were obtained. This seems to contradict the results shown in [4] but it could be due to the specific hyperparameters being used, so it warrants more looking into in future work.



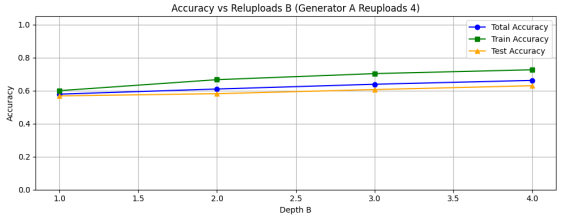
(a) Reupload Depth 1



(b) Reupload Depth 2



(c) Reupload Depth 3



(d) Reupload Depth 4

Figure 7: Comparison of outputs for different reupload circuit depths.

3.2.2 Width

Here, we see the effect of reuploading data by changing the width of the circuit. Each qubit has the same encoding as given in Fig. 8. From Fig. 9, we can see that as we increase the width of the PQC, the accuracies(train and test) increase. This suggests that as we increase the width of the PQC, the expressivity and generalizability of the PQC increase, which follows from the results shown in [4].

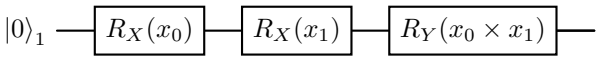
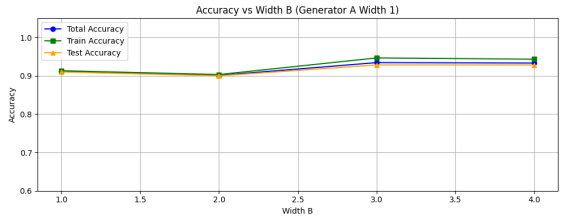
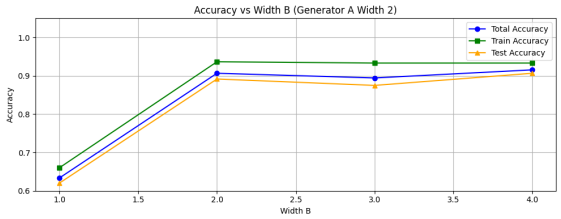


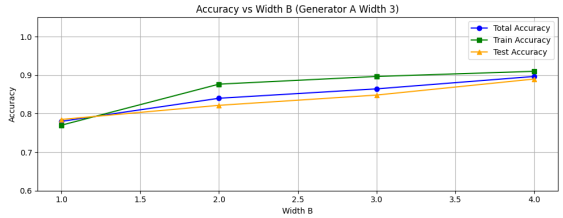
Figure 8: Encoding used in each qubit when varying width



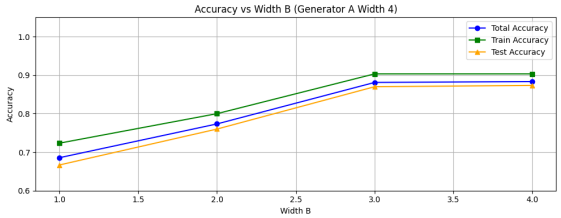
(a) Width 1



(b) Width 2



(c) Width 3

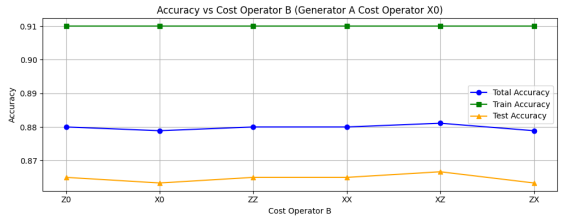


(d) Width 4

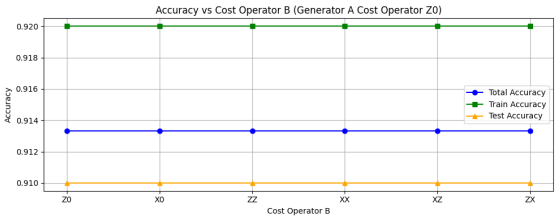
Figure 9: Comparison of outputs for different circuit widths.

3.3 Cost Operator

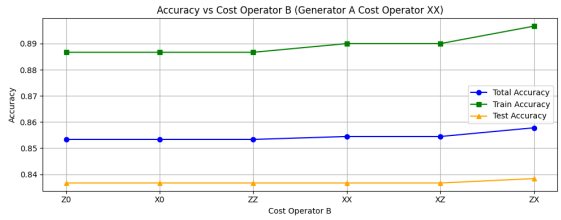
Here we keep everything the same in the 2-qubit 4 variational layer PQC but change the cost operator. From Fig. 10 we can see that in general, the cost operator being used doesn't really affect the performance of the PQC.



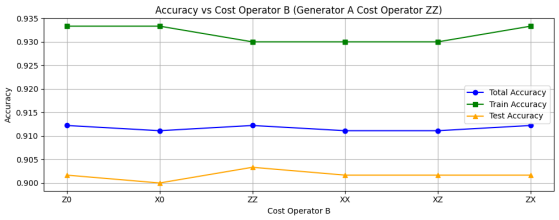
(a) Cost Operator X_0



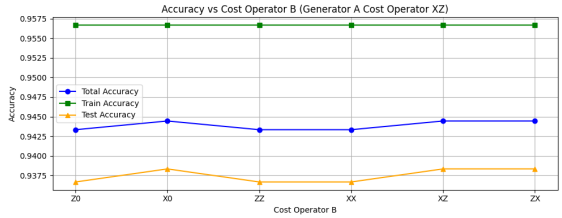
(b) Cost Operator Z_0



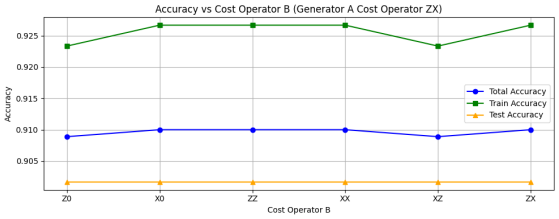
(c) Cost Operator XX



(d) Cost Operator ZZ



(e) Cost Operator XZ



(f) Cost Operator ZX

Figure 10: Comparison of outputs corresponding to different cost operators

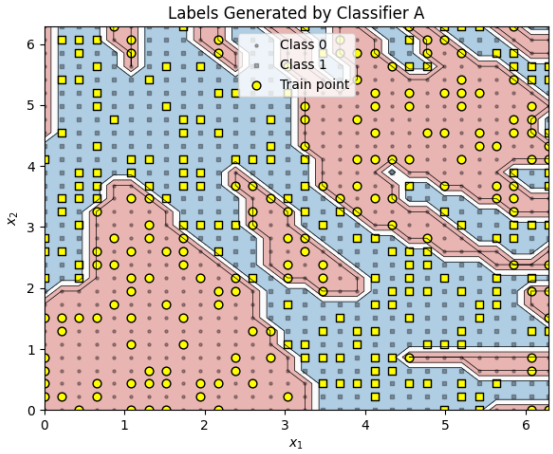
3.4 Different Encoding Schemes

Here we keep everything the same but vary the data encoding layer. The encodings used are original (the encoding mentioned in Section 2.2), Rx (only X rotations), Chebyshev Sparse and Tower encodings [6], and Fourier encoding[4]. We can see the data generated in Fig. 11.

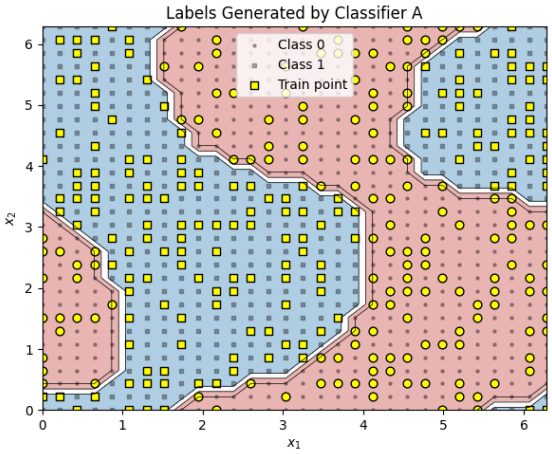
From Fig. 12, we see that the model that performs the best is the one with the same encoding scheme. This seems to imply that the different encoding schemes have different inductive biases [7] and is worth looking into further in future work.

3.5 Regularization

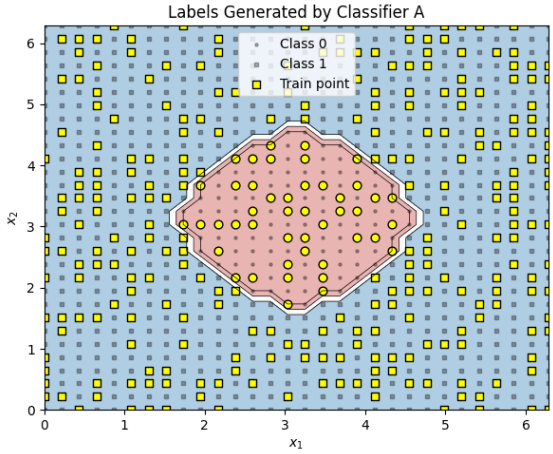
Here, we have a single generating model and are using the same model to learn. Instead, during the training, we are comparing different regularization. No Reg stands for No regularization, L1 for L1 regularization, L2 for L2 regularization and L1+L2 for L1 and L2 regularization both. From Fig. 13, we can see that L1 regularization has the highest train accuracy while No regularization has the highest test accuracy. This could imply that the model considered does not have a very high capacity compared to the task, which could warrant regularization to avoid overfitting.



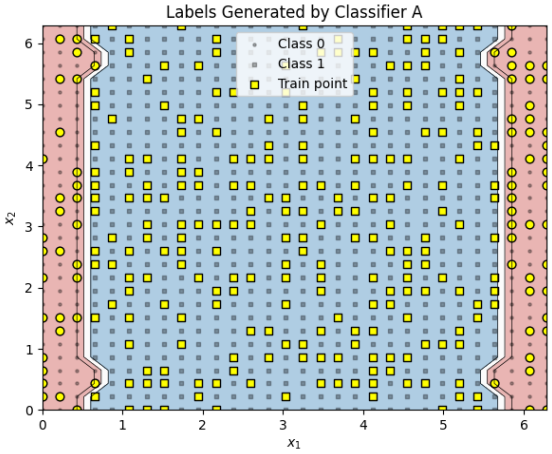
(a) Original Encoding



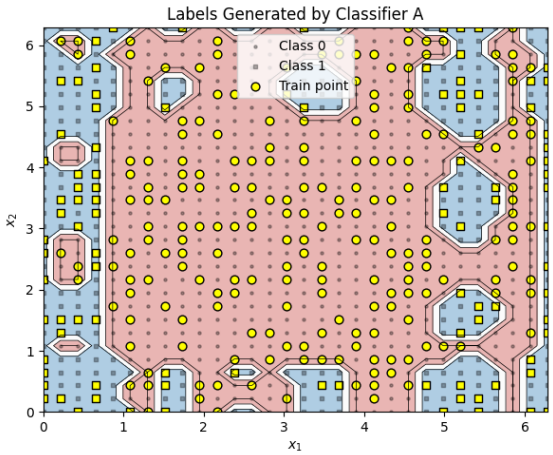
(b) RX Encoding



(c) Chebyshev Sparse Encoding

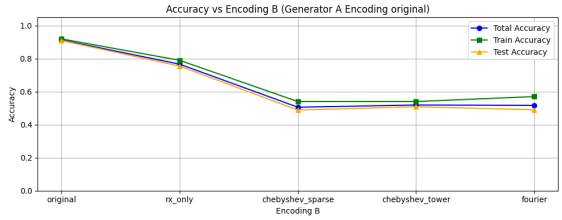


(d) Chebyshev Tower Encoding

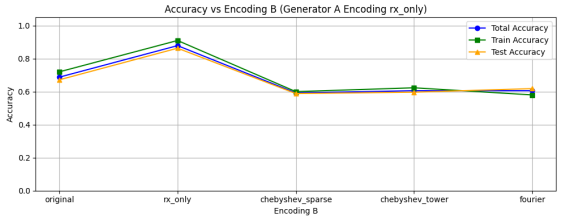


(e) Fourier Encoding

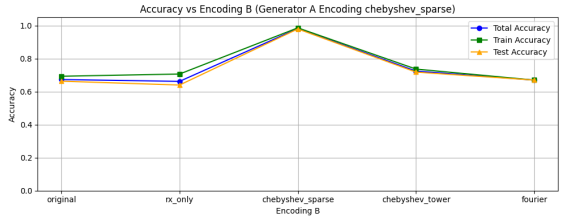
Figure 11: Decision boundaries for data generated by different encoding schemes



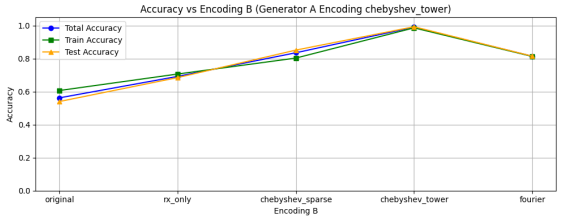
(a) Original Encoding



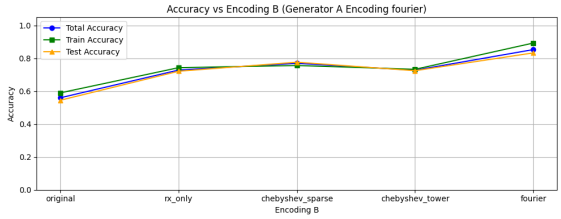
(b) Rx Encoding



(c) Chebyshev Sparse Encoding



(d) Chebyshev Tower Encoding



(e) Fourier Encoding (Full Width)

Figure 12: Comparison of different encoding schemes

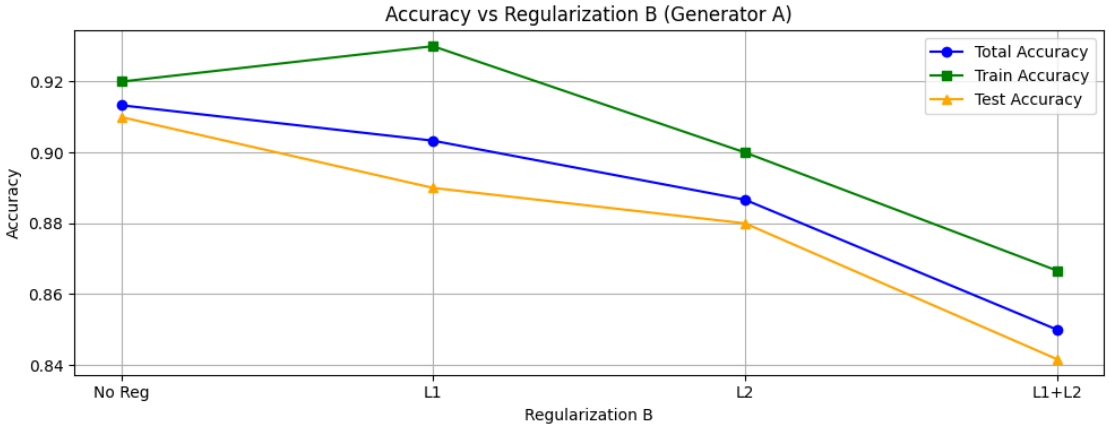


Figure 13: Comparison of outputs for different regularization schemes

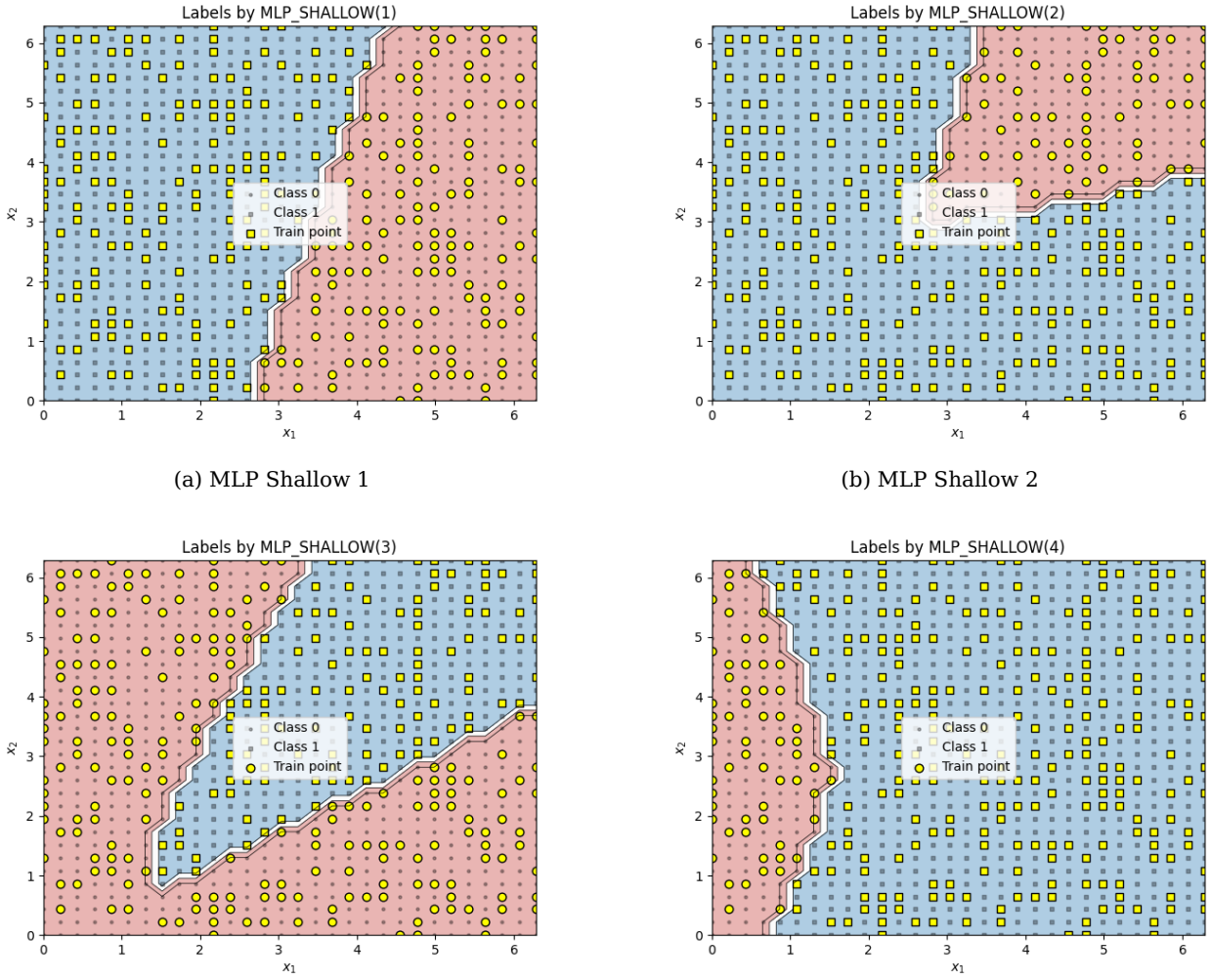
3.6 Quantum vs Classical

Here we compare quantum models to classical neural networks. For the quantum models, we use 4 models with varying widths(same as in Section 3.2.2). For each PQC width, we have 2 neural networks. First is MLP_shallow, which has a similar number of parameters as the PQC but only 1 hidden layer; the other is MLP_dee,p which has a similar number of parameters as the PQC but has the same width and is thus a deeper network. Here we will be looking at the data generated by MLP_shallow and the

PQCs, since the data generated by MLP_deep always gave all data points the same label.

3.6.1 MLP Shallow

In Fig. 14 we can see the data generated by the classical shallow neural networks.



(a) MLP Shallow 1

(b) MLP Shallow 2

(c) MLP Shallow 3

(d) MLP Shallow 4

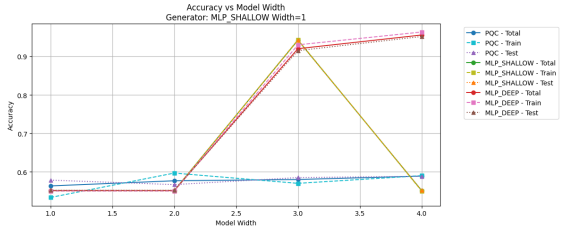
Figure 14: Decision boundaries for data generated by the shallow MLP models

As seen in Fig. 15, in general, the classical networks(shallow and deep) outperformed the PQCs in both train and test data.

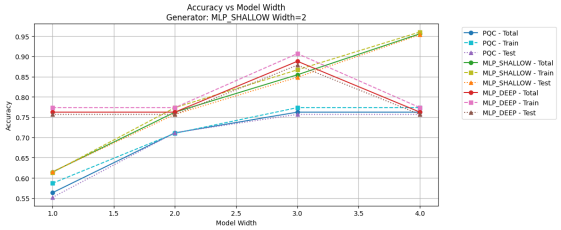
3.6.2 PQC

In Fig. 16 we can see the data generated by the PQCs.

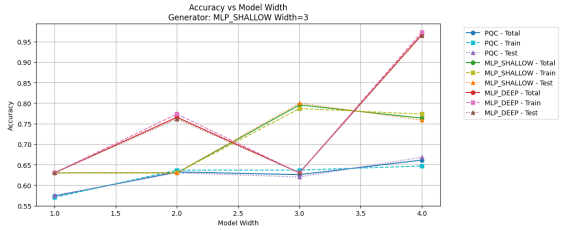
As seen in Fig. 17, in general, PQCs outperformed the classical networks(shallow and deep) in both train and test data. From this and from 3.6.1 we can conclude that PQCs and classical neural networks have different inductive biases [7], causing PQCs to outperform classical networks when data is generated by PQCs and vice versa. This was observed even when the learning rate was changed and even when a "simpler" encoding scheme was used(Rx only). This warrants more looking into to see whether this could be explained using the partial fourier sum [4] representation of quantum models



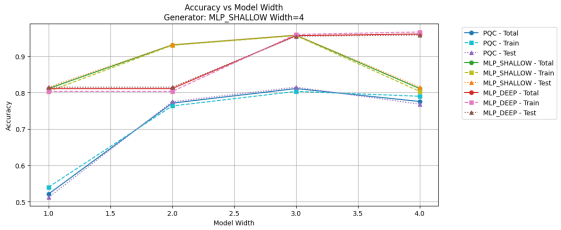
(a) Output 1



(b) Output 2



(c) Output 3



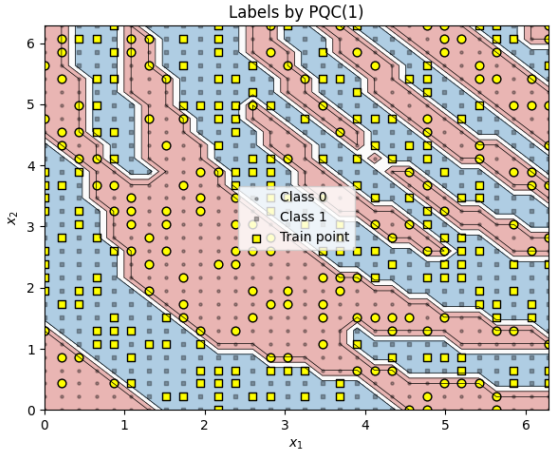
(d) Output 4

Figure 15: Comparison of quantum and classical models on data generated by MLP shallow networks

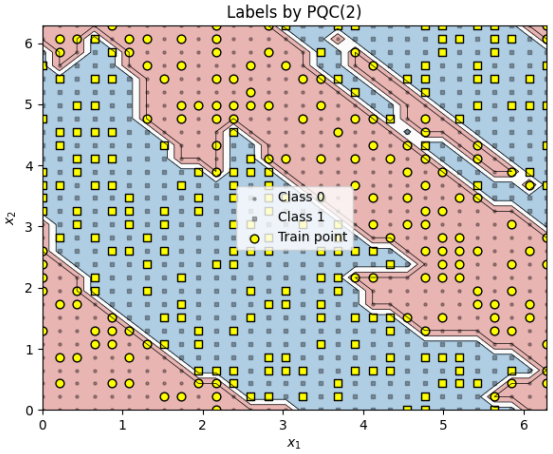
and an equivalent one for classical neural networks. Further research here could shed more light on the relation and differences between classical and quantum models.

4 Conclusion

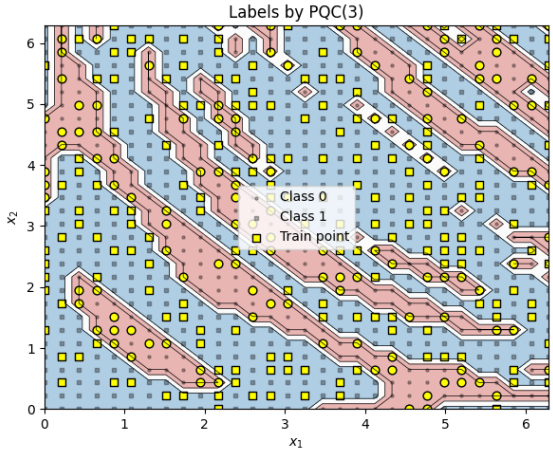
In this work, I compare different depths, reuploading schemes, cost operators, encoding schemes, regularization, and compare quantum models with classical models. When comparing encoding schemes and when comparing quantum models to classical models, we saw that they have an impact on performance which could be explained by them having different inductive biases, i.e., they favour certain solutions due to their structure. Further work involves comparing other hyperparameters like different variational layers, presence or type of entangling gates and looking more into the inductive biases of quantum models and classical models, as well as different encoding schemes.



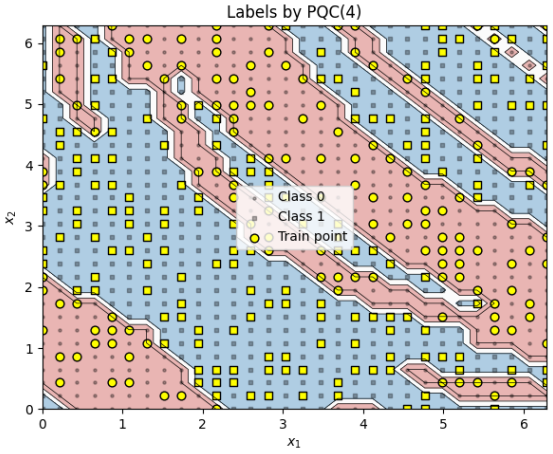
(a) PQC Width 1



(b) PQC Width 2

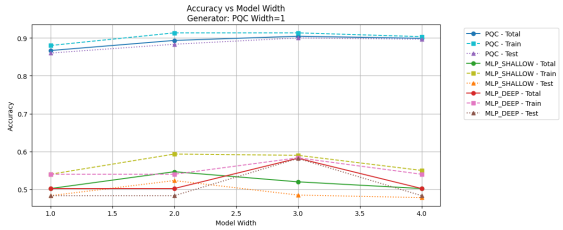


(c) PQC Width 3

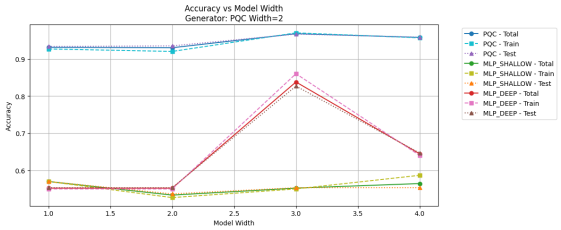


(d) PQC Width 4

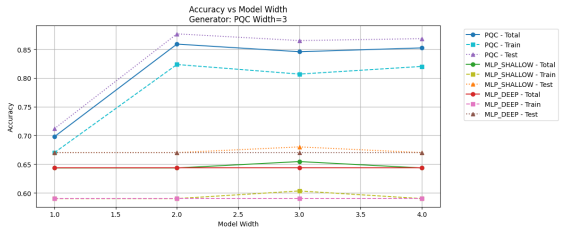
Figure 16: Decision boundaries for data generated by the PQC models



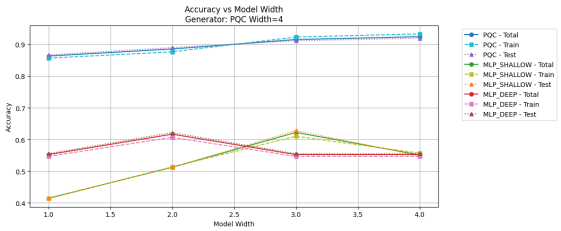
(a) Output 1



(b) Output 2



(c) Output 3



(d) Output 4

Figure 17: Comparison of quantum and classical models on data generated by PQCs

References

[1] H.-Y. Huang, Y. Liu, M. Broughton, *et al.*, "Learning shallow quantum circuits," in *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, ser. STOC '24, ACM, Jun. 2024, pp. 1343–1351. DOI: 10.1145/3618260.3649722. [Online]. Available: http://dx.doi.org/10.1145/3618260.3649722.

[2] V. Havlíček, A. D. Córcoles, K. Temme, *et al.*, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, Mar. 2019, ISSN: 1476-4687. DOI: 10.1038/s41586-019-0980-2. [Online]. Available: http://dx.doi.org/10.1038/s41586-019-0980-2.

[3] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, "Circuit-centric quantum classifiers," *Physical Review A*, vol. 101, no. 3, Mar. 2020, ISSN: 2469-9934. DOI: 10.1103/physreva.101.032308. [Online]. Available: http://dx.doi.org/10.1103/PhysRevA.101.032308.

[4] M. Schuld, R. Sweke, and J. J. Meyer, "Effect of data encoding on the expressive power of variational quantum-machine-learning models," *Phys. Rev. A*, vol. 103, p. 032 430, 3 Mar. 2021. DOI: 10.1103/PhysRevA.103.032430. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.103.032430.

[5] M. C. Caro, E. Gil-Fuster, J. J. Meyer, J. Eisert, and R. Sweke, "Encoding-dependent generalization bounds for parametrized quantum circuits," *Quantum*, vol. 5, p. 582, Nov. 2021, ISSN: 2521-327X. DOI: 10.22331/q-2021-11-17-582. [Online]. Available: http://dx.doi.org/10.22331/q-2021-11-17-582.

[6] O. Kyriienko, A. E. Paine, and V. E. Elfving, "Solving nonlinear differential equations with differentiable quantum circuits," *Physical Review A*, vol. 103, no. 5, May 2021, ISSN: 2469-9934. DOI: 10.1103/physreva.103.052416. [Online]. Available: http://dx.doi.org/10.1103/PhysRevA.103.052416.

[7] T. Mitchell, "The need for biases in learning generalizations," Oct. 2002.