FYS4411 Project 2: Quantum Machine Learning

Annika Eriksen, Linus Ekstrøm

April 2021

Abstract

In this project we investigate the power of variational quantum circuits (VQC's), sometimes also referred to as parametrized quantum circuits (PQC's), which is a hot topic within the emerging field of quantum machine learning (QML). We construct 5 different models of increasing complexity to test the effects of model complexity and repeated data encoding. We apply our models to the Iris dataset, and we find that high learning rates $\gamma \approx 1$ performs well, *in contrast to classical machine learning*. We also observe that the convergence rate of VQC's is very high. Parameterising the circuit with a double ansatz gives a 100% accuracy rate after 5 epochs. At this point the cross-entropy lies at about 0.375 and falls to 0.13 after 10 epochs. Github link to project code `https://github.com/ageriksen/Quantum-Machine-Learning`.

Contents

1 Introduction

The project aims to explore *quantum machine learning*[14]. In broad strokes it is one of the most exciting marriages of some of the most powerful concepts and ideas developed in the last century. As Maria Schuld defines it: '*machine learning with quantum computers or quantum-assisted machine learning*' [11] It is a field in constant development, where practically nothing is set in stone.

We examine the well known "Iris" dataset imported from Scikit-learn, and we will be using so-called *parameterized quantum circuits* to do our machine learning. The main focus of study in this project will be the efficiency of various *parameter encodings* and *model ansatzes* in terms of the model convergence and prediction accuracy on the well known iris-dataset.

The paper introduces first some theory of basic Quantum mechanics and Quantum computing and moves on to Quantum Circuits. There is some info about the Iris data set and finally theory regarding parameterized quantum circuits.

Further is a description of the method for implementing the theory discussed through python programming, moving through the general steps of importing and preprocessing the data as well as encoding the features and implementing a model ansatz for the circuit to train on.

The results of the experiments are presented over the 5 different models used and finally a congregation of the best simulations per model is presented.

The results are discussed per model and in more general terms discussing aspects and faults of the encoding and methods used.

Finally, a conclusion of the best models for the parameterized circuits and the more ideal values for learning rates and shots are presented. Lastly there is also speculations about further research and improvements that could be done for the present data.

2 Theory

The main theory here is in regards to basic quantum computing and quantum circuits, with a quick introduction to the encoding.

The data set used is also examined shortly.

Further on is overview over the parameterized quantum circuits and how the model is implemented on these. as well as the optimization scheme for the model parameters.

2.1 Introduction to Quantum Computing and Quantum Circuits

Quantum computing is the extension of the field of classical computing. Where classical computing uses bits that can take values of either 0 or 1, quantum computing has an analogue: the *qubit* which has a lot more possible states than the classical bit. It is common to think of qubits in terms of the classical computational basis 0 and 1. Representing these in the notation of quantum mechanics we have: $|0\rangle$ and $|1\rangle$. We mentioned that there are a lot more possible states for the qubit, they can actually assume values that 'lie in-between' the computational basis states. In addition it is possible to form *linear combinations* of states

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{1}$$

since we are dealing with quantum mechanical objects here, we have the usual requirement that probabilities add up to one, and that the classical probability is the square of the probability amplitude. $|\alpha|^2 + |\beta|^2 = 1$. Said in other words: a qubit is any quantum mechnical system with two-dimensional state space $\mathcal{H}$ and a distinguishable orthonormal basis $|0\rangle, |1\rangle \in \mathcal{H}$. When thinking of single qubits it is often beneficial to think in terms of the so-called Bloch sphere, shown in figure 1.

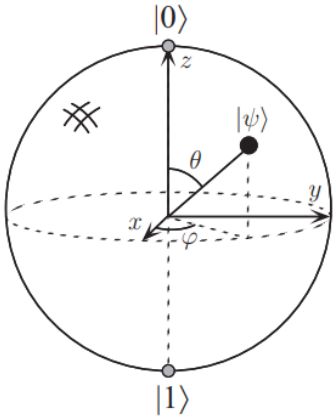


Figure 1: Bloch Sphere representation p.15 Nielsen [10]

where we have rewritten (1) into

$$|\psi\rangle = e^{i\gamma}\big(\cos\frac{\theta}{2}\,|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}\,|1\rangle\big)$$
$$\implies \cos\frac{\theta}{2}\,|0\rangle + \sin\frac{\theta}{2}\,|1\rangle$$

because global phase factors have no observable effects in quantum mechanics.

Thus we understand that we can represent data in terms of the computational basis states in much the same way as we do for classical computing. The main difference is that a classical bit is like a coin, it is either 0 or 1.[1] Whereas, a qubit can exist in a *continuum of states* between the computational basis states until it is observed. Packed in this statement lies the power of quantum computing, and the hope of the field of quantum computing

[1] In fact this statement is fairly redundant as the word *bit* is a contraction of *binary* and *digit*.

is to be able to use this immense computational power tucked away in this continuum of states, which grows exponentially with the number of qubits. Said in other words, to describe the state of n bits one would need n boolean variables $a \in [0, 1]$ where to fully describe the state of n qubits you need 2^n amplitudes $\alpha \in \mathbb{C}$. There are other concepts as well such as *quantum parallelism*, *interference* and *entanglement* which also potentially offer computational methods that are physically inaccessible on a classical computer. More information on these subjects can be found in ([10], [2])

2.2 Fisher's Iris-flower Multivariate dataset

Introduced in the 1936 paper *The use of multiple measurements in taxonomic problems* [8] by the outspoken eugenicist and statistician Ronald Fisher published in the *Annals of Eugenics*. The publications therein has been released as a matter of public record, and is in no way an endorsement of the views expressed within.

Considering references brought up in the first paragraph regarding skull shapes this is likely a wise decision. The data set was gathered by an American botanist Edgar Anderson who was working with Fisher at the time.

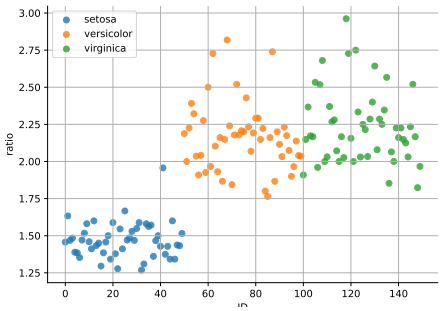


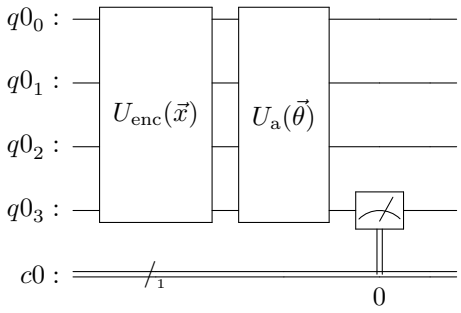
Figure 2: Scatter plot of the iris set with sepal width and length for the 3 species.

The data set consists of 50 samples of 3 species found in the same pasture in Quebec, Canada. 4 features were measured per sample: length and width of the sepals and petals, in centimeters. A scatter-plot of the sepal width and length can be seen in figure 2. These four were used by Fisher to develop a linear discriminant method to distinguish the species. The 3 species are *Iris Setosa, Iris Versicolor* and *Iris Virginica*.

Based on Fisher's model, the set has become a typical test case for many statistical classification techniques, such as Support Vector Machines and is cited extensively [7].

2.3 Parametrized Quantum Circuits

Our method of quantum machine learning will be using so-called parametrized circuits [3]. This is essentially a hybrid-quantum machine learning model. Hybrid because both the input data and the optimization algorithm are classical. A parametrized quantum circuit can be thought of as consisting of three main parts: *the encoder* $U_{\text{enc}}(\vec{x})$ which does as the name suggests and encodes our sample into a quantum state, *the ansatz* $U_{\text{a}}(\vec{\theta})$ which holds our tune-able model parameters $\vec{\theta}$ and can be thought of as a quantum parallel to a classical machine learning model, and finally the *measurement / model optimization* part which lets us read out predictions from our circuit and optimize our model parameters in a method somewhat similar to classical back-propagation.



We can see all of these parts described in the circuit presented above. Do note that four input qubits was here used for simplicity of drawing the circuit, but in principle the diagram should have been drawn with n input qubits. In principle measuring more than just the 'last' qubit is possible and it is still an active research topic as to which method produces better results. One paper using measurement of all the qubits is [1]. After the measurement part of our model we move over into the classical computing domain where we are able to update our model parameters $\vec{\theta}$ in order to move our model output closer to the desired output, reminiscent to the classical way of training a neural network.

Next we go into more details on each of the parts of our parameterized quantum circuit.

2.3.1 Feature Encoding

One of the things we will be testing different implementations of in this project is different ways of feature encoding. Since this whole field of quantum machine learning is still in its infancy, there is a 'test-and-see' to the specifics of our feature encoding, rather than encoding schemes based on purely mathematics. This is also arguably one of the reasons this subject is so exciting!

For this work we will be min-max-scaling our features before sending the data into our circuit for training. To do this we use the well known MinMaxScaler from the

scikit-learn preprocessing library to scale our inputs (features) between 0 and 1. This is customary practice in classical machine learning and is often helpful in generating better predictors.

Moving onto how we encode our features into quantum states using our data encoder circuit U_{enc} as shown in above. The way we have setup our circuit is with four qubits, one for each feature in the iris dataset. What our encoder circuit does is for each such feature $G(x) = e^{-ix\frac{\theta}{2}\sigma_X}$. To use terms that mirror Maria Schuld et al in [12], we say that we encode our data using, *in our 4-qubit implementation*, four encoding gates $G(x)$ in *parallel*.

$$e^{-i\frac{\pi x_1}{2}\sigma_X} \otimes e^{-i\frac{\pi x_2}{2}\sigma_X} \otimes e^{-i\frac{\pi x_3}{2}\sigma_X} \otimes e^{-i\frac{\pi x_4}{2}\sigma_X}$$

where σ_X is the familiar Pauli-X-gate. We ended up adding a '*serial entanglement*' to our encoding in order to make our '*double encoding*' model make sense so that our full encoding block looks like the above multiplied with

$$C_{X,\text{chained}} = \big(C_X \otimes I \otimes I\big)\big(I \otimes C_X \otimes I\big)\big(I \otimes I \otimes C_X\big)$$

Thus our whole encoder can be written as[2]

$$U_{\text{enc}} = C_{X,\text{chained}}\Big(e^{-\frac{\pi x_i}{2}\sigma_X}\Big)^{\otimes 4}$$

This way of encoding can be interpreted as "time evolution" by a Hamiltonian described by σ_X. Take note that since we have scaled our features between 0 and 1, we are essentially remapping like $\{0, 1\} \mapsto \{0, \pi\}$. Initially we had a factor 2π here, however that resulted in mapping 0 and 1 to the same state, which is undesirable. Factors different to π may be used in this model depending on the scaling of the input data among other things. In the aforementioned paper [12] it is shown that: "*repeated feature encoding linearly extend the frequency spectrum*" of the quantum model. The topics discussed in that paper are a bit more advanced than the scope of this project, but in essence their results, tells us that with increased number of encoding 'passes' the higher the number of functions the model '*has access to*'. It is worth mentioning that in the same paper they mention that their work has a large overlap with another independent work [9] by Francisco Javier Gil Vidal and Dirk Oliver Theis. Their work propose alternate 'functions' for encoding the data into rotations. For example our way of doing it would be referred to as *linear input encoding*. Vidal and Theis speak to this, but also consider arcsin *input encoding as well*. However, both papers are clear in the fact that multi-pass feature encoding does increase the number of functions a variational quantum circuit 'has access to'. One of the main foci of this project will be to test the effect of multi-pass feature encoding.

2.3.2 Ansatz

The other building block of our model is what we referred to as the *ansatz* $U_a(\vec{\theta})$. Borrowing a bit again from Schuld et. al [12]. They propose a *partial fourier series* representation of the variational quantum circuit. From this representation we are able draw the conclusion: while repeated encodings increase the number of functions the model has access to, the ansatz defines a property *expressivity* which determines how the model is able to combine the functions which it has access to. However, that paper concerns itself primarily with the effects of repeated encoding, there have been works looking specifically at this property expressivity ([13], [6], [4]). The paper by Sim, Johnson and Aspuru-Guzik categorizes 19 different model ansatzes, however their discussion is a bit beyond the scope of this project. [3]. As for the Chen et. al [4] paper they mention the concept of introducing non-linearity directly into the model ansatz in the form of measurements, which as we know are non-linear operators in quantum mechanics.

2.3.3 Full Model

In this work we investigate the efficacy of four different variational quantum models. We we wanted to look at the convergence rate using single encoding and a simple ansatz, double encoding blocks with a simple ansatz, single encoding with a 'double' ansatz (repeated twice) and a model with both double encoding and a double ansatz. We were interested to find out whether the model with the simple ansatz and double encoding for example was able to compete with the model using a more complicated ansatz implementation.

2.3.4 Parameter Optimization and the Gradient Shift Rule

In order to train our model we have to have a way to update our model parameters $\vec{\theta}$. For this project we will stick to the steepest descent method, however with future work using something like stochastic gradient descent might provide better results if the models get larger. Another option might be to use something like the ADAM optimizer for even faster convergence. Nevertheless, to use gradient descent we obviously need the gradient of with respect to each parameter θ_j for each sample. We have opted to use the binary cross-entropy as our loss function

$$L = -\frac{1}{N}\sum_{i=1} y_i \ln\Big(f(x_i;\vec{\theta})\Big) + (1 - y_i)\ln\Big(1 - f(x_i;\vec{\theta})\Big)$$

where $f(x_i;\vec{\theta})i$ is our model prediction, y_i is the target, and N is the number of samples. We now take the derivative with respect to our model output for each sample i

[2] Last minute change in how we did our encoding left little time to check the math behind this. Should be right.

[3] As a note for further work in this subject it would be very interesting to use the proposed expressivity and entanglement capability measures described in [13]

and write $f(x_i, \vec{\theta}) = f_i$ for clarity. Then we have

$$\frac{\partial L}{\partial f_i} = -\frac{1}{N}\sum_{i=1}(\frac{y_i}{f_i} - (1-y_i)\frac{1}{1-f_i})\frac{\partial f_i}{\partial \theta_j}$$

where we have used the chain rule. Now all we are missing is this term $\partial f_i/\partial\theta_j$. We can find these derivatives with respect to our model parameters using the so called parameter shift rule which is outlined in this paper [5] by Gavin Crooks.

$$\frac{\partial f_i}{\partial \theta_j} = \frac{f(\theta_1, \cdots, \theta_j + \frac{\pi}{2}, \cdots \theta_k) - f(\theta_1, \cdots, \theta_j - \frac{\pi}{2}, \cdots \theta_k)}{2}$$

From this paper we can conclude that the parameter shift rule yields the correct gradient w.r.t. θ_j so long as the gates in our model can be decomposed into a product of simpler gates each of which are parameter-shift differentiable. We can also surmise that, *and this is the result we will use*, that the parameter-shift rule can be directly applied to gates with two unique eigenvalues. Thus, we are able to directly use the parameter-shift rule and we are therefore able to calculate the gradient of the binary cross-entropy w.r.t. each model parameter θ_j.

Once we have these we just use a standard gradient descent where we modify our previous $\vec{\theta}$ by subtracting by the above expression times a learning rate γ.

3 Method

To implement our model we created a class, QML. The class sets up the parametrized quantum circuit with encoder and ansatz parts as explained in the previous section. We outline the code structure of our QML class below. Note that this is pseudocode

```
class QML:
def __init__(self,
        n_quantum, n_classic,
        featureMatrix, n_model_parameters,
        seed, shots=1000,
        backend='qasm_simulator',
        model="basicModel", lossfunc="BCE",
        delLoss="BCEderivative"):

    def basicModel(self):

    def doubleAnsatz(self):

    def lessEntangled(self):

    def doubleEncoding(self):

    def doubleAnsatzdoubleEncoding(self):

    def modelCircuit(self, printC=False):

    def BCE(self, out, target):

    def BCEderivative(self, out, target):

    def train(self, target,
        epochs=100,
        learning_rate=.1,
        debug=False):
```

The main points of the code are the import and preprocessing of data from the iris set to send into the class. QML takes in the feature matrix and target as well as the model parameters and the specific model we want to test.

The call structure begins with initializing the object. The initialization defines most of the class instance's facets. The qubits and classical bits as well as the seed are set. The feature matrix is also set here. This is also where the model parameters θ are initialized as a vector of uniformly distributed angles between 0 and 2π. The next call of the class is the train function which sets up the model and trains it over the set epochs. Train returns the θ values of the model as well as the cross-entropy loss function and the accuracy of each epoch.

3.1 Importing and Preprocessing the Data

The data used is the iris flower set, which is imported as a data frame via the sklearn module. We cut away 1 of the 3 classes so we have a binary classification and scale the features to 1 using the sklearn MinMaxScaler. Within the model, the features are mapped like angles along a half circle span. The half circle is to maintain a maximal distance between features scaled towards 0 and 1. These would be mapped to the same angle otherwise.

3.2 Encoder

The basic encoding of the data consist in taking a scaled feature matrix and selecting a sample to view. the features of the sample are then encoded onto a bloch-sphere by applying a rotation around the x axis with the "RX" gate. For this, the scaled values are converted to radians through scaling by π. After rotating the features, the qubits are coupled together with controlled x gates, "CX"

3.3 Ansatz

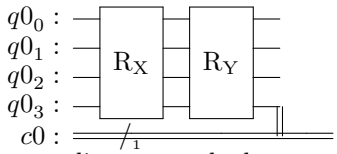
The basic ansatz applies a set of rotations around the y axis through the "RY" gate of the parameters $\vec{\theta}$ making up the model for the system, maintaining the last one as a bias. After the weights are applied to the feature qubits, they are entangled into superpositions using "CX" gates. This controls the relationship between the features.

Finally the bias is applied as a rotation around the y-axis at the end just before before measurement. This last step is equivalent across the different models.

3.4 Building the model

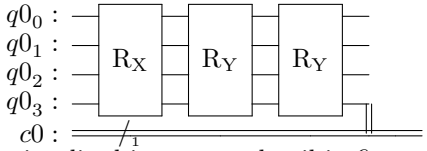
The main area of study for the project is in the variations on the model setup for prediction accuracy. This then means the types, and number, of rotations for the features as well as the weights θ.

The most basic circuit build is a single encoding of the data as well as a single ansatz rotation applied, before the final bias rotation and measurement. A visualization of the circuit is



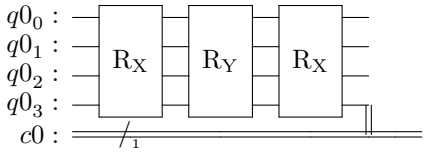
where the encoding around the x axis is given the moniker R_X and the ansatz rotating around the y axis has the moniker R_Y. The circuit model is illustrated in greater detail in figure 3.

The next step in the circuit model is to encode further weights θ. This doubles the weights used. The intent is to increase the number of tune-able parameters to check for improvements in the model adaptability. the basic idea is similar, with



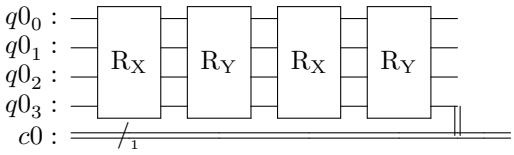
This is visualised in greater detail in figure 5.

A third model is the re encoding of the features after the weights are applied. This effectively increases the polynomial degree of the model and could lead to greater responsiveness in the predictions. The basic layout here is then



Which can be seen in more detail in figure 7

Finally a fourth model we test is both encoding the features twice and applying weights twice. This allows for both a higher polynomial degree to the model and more tune-able parameters. The base layout then becomes



With a greater detail in figure 11

The circuit model is run for a given number of "shots" which gives us an expectation value for prediction which can then be used to compare with the target and calibrate the model with cross entropy and gradient descent.

3.5 Training the Model and Segmenting Feature Matrix

To apply the model on the iris set, each sample has to be sent in separately. The same weights are used for all samples per epoch of the training run while the gradient of the ansatz angles and the loss function are calculated and stored. Having run through the samples, the average gradient of the loss function as well as for the separate θ values are applied to the current model values to correct for the next epoch scaled by a learning rate. The learning rate is among the variables we test for in the various runs.

3.6 Returning and Storing Data

After training the model, the model values, θ, as well as the mean loss and the accuracy for each epoch are returned out of the class and stored in a data file with some metadata on the 1st line and the θ, loss and accuracy on following lines.

4 Results

4.1 Basic Model

First up is the '*basic*' model of one implementation each of encoder and ansatz. the model is illustrated in figure 3

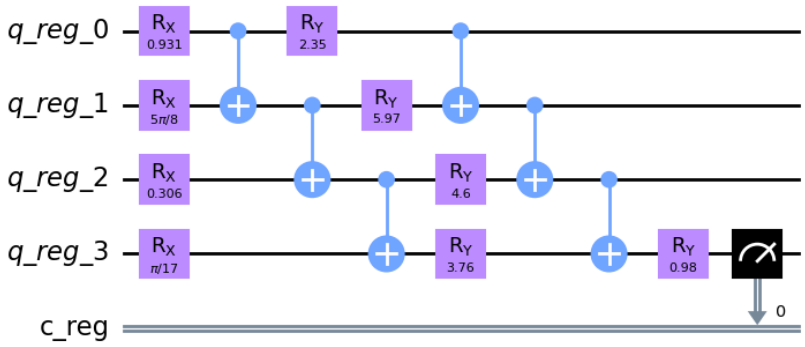
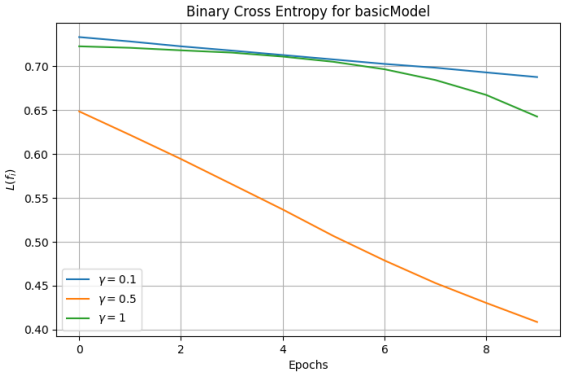
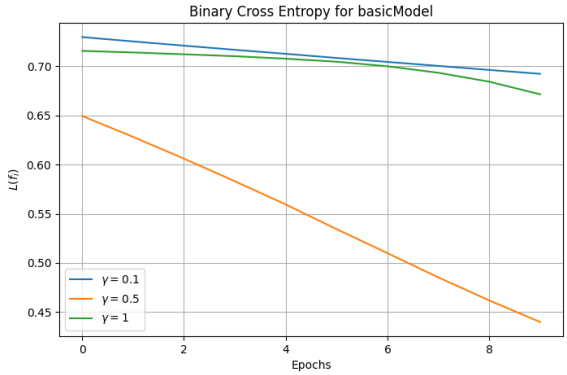


Figure 3: Circuit diagram of our most *basic* model. Note that the values in the R_X and R_Y gates correspond to values for the first sample in the first epoch. The initial R_X gates represent the encoding of features to corresponding qubits as well as the coupling of the different qubits through CNOT gates. The R_Y gates represent the ansatz for the model and adds a rotation according to the weights θ. the R_Y gate at the end serves as a bias to the model before the measurement represented by the needle on the gauge.

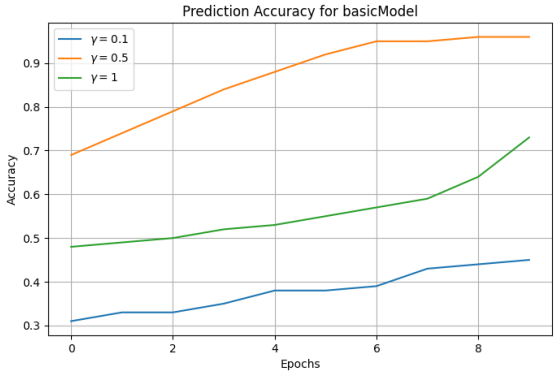
Executing the model over 10 epochs gives us an image of an initially high cross entropy for both 1000 and 10000 'shots' of the circuit and a fairly low accuracy. Performance measurements binary cross entropy and prediction accuracy as a function of the learning rate γ and the number of shots used is shown in figure 4.



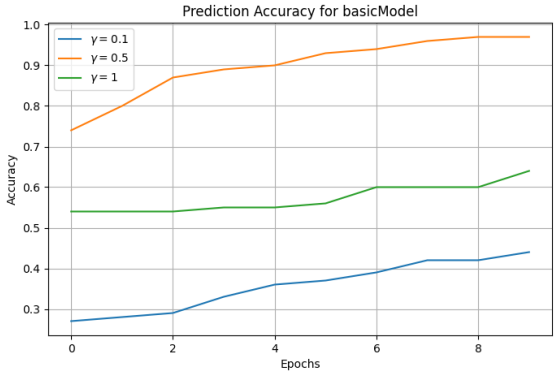
(a) shots = 1000



(b) shots = 10000



(c) shots = 1000



(d) shots = 10000

Figure 4: Binary cross entropy and accuracy for the basic model with single encoding and single ansatz. the learningrate $\gamma = 0.5$ has a significant gap with regards to the 2 other variables for all 4 plots. For learning rate $\gamma = 1$ the entropy finds a minimum at 0.40, while the accuracy approaches 1 after 10 epochs.

Figure 4 shows in both (a) and (b) a steep descent of the entropy for $\gamma = 0.5$ while the 1 and 0.1 remain high. The $\gamma = 1$ does seem to drop further than 0.1. For the accuracy in (c) and (d), $\gamma = 0.5$ outperforms the 2 extremes by reaching a near 100% accuracy after 6 to 8 epochs.

4.2 Double Ansatz

Following is the '*double ansatz*' model visualized in figure 5.

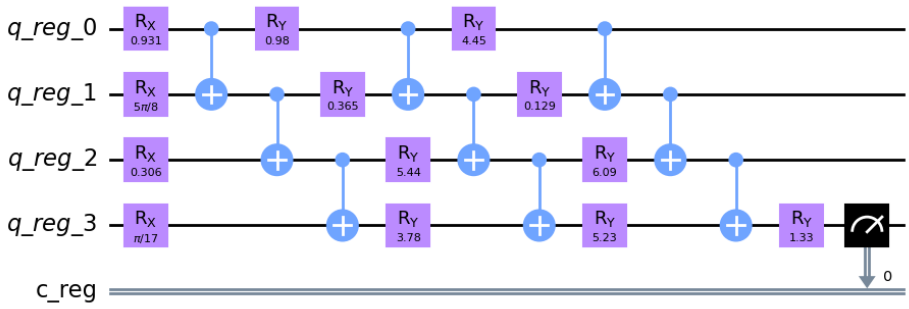
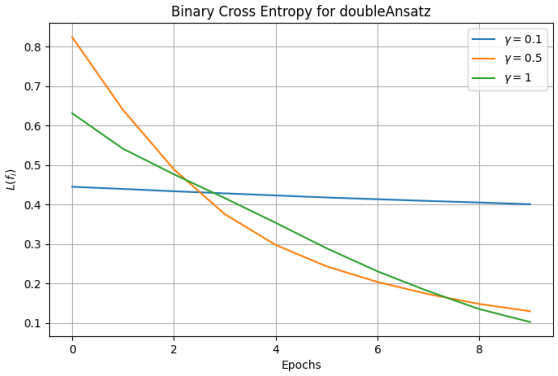
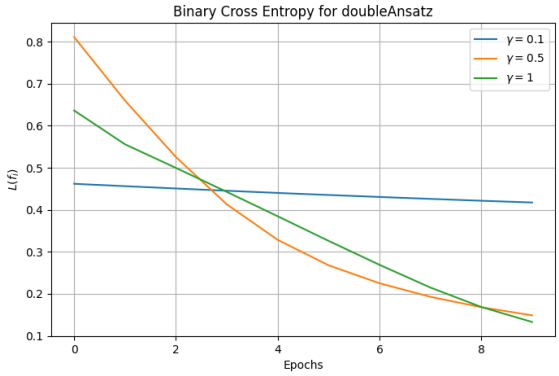


Figure 5: Circuit diagram of the *double ansatz* model. The values shown do not represent the final model values. The first 2 rows of gates are as in figure 3. Following this is a second row of rotations adding additional tune-able elements to θ.

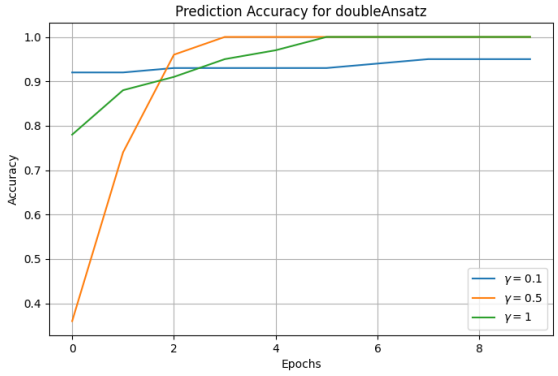
Figure 6 shows the binary cross-entropy and accuracy over 10 epochs of running the model with both 1000 and 10000 shots. Both (a) and (b) shows a near static value of cross-entropy for $\gamma = 0.1$ while 0.5 and 1 fall of seemingly exponentially. In both cases, the lower cross-entropy approaches 0.1. For the accuracy in (c) and (d) there is a difference of at most .1 or 10% between the learning rates after 2 epochs. The $\gamma = 0.5$ accuracy is worse than a random guess to begin with, but quickly grows to 100% with $\gamma = 1$. The minimal learning rate keeps a consistently high accuracy.



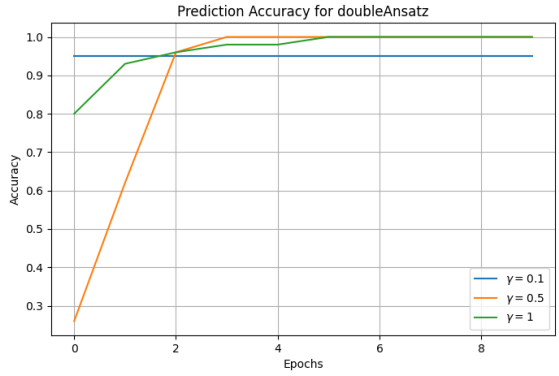
(a) shots = 1000



(b) shots = 10000



(c) shots = 1000



(d) shots = 10000

Figure 6: Binary cross entropy and accuracy for the double ansatz model. For (a) and (b) the learning rate $\gamma = 0.1$ drops off much slower than the 2 greater values. the $\gamma = 0.5$ and 1 follow a similar curve. $\gamma = 1$ achieves the lowest cross-entropy. $\gamma = 0.5$ shows an almost exponential shape. Prediction accuracy in (c) and (d) is above 90% for all 3 learning rates, with $\gamma = 0.5$ and 1 reaching 100% accuracy after 5 epochs.

4.3 Double Encoding

Illustrated in figure 7 is the '*double encoding*' model. The circuit is executed for each sample over 10 epochs.

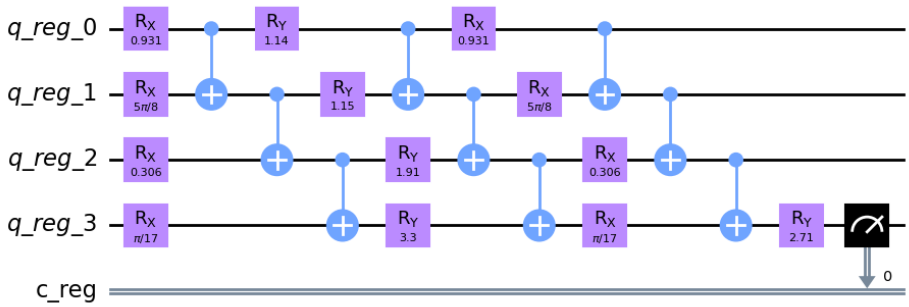
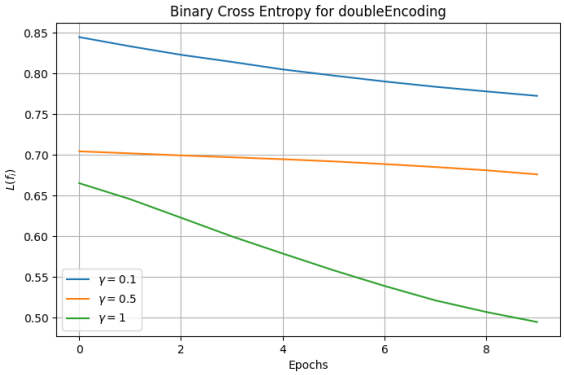


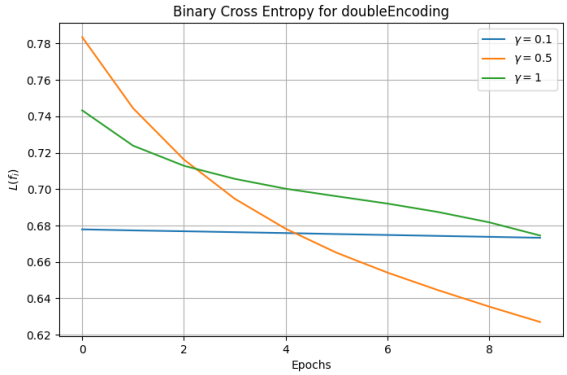
Figure 7: Circuit diagram of the *double encoding* model, same as before with regards to the gate values. The diagram holds the same basic setup as in figure 3, with the addition of a re-encoding of the features to the circuit before bias and measurement.

with performance metrics Figure 8 has a fairly spread out cross-entropy for the 1000 shots in (a) while the greater number of shots in (b) has more convergent cross-entropy between the learning rates. $\gamma = 1$ gets a cross-entropy of 0.5 after 10 epochs in (a), while the $\gamma = 0.5$ learning rate in (b) wins out at a cross-entropy of 0.62. The other 2 learning rates results in similar cross-entropy at the end in (b)

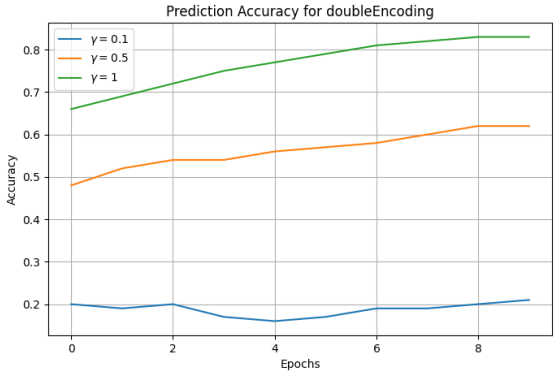
For accuracy $\gamma = 0.1$ has a fairly static value for both (c) and (d), but (c) has an accuracy of 0.2 which is significantly worse than a random guess. (d) sees a significant improvement over random guesses, but the adjustment of accuracy over epochs at the low learning rate is comparatively small. Both $\gamma = 0.5$ and 1 grow to over 0.6 and 0.8 accuracy respectively in (c). For (d) the larger learning rates start at a poor accuracy and grow over the epochs to 0.7 equaling or surpassing the accuracy of $\gamma = 0.1$.



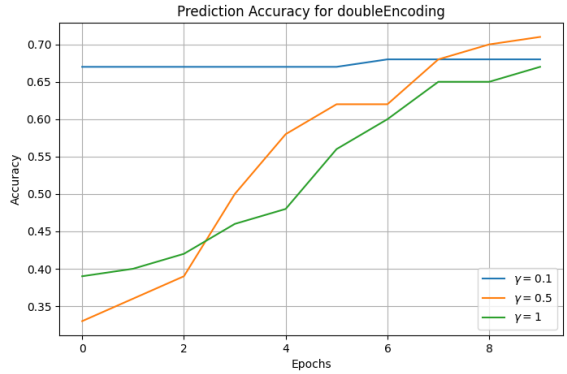
(a) shots = 1000



(b) shots = 10000



(c) shots = 1000



(d) shots = 10000

Figure 8: Binary cross entropy and accuracy for the double encoding model. The cross-entropy between the number of shots in (a) and (b) here have differing profiles. for the 1000 shot model the largest learning rate $\gamma = 1$ gets down to 0.5, against the rough 0.7 of $\gamma = 0.5$. The 10000 case in (b) has $\gamma = 0.5$ as the lowest entropy instance, but the numerical value stays at 0.63 which is lower than in the 1000 shot case, but higher than the lowest value of (a). The accuracies of (c) correspond to the entropies in (a), with $\gamma = 1$ reaching an accuracy of over 0.8, with $\gamma = 0.5$ and 0.1 bellow. *gamma* $= 0.5$ is at about the rate of random guessing while 0.1 predicts far less accurately than pure guesswork. The 10000 shot case in (d) has a greater clustering of eccuracy at the ends whit $\gamma = 0.1$ mostly constant at above 0.65 while the $\gamma = 0.5$ grows to above 0.7 after 10 epochs and $\gamma = 1$ climbing to an accuracy of similar to $\gamma = 0.1$.

4.4 Less entangled(Bob Ross) model

When running our simulations we did not notice we had accidentally coded our double encoding model 'wrong'. We therefore ended up with an additional model we have decided to call the '*less entangled*' model[4] as it is like the double encoding model except it is missing the entangling portion of our ansatz. The circuit is illustrated in figure 9.

Figure 10 shows cross-entropy in the range $[0.8, 0.4]$ for both (a) and (b) with very similar curves between the 2 sets of shots. The lowest entropy in both cases is the learning rate $\gamma = 1$. The accuracy shows varies more between (c) and (d) both $\gamma = 0.5$ and 1 reaching an accuracy of about 0.9.

[4]We were debating on calling it the Bob Ross model simply because it turned out to be a 'happy little accident' that we ended up with more than intended data to present.

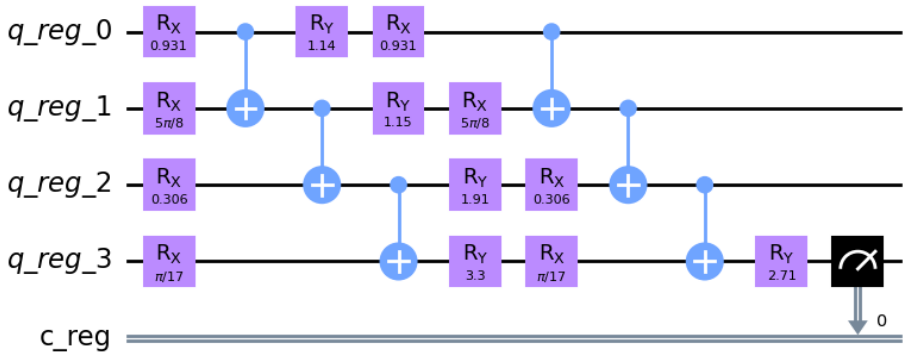
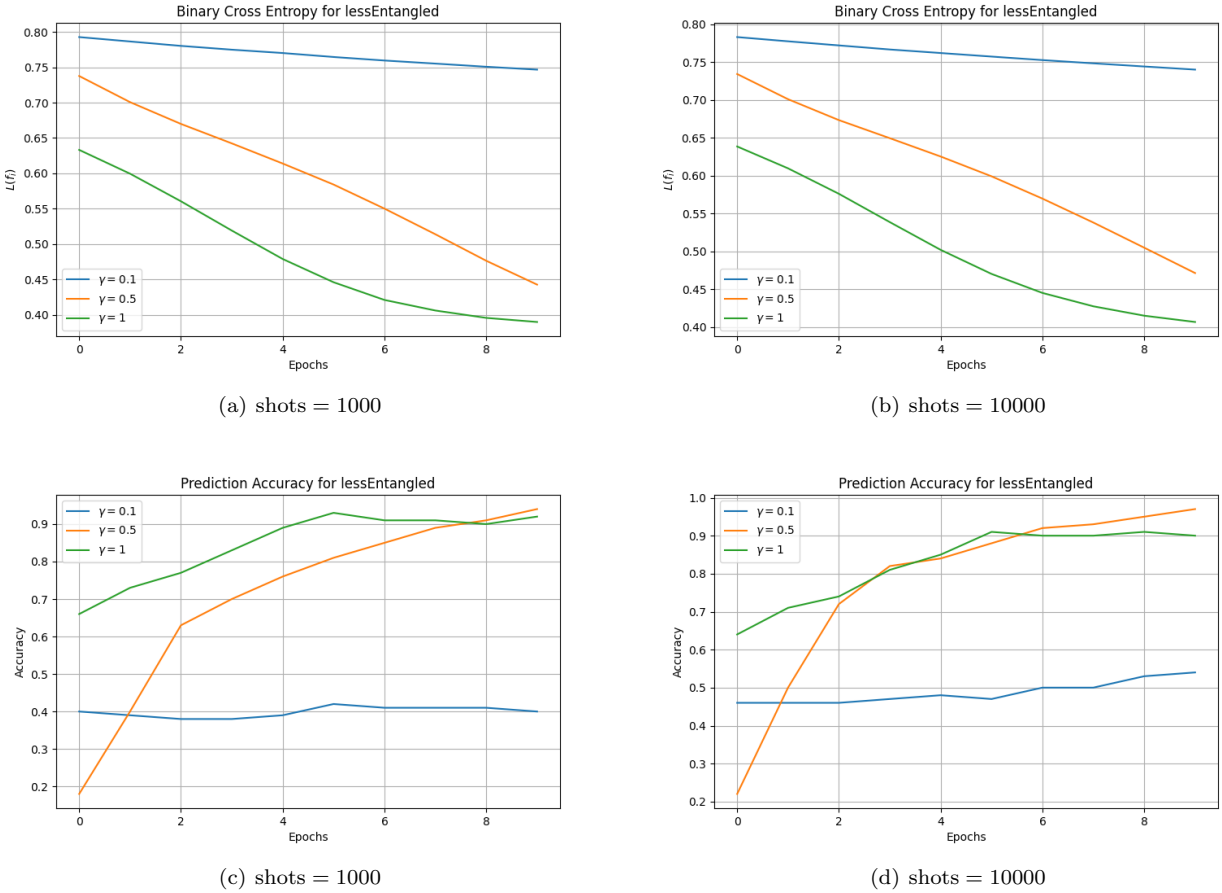


Figure 9: The circuit diagram for the less entangled model. the initial encoding and ansatz is performed almost like in fig 3, except the entanglement isn't performed afterwards. The application of the second encoding then goes on the model before entangling happens between the qubits after the weights.



(a) shots = 1000

(b) shots = 10000

(c) shots = 1000

(d) shots = 10000

Figure 10: Binary cross entropy and accuracy for the less entangled model. For cross-entropy the $\gamma = 1$ got the lowest result in both cases. Both (a) and (b) had similar cross-entropy results with the lowest winding up at about 0.4. The accuracy differs a little more between the 2 models. (d) has the highest end accuracy with $\gamma = 0.5$ reaching close to 1. (c) has a more similar accuracy between $\gamma = 0.5$ and 1 with both crossing slightly above 0.9 accuracy, for (c) the accuracy of $\gamma = 1$ dips down slightly before increasing again at the end.

4.5 Double Ansatz Double Encoding

The last model tested, the '*double ansatz double encoding*' model illustrated in figure 11. This one too is trained over 10 epochs.

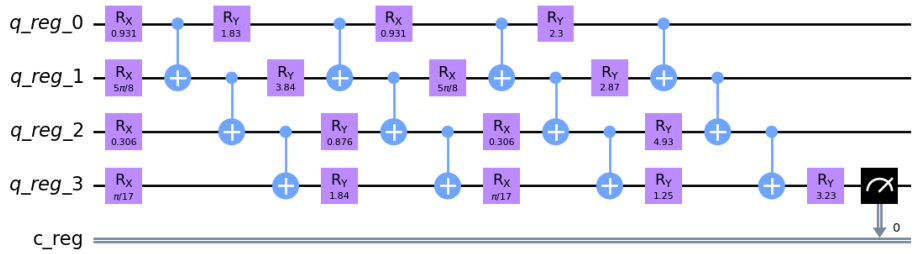
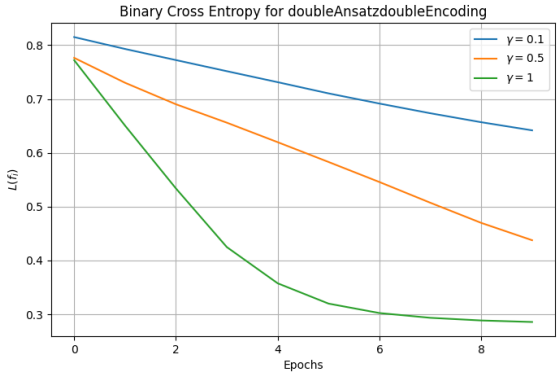
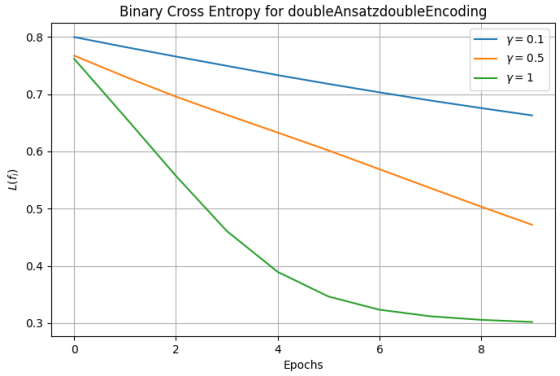


Figure 11: Double ansatz double encoding model. This circuit is essentially a doubling of the basic model in figure 3. this aims to increase the polynomial degree of the model as well as increasing the ammount of tunable parameters for the model.

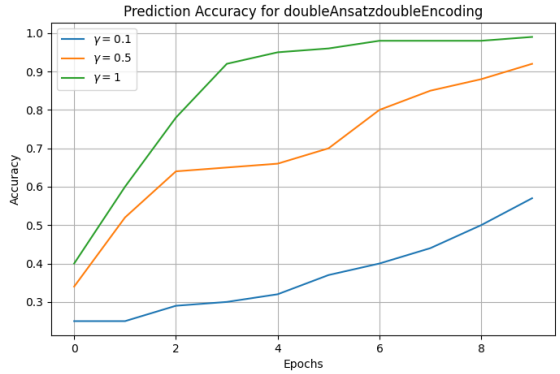
Figure 12 shows a learning rate $\gamma = 1$ whose cross-entropy quickly falls off and seems to flatten after around 6 epochs at a little less than 0.3 for (a) and a little over for (b). This learning rate is also notable for the exponential drop-off against the more linear change of the other 2. For both sets of runs, (c) and (d) shows $\gamma = 1$ reaching about 100% accuracy after 4 to 6 epochs. $\gamma = 0.5$ increases in accuracy with epochs and for the epochs run, $\gamma = 0.1$ falls behind the other 2.



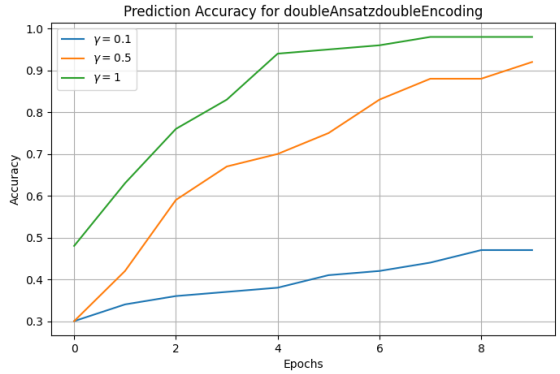
(a) shots = 1000



(b) shots = 10000



(c) shots = 1000



(d) shots = 10000

Figure 12: Binary cross entropy and accuracy for the double ansatz double encoding model. (a) and (b) both show cross-entropy down to 0.3 with the learning rate $\gamma = 1$ plateauing at around 6 epochs. For (c) and (d) the accuracy of $\gamma = 1$ reaches 1 at about 6 epochs and maintains through the simulation.

4.6 Best Performing Models

Finally, table 1 shows our best model fits for each of the 5 models tested. For 3 of the 5, a learning rate of $\gamma = 0.5$ provided the best fit. 3 out of 5 predicted better at 1000 rather than at 10000 shots per circuit. Both the greatest accuracy and lowest cross-entropy was the *double ansats* model, see diagram 5. Close was the second highest accuracy and the second lowest cross-entropy with *double ansats, double encoder* model.

Model	Acc.	Loss	γ	n_{shots}	Model Parameters
Basic	0.97	0.44	0.5	10000	$\theta_1 = 1.2429$ $\theta_2 = 0.2471$ $\theta_3 = 5.8989$ $\theta_4 = 3.4169$ $\theta_5 = 3.6160$
Double Ans.	1.00	0.13	0.5	1000	$\theta_1 = 3.0027$ $\theta_2 = 6.0001$ $\theta_3 = 1.5082$ $\theta_4 = 3.5964$ $\theta_5 = 3.1384$ $\theta_6 = 0.3445$ $\theta_7 = 4.5028$ $\theta_8 = 1.5651$ $\theta_9 = -0.1762$
Double Enc.	0.83	0.49	1	1000	$\theta_1 = 0.1860$ $\theta_2 = 6.1587$ $\theta_3 = 5.2523$ $\theta_4 = 2.1526$ $\theta_5 = 0.0926$
Less Entgl.	0.97	0.47	0.5	10000	$\theta_1 = 3.2799$ $\theta_2 = 3.3743$ $\theta_3 = 1.3437$ $\theta_4 = 5.7130$ $\theta_5 = 3.8065$
Double AnsEnc.	0.99	0.29	1	1000	$\theta_1 = 4.5055$ $\theta_2 = 4.7880$ $\theta_3 = 1.1718$ $\theta_4 = 3.0730$ $\theta_5 = 0.3079$ $\theta_6 = 5.7312$ $\theta_7 = 2.8377$ $\theta_8 = 1.5145$ $\theta_9 = 2.23e^{-4}$

Table 1: Best performing models with regards to accuracy. The full data is available at `https://github.com/ageriksen/Quantum-Machine-Learning/tree/main/src/data`. The lowest and second lowest cross-entropy as well as the highest and second highest accuracy are the double ansats and double ansatz-double encoding models. The majority of the best fits are from the 1000 shot circuits and most have a learning rate of $\gamma = 0.5$.

5 Discussion

A good amount of the challenge of quantum computing and quantum machine learning lies in accessing the 2^n-dimensional Hilbert space of the system. The different encoding- and ansatz methods here are an attempt at encoding the system into the space available in configurations to explore the space.

The data set samples are not scrambled prior to input, but the model currently only conducts training and there are no resampling methods or splitting of the data. Therefore, so long as we go through equally many of each class the model should predict based on features and not a sample ordering.

5.1 The Models

The basic model performed quite well overall. the learning rate $\gamma = 0.5$ gave a cross-entropy of about 0.4, which among the higher cross-entropies for good results, but the accuracy of prediction reached nearly 100%. During testing there seemed to be "magic" seeds which would either result in a nearly perfect prediction from the word go, or close to no correct predictions. For the poor seeds, this meant very gradual increase in the accuracy and slow descent for the cross-entropy. The model therefore could be said to be dependent on starting conditions and does not produce the best entropy overall.

The double ansatz allows the model nearly double the tune-able parameters θ and the cross-entropy responds well to this. the poorest performing learning rate, $\gamma = 0.1$ decreases very slightly through cross-entropy 0.4. This is then better than the best fit of the basic model. Here, the lowest cross-entropy lies with learning rate $\gamma = 1$ at about 0.1, though the $\gamma = 0.5$ learning rate is follows along on a more exponential drop. As for the accuracy, the $\gamma = 0.5$ learning rate here reaches 100% at 5 epochs with a steep incline from less than random guessing accuracy at 40%. This is probably the best fit amongst those we tested.

Double encoding on the circuit allows for accessing the Hilbert space with a higher polynomial degree of our model. Somewhat like how a 3rd degree polynomial

might fit some data better than a 2nd degree. This model has a good deal of difference in the cross-entropy and accuracy between the model's number of shots. The 1000 shot model has the lowest cross-entropy and the highest accuracy, with the learning rate $\gamma = 1$. While the 10000 case has a greater grouping of the various learning rates, with a steeper ascent for the learning rates $\gamma = 0.5$ and 1. The $\gamma = 0.1$ learning rate remains almost static across both runs of shots.

For the less entangled model, the profiles of the different shot circuits are quite similar, with some differences in the accuracy of the runs. The cross-entropy is along with the basic model at the higher end of final cross-entropy we found at about 0.4. The $\gamma = 0.1$ learning rate changes little over the epochs and stays at about or bellow a random guess accuracy. The other learning rates to reach an accuracy rate of 0.9, so the model can predict quite well the species in training.

Finally, for the double ansatz, double encoding model there is a rapid fall off for the learning rate $\gamma = 1$, which falls off seemingly exponentially down to a cross-entropy of 0.3. This learning rate also reaches a 100% accuracy around the time where the cross-entropy flattens. The other 2 learning rates have a more linear shape in the cross-entropy and don't fall of nearly as quickly. The $\gamma = 0.5$ learning rate does come closer to the best fit here though not quite and some epochs later. This might taper off with increased epochs run or it might catch up and exceed the performance of $\gamma = 1$, but the data supports the latter learning rate now.

Overall performance seems to be best with the double ansatz and the double ansatz double encoding. This is the case for both accuracy and cross-entropy. These models both share the most tune-able parameters, which could indicate a strong connection there.

There also seems to be a better prediction for the case of 1000 shots rather than the 10000 shot circuit. Though we don't have a clear reason for this, it is possible that some "fuzziness" in the gradient calculations using the statistics dependent on the number of shots could benefit from the lack of precision. Perhaps a chance to move out of local minima.

5.2 Further discussion

The main area of interest was the encoding of the data from the classical set and into the quantum mechanical model to facilitate the prediction. The tendency as mentioned above seems to be that a greater number of tune-able parameters improve the performance of the model. The comparison ends where we have not tested whether this holds true for arbitrarily many model parameters. It is possible that increasing the number of parameters in the model would produce negligible improvements for the added cost of finding the gradients for all of them.

Similarly, the double encoding seems to provide little improvements over the basic model, but the extra encoding could be more responsive outside the number of epochs we simulated. Additionally, there seems to be sensitivity to the performance of the basic model depending on the input parameters. If the double encoding is more stable it could be preferable.

It is also likely that most of these models would hit a point where the entropy and accuracy would flatten out. There are a lot of models that seem to have near linear behaviour over the epochs tested, and this could be a poor representation of the result of further training beyond the 10 epochs currently used. The reason for the few epochs stems from issues during the final development of the code where the simulations would slow down per iteration, considerably so by the 4th or 5th epoch. A proper solution was not found in time to allow running the data and the setup of 10 epochs was decided.

During discussions of the issue, some issues emerged regarding the naming of the qubits in the circuits. this seemed to indicate creations of new circuits for each run of the model. With 3 calls of the model per sample for 100 samples per epoch meant that this quickly would take greater and greater toll on the machine. An attempt to solve this was made in explicitly naming the registers used so as to avoid creating new ones with each call. Explicitly deleting the circuits at the end did not improve performance. Another solution which was presented later of removing the quantum registers could also prove a solution given that the garbage collector was suggested to more easily handle this.

While there seemed to be some slowdown on a windows machine running the simulations, the issue disappeared after the meeting. Given the lack of time, the limited runs were performed and presented. The explicit naming solution did not seem to work on the linux machine attempting to run the simulations.

This project has mostly been concerned with documenting the effect of training a quantum model to predict classical data. Considerations have not been made with regards to over-fitting and bias in the model and no control test set has been used. In addition there has been no comparison with classical methods such as logistic regression.

In a sense, the Iris set can be quite simple and there are here only 4 features that are fairly distinctly separated, especially when removing 1 of the 3 classes to make use of a binary classification.

The models we used all measured the state onto a single bit. This compresses the state down to a single qubit. Methods utilizing more of the quantum state through examining the parity of the readouts from each qubit could show other aspects of the system.

Finally, the code is not optimised for performance as it stands. parallelisation could likely speed up the processing of the data quickly. This both in the simultaneous testing of models, shots and learning rates but

within each epoch the samples are largely independent until collecting the data at the end.

6 Conclusion

The project tested a subset of the Iris data set with the excision of the 3rd species. the sample features were scaled to 1 and sent into a *QML* class who trained a given model of θ values. A training method went through each sample and input them into a parameterized quantum circuit. The circuit encoded the features and model parameters in 5 different ways according to models specified on class initialization.

The 5 models were tested for 2 sets of shots, 1000 and 10000, determining how many simulations of the circuit would be taken into account for prediction of the set. For each epoch of the training, an average of the gradient for the model parameters over all samples was used for training the model. the learning rate was varied between $[0.1, 0.5, 1]$. Over all 5 models, the 1000 shot model and the larger 2 learning rates tended to perform better, 0.5 performing somewhat above 1.

After running through the epochs, the model parameters θ, the mean cross-entropy for over all samples per epoch and the accuracy of prediction per epoch was returned and stored in a data folder and later used for plotting and examining the results.

The 2 best models both applied the ansatz twice. So far there seems to be a coupling between number of tune-able parameters and the accuracy of the fit as well as the entropy. This then would be the results suggested by these experiments, although they are limited in the areas discussed above.

Further exploration would be useful in terms of exploring different gradient descent methods, such as ADAM and reading out predictions from the circuit through different methods than entangling all qubits down to a single measured qubit. Exploring different loss functions would likely also be interesting to check. Parallellization could likely speed up processing with splitting both the model types as well as the separate samples.

Further explorations of the models through both more variation in number of shots and learning rates as well as running the systems over more epochs and explore the greater shapes of the loss and accuracy. To test for dependency on the models for given initial conditions, varying seeds and initial model values would also be interesting to explore.

The performance of the model should be checked to a test set, rather than using the entire sample size for training. This would also allow checking for over-fitting on the model for the various encoder and ansatz permutations. A test set would also allow for greater comparisons of the model to classical methods.

The Iris set is fairly simple and performance here is not necessarily indicative of general performance. Exploring other sets, such as the Wisconsin cancer data would help provide greater insight into overall performance.

References

[1] Amira Abbas, David Sutter, Christa Zoufal, Aurélien Lucchi, Alessio Figalli, and Stefan Woerner. The power of quantum neural networks. 2020.

[2] Héctor Abraham, AduOffei, Rochisha Agarwal, Ismail Yunus Akhalwaya, Gadi Aleksandrowicz, Thomas Alexander, Matthew Amy, Eli Arbel, Arijit02, Abraham Asfaw, Artur Avkhadiev, Carlos Azaustre, AzizNgoueya, Abhik Banerjee, Aman Bansal, Panagiotis Barkoutsos, Ashish Barnawal, George Barron, George S. Barron, Luciano Bello, Yael Ben-Haim, Daniel Bevenius, Arjun Bhobe, Lev S. Bishop, Carsten Blank, Sorin Bolos, Samuel Bosch, Brandon, Sergey Bravyi, Bryce-Fuller, David Bucher, Artemiy Burov, Fran Cabrera, Padraic Calpin, Lauren Capelluto, Jorge Carballo, Ginés Carrascal, Adrian Chen, Chun-Fu Chen, Edward Chen, Jielun (Chris) Chen, Richard Chen, Jerry M. Chow, Spencer Churchill, Christian Claus, Christian Clauss, Romilly Cocking, Filipe Correa, Abigail J. Cross, Andrew W. Cross, Simon Cross, Juan Cruz-Benito, Chris Culver, Antonio D. Córcoles-Gonzales, Sean Dague, Tareq El Dandachi, Marcus Daniels, Matthieu Dartiailh, DavideFrr, Abdón Rodríguez Davila, Anton Dekusar, Delton Ding, Jun Doi, Eric Drechsler, Drew, Eugene Dumitrescu, Karel Dumon, Ivan Duran, Kareem EL-Safty, Eric Eastman, Grant Eberle, Pieter Eendebak, Daniel Egger, Mark Everitt, Paco Martín Fernández, Axel Hernández Ferrera, Romain Fouilland, FranckChevallier, Albert Frisch, Andreas Fuhrer, Bryce Fuller, MELVIN GEORGE, Julien Gacon, Borja Godoy Gago, Claudio Gambella, Jay M. Gambetta, Adhisha Gammanpila, Luis Garcia, Tanya Garg, Shelly Garion, Austin Gilliam, Aditya Giridharan, Juan Gomez-Mosquera, Gonzalo, Salvador de la Puente González, Jesse Gorzinski, Ian Gould, Donny Greenberg, Dmitry Grinko, Wen Guan, John A. Gunnels, Mikael Haglund, Isabel Haide, Ikko Hamamura, Omar Costa Hamido, Frank Harkins, Vojtech Havlicek, Joe Hellmers, Łukasz Herok, Stefan Hillmich, Hiroshi Horii, Connor Howington, Shaohan Hu, Wei Hu, Junye Huang, Rolf Huisman, Haruki Imai, Takashi Imamichi, Kazuaki Ishizaki, Raban Iten, Toshinari Itoko, JamesSeaward, Ali Javadi, Ali Javadi-Abhari, Wahaj Javed, Jessica, Madhav Jivrajani, Kiran Johns, Scott

Johnstun, Jonathan-Shoemaker, Vismai K, Tal Kachmann, Akshay Kale, Naoki Kanazawa, Kang-Bae, Anton Karazeev, Paul Kassebaum, Josh Kelso, Spencer King, Knabberjoe, Yuri Kobayashi, Arseny Kovyrshin, Rajiv Krishnakumar, Vivek Krishnan, Kevin Krsulich, Prasad Kumkar, Gawel Kus, Ryan LaRose, Enrique Lacal, Raphaël Lambert, John Lapeyre, Joe Latone, Scott Lawrence, Christina Lee, Gushu Li, Dennis Liu, Peng Liu, Yunho Maeng, Kahan Majmudar, Aleksei Malyshev, Joshua Manela, Jakub Marecek, Manoel Marques, Dmitri Maslov, Dolph Mathews, Atsushi Matsuo, Douglas T. McClure, Cameron McGarry, David McKay, Dan McPherson, Srujan Meesala, Thomas Metcalfe, Martin Mevissen, Andrew Meyer, Antonio Mezzacapo, Rohit Midha, Zlatko Minev, Abby Mitchell, Nikolaj Moll, Jhon Montanez, Gabriel Monteiro, Michael Duane Mooring, Renier Morales, Niall Moran, Mario Motta, MrF, Prakash Murali, Jan Müggenburg, David Nadlinger, Ken Nakanishi, Giacomo Nannicini, Paul Nation, Edwin Navarro, Yehuda Naveh, Scott Wyman Neagle, Patrick Neuweiler, Johan Nicander, Pradeep Niroula, Hassi Norlen, NuoWenLei, Lee James O'Riordan, Oluwatobi Ogunbayo, Pauline Ollitrault, Raul Otaolea, Steven Oud, Dan Padilha, Hanhee Paik, Soham Pal, Yuchen Pang, Vincent R. Pascuzzi, Simone Perriello, Anna Phan, Francesco Piro, Marco Pistoia, Christophe Piveteau, Pierre Pocreau, Alejandro Pozas-Kerstjens, Milos Prokop, Viktor Prutyanov, Daniel Puzzuoli, Jesús Pérez, Quintiii, Rafey Iqbal Rahman, Arun Raja, Nipun Ramagiri, Anirudh Rao, Rudy Raymond, Rafael Martín-Cuevas Redondo, Max Reuter, Julia Rice, Matt Riedemann, Marcello La Rocca, Diego M. Rodríguez, RohithKarur, Max Rossmannek, Mingi Ryu, Tharrmashastha SAPV, SamFerracin, Martin Sandberg, Hirmay Sandesara, Ritvik Sapra, Hayk Sargsyan, Aniruddha Sarkar, Ninad Sathaye, Bruno Schmitt, Chris Schnabel, Zachary Schoenfeld, Travis L. Scholten, Eddie Schoute, Joachim Schwarm, Ismael Faro Sertage, Kanav Setia, Nathan Shammah, Yunong Shi, Adenilton Silva, Andrea Simonetto, Nick Singstock, Yukio Siraichi, Iskandar Sitdikov, Seyon Sivarajah, Magnus Berg Sletfjerding, John A. Smolin, Mathias Soeken, Igor Olegovich Sokolov, Igor Sokolov, SooluThomas, Starfish, Dominik Steenken, Matt Stypulkoski, Shaojun Sun, Kevin J. Sung, Hitomi Takahashi, Tanvesh Takawale, Ivano Tavernelli, Charles Taylor, Pete Taylour, Soolu Thomas, Mathieu Tillet, Maddy Tod, Miroslav Tomasik, Enrique de la Torre, Kenso Trabing, Matthew Treinish, TrishaPe, Davindra Tulsi, Wes Turner, Yotam Vaknin, Carmen Recio Valcarce, Francois Varchon, Almudena Carrera Vazquez, Victor Villar, Desiree Vogt-Lee, Christophe Vuillot, James Weaver, Johannes Weidenfeller, Rafal Wieczorek, Jonathan A. Wildstrom, Erick Winston, Jack J. Woehr, Stefan Woerner, Ryan Woo, Christopher J. Wood, Ryan Wood, Stephen Wood, Steve Wood, James Wootton, Daniyar Yeralin, David Yonge-Mallo, Richard Young, Jessie Yu, Christopher Zachow, Laura Zdanski, Helena Zhang, Christa Zoufal, Zoufalc, a kapila, a matsuo, bcamorrison, brandhsn, nick bronn, brosand, chlorophyll zz, csseifms, dekel.meirom, dekelmeirom, dekool, dime10, drholmie, dtrenev, ehchen, elfrocampeador, faisaldebouni, fanizzamarco, gabrieleagl, gadial, galeinston, georgios ts, gruu, hhorii, hykavitha, jagunther, jliu45, jscott2, kanejess, klinvill, krutik2966, kurarrr, lerongil, ma5x, merav aharoni, michelle4654, ordmoj, sagar pahwa, rmoyard, saswati qiskit, scottkelso, sethmerkel, shaashwat, sternparky, strickroman, sumitpuri, tigerjack, toural, tsura crisaldo, vvilpas, welien, willhbang, yang.luh, yotamvakninibm, and Mantas Čepulkovskis. *Qiskit: An Open-source Framework for Quantum Computing*. 2019.

[3] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, Nov 2019.

[4] H. Chen, L. Wossnig, S. Severini, H. Neven, and M. Mohseni. Universal discriminative quantum neural networks. *Quantum Machine Intelligence*, 3(1), Dec 2020.

[5] Gavin E. Crooks. Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition, 2019.

[6] Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, and Dacheng Tao. Expressive power of parametrized quantum circuits. *Physical Review Research*, 2(3), Jul 2020.

[7] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[8] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[9] Francisco Javier Gil Vidal and Dirk Oliver Theis. Input redundancy for parameterized quantum circuits. *Frontiers in Physics*, 8:297, 2020.

[10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[11] Maria Schuld. *Supervised learning with quantum computers*. Springer, 2018.

[12] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, 103(3), Mar 2021.

[13] Sukin Sim, Peter D. Johnson, and Alán Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070, Oct 2019.

[14] Kristian Wold and Stian Bilek. Fys4411 project 2: Quantum machine learning. 2021.