A Quantum Approach to Machine Learning for Binary Classification

Charlie Davies, Harry Durnberger, and Stephen Mcsweeney
The University of Nottingham, School of Physics and Astronomy
(Dated: May 16, 2023)

Quantum machine learning, a blend of quantum mechanics and classical machine learning, remains a relatively new field of study in which unanswered questions vastly outweigh current definitive solutions. Due to the complexity of quantum mechanics and the computational power required for machine learning, exploring practical approaches to combining both fields is only recently made possible. This paper provides a simple analysis of quantum machine learning for binary classification using the MNIST dataset. The goal is to observe quantum machine learning in its most rudimentary form whilst measuring the effects of data encoding methods, ansatz repetitions and gradient approximation methods on the performance of the model.

I. INTRODUCTION

Machine learning, a sub-field of artificial intelligence, aims to emulate intelligent human behaviours by using data to train its understanding of patterns in numerical data [1]. It employs algorithms which, unlike traditional programming techniques, allow the software to learn through experience without explicit direction. Such algorithms are widely used in various applications such as recommendation systems, image recognition, text analysis and signal analysis [2]. This paper explores the intersection of both machine learning and quantum mechanics. The latter is a foundational theory in physics which explains the fundamental characteristics and behaviours of matter and energy at atomic and subatomic levels [3]. It introduces concepts such as qubits, superposition and entanglement which are the key differences between quantum and classical approaches to physics.

Similar to classical bits, qubits define a range of states a single piece of information can sit in at any given time [4]. However, unlike classical bits which sit as either 0 or 1, qubits, with the help of superposition, can sit within any variation of multiple states at the same time. The state of a qubit is given as:

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad |\alpha|^2 + |\beta|^2 = 1$$

For any given qubit Ψ, probabilistic variables are applied to states 0 and 1. The addition of the absolute α and β values squared always equals 1. The state of a qubit exists within a Bloch Sphere [5]. As shown in FIG. 1, the state of a qubit is defined by two angles, ϑ and φ.

The basis states of a qubit are represented as follows. Any other state is a superposition of these basis states.

$$\begin{aligned} If \quad \Theta = 0 \rightarrow \quad |\Psi\rangle = |0\rangle \\ If \quad \Theta = \pi \rightarrow \quad |\Psi\rangle = |1\rangle \end{aligned}$$

$$\begin{aligned} |\Psi*\rangle : \Theta\tfrac{\pi}{2}, \quad \phi = 0 \\ |\Psi*\rangle : \tfrac{1}{\sqrt{2}}|0\rangle + \tfrac{1}{\sqrt{2}}|1\rangle \end{aligned}$$

Superposition is introduced as a Hadamard quantum logic gate [7]. This is what allows the qubit to exist in the continuous range between 0 and 1. The Hadamard formula can be observed in the last basis state shown above. For the sake of brevity, the final concept to discuss in a brief overview of quantum mechanics is entanglement [8]. This idea suggests that the states of two subatomic particles can be connected regardless of their distance in space. If the state of one changes, so does that and it does so in tandem with the other. In the field of physics, entanglement typically occurs in one of four ways. This can be either entanglement by birth, second-generation entanglement, accident or by interaction [9]. However, in quantum machine learning this phenomenon is simulated via CNOT gates [10] in Pennylane.

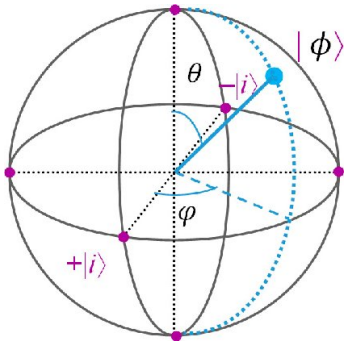

FIG. 1. [6] Bloch Sphere

Combining these concepts with machine learning provides a powerful tool for some very specific applications where classical machine learning falls short [11]. These include Shor's factoring algorithm and quantum simulations where classical computation takes far longer. This paper explores a rudimentary assessment of quantum machine learning via binary classification. Some experiments have been run to measure the effects of image encoding, ansatz variations and optimisation methods on the final model evaluation metrics and convergence times over epochs. The MNIST dataset [12] has been used due to its simplicity. Classical machine learning can very easily separate the classes which are numbers from 0-9. The data consists of 60,000 evenly represented grey-scale 28x28 example images of each class.

II. METHODOLOGY

As discussed in the previous section, this paper focuses on running experiments in three different scenarios. In order, these experiments revolve around changes to image encoding, ansatz variations and optimisation methods. At each stage, the best results from the previous set of experiments are used as a standardisation for the following set. In other words, the best results for image encoding are used across all tests for ansatz variation, and these with the best ansatz are then used across all optimisation scenarios. To move towards obtaining the desired metrics, the first step is to determine which programming language and its associated modules are used for the project. All models have been programmed and run in Python 3.11 [13] along with TensorFlow [14], Pennylane [15], Pandas [16], NumPy [17], and Matplotlib.pyplot [18]. Using Pennylane, the quantum circuit can be instantiated via QNode where the encoding, wires, weights, differentiation method and ansatz can be easily defined and manipulated with Python code. This can then be formatted as a typical TensorFlow layer meaning the rest of the steps are the same as classical machine learning in Python. These steps include defining the overall architecture, the input shapes and functions for predicting and evaluating the performance of the model. NumPy provides tools for easy manipulation of data structures and numerical data and Pandas offers easy solutions for formatting and exporting this data. Finally, Matplotlib.pyplot generated the figures used in the paper.

In each of the following steps, a description of each method used should give some insight into the inner machinations of each technique and the differences between them. In the first set of experiments, each model is run for 2 layers over 20 epochs. The adjoint differentiation method [19] is used along with the Adam optimiser [20] as a default starting point. Since classical backpropagation cannot be utilised in quantum circuits, adjoint differentiation calculates the gradient by constructing an adjoint circuit, taking the complex conjugate of all the gates in the original circuit and running it through in reverse. The expectation value of parameter-dependent operators is used to measure the gradient of the cost. Adam is then used to adjust the weights in the network. The evaluation metrics used to measure the performance of each model are accuracy and loss. A custom accuracy function is used to calculate the differences between tensor values, in this case, the outputs and the expected outputs. Hinge loss is used to measure the numerical difference between incorrect classifications. The formula for hinge loss is defined as:

$$L = max(0, 1 - y * f(x)) \rightarrow y = actual, f(x) = prediction$$

Here, the loss is measured as a maximum function returning 0 if the result is positive, else the result itself.

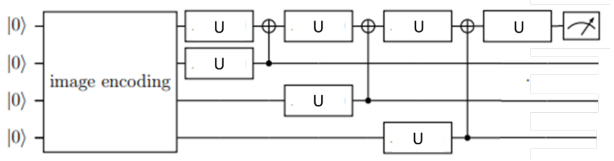

FIG. 2. Ansatz Variation 1

Finally, an ansatz, or quantum circuit, is required to run through quantum manipulations of the qubits. The example in FIG. 2 is the first variation used in this paper. It features four qubits, initialising in state 0, as four wires of a circuit. After each unitary, a CNOT gate is applied four times before measuring the state and expectation value of the circuit. In quantum circuits, time advances from left to right. At first, the selected image encoding is applied before the rotations [21] and entangling are implemented. This is just one example but realistically any number of rotations and entanglements can be applied. Each rotation is governed by Θ parameters. They are adjusted during the training process in an attempt to minimise the loss. As with classical machine learning, this quantum circuit can be repeated over many layers to increase accuracy. However, this drastically affects the time the simulations take to run due to the complexity of the model. The final aspect of the circuit is measuring the state. In this case, the state of the first qubit is measured after all operations are applied. During observation, the state of the qubit collapses into either a 0 or a 1, and the result corresponds to a prediction of the classes in the data based on the adjusted parameters. The difference of the result to the expectation is used to backpropagate over the network and update them.

A. Image Encoding

Image encoding is the process of restructuring the data in an image, in this case, a 28x28 matrix, for preparing it as an input into a machine learning algorithm [22]. Three different methods of encoding are explored in this paper. These are basis encoding, amplitude encoding and Flexible Representation of Quantum Images (FRQI). In quantum machine learning, basis encoding is the most representative of the original data since each pixel in the image is assigned a qubit. The normalised greyscale value of each pixel is encoded by a Pauli-Y, or σ_2 [23], rotation with a corresponding angle. For example, in each pixel:

$$x \in (0, 1) \rightarrow R_Y(\pi x)|0\rangle = \cos\left(\frac{\pi x}{2}\right)|0\rangle - \sin\left(\frac{\pi x}{2}\right)|1\rangle$$

The greyscale value of the pixel is then extracted by measuring the probability amplitude of the corresponding qubit, using the following equation:

$$x = (2/\pi)\arccos(\sqrt{p(|0\rangle)})$$

Using this method, basis encoding provides a direct mapping of pixels in the image to the qubit states. However, a major flaw of this method is that, since the images convert from 28x28 to 1x784, full encoding would demand 784 qubits from a quantum computer. Given the most advanced quantum computers currently run around 400 high-quality qubits [24], this is not a feasible solution in a real-world scenario. For the purposes of demonstration in the project, the dimensions of the images have been reduced to a 1x9 slice of the middle portion of each image, so that only 9 qubits are required. In this process, the majority of the information is lost and this inherent flaw will be reflected in the discussion portion of the paper.

The second method, amplitude encoding, takes advantage of exponential increases in possible states with each addition of a qubit [21]. This means an N-qubit system can be represented by 2^N complex numbers. The image is fully represented in the probability amplitudes for each basis state. Therefore, an image with M pixels can be completely represented within $log_2 M$ qubits. In this project, 28x28 images can be represented by 10 qubits. For each pixel value x_i, the normalised value is:

$$y_i = \frac{x_i}{\sum_i x_i^2}$$

Alternatively, amplitude encoding can be represented as:

$$|\psi\rangle = \sum_i y_i |i\rangle$$

Where $|i\rangle$ is the basis state and the binary representation of i. The probabilities of each bitstring are measured and the normalised values of each pixel are calculated by:

$$y_i = \sqrt{p(|i\rangle)}$$

The drawback to this method is that the normalisation of each pixel value means the information about the contrast in the image is lost which may potentially impact the final performance of the model. The third method observed in Flexible Representation of Quantum Images [25]. Whilst amplitude encoding alleviates the issue with qubit requirements by reducing pixel information on a logarithmic scale, information is still lost. FRQI addresses both issues by using the same encoding method as amplitude encoding but by also employing an extra qubit to store the value of each pixel directly. These additional qubits are known as positional encoding, or label, qubits and value encoding, or value, qubits. By entangling these qubits, each image is represented as:

$$|\psi\rangle = \sum_i |i\rangle \otimes \left(\cos\left(\frac{\pi x_i}{2}\right)|0\rangle + \sin\left(\frac{\pi x_i}{2}\right)|1\rangle\right)$$

Using this method, 11 qubits can be used to fully represent all information in the images without any loss. This, therefore, should stand the highest chance of reaching the best accuracy across the tests. During the experiments run for image encoding, the aforementioned defaults and ansatz are used. The only changes are the number of qubits simulated and the required code for the different methods. Each test was run for binary classification of both 0/1 and 3/6 classes. The best results from these tests are taken as the new default for the next set.

B. Ansatz Variations

The second set of experiments run revolved around the default ansatz against another custom solution to measure the effects on performance. The two designs are run for 1, 2, 3 and 4 layers [26]. The structure of the second ansatz design can be observed in FIG.3.

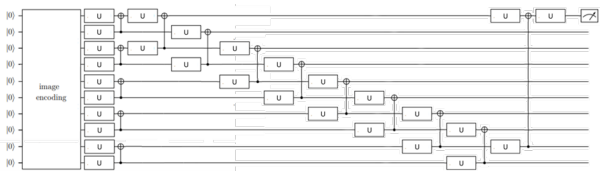

FIG. 3. Ansatz Variation 2

In this diagram, the U, or unitary, gates represent the general rotations applied to each qubit. These are either R_x or R_y rotations applied at angles defined by learnable parameters. After each unitary is a CNOT gate which applies the entanglement. The more qubits are entangled with each other, the higher the number of states the circuit is able to utilise increases exponentially. Only the first qubit in the first ansatz is measured since all others are entangled in it. The second one follows a more complex design. This circuit offers an approach where entanglement is distributed across all qubits. This could allow the circuit to find more complex patterns in each image at the sacrifice of training [27]. This ansatz may also perform better when taking multiple measurements as the output.

C. Optimisation Techniques

Finally, the last set of experiments revolves around changes in the optimisation method used to calculate the gradients with regard to loss for each parameter. All tests prior have used the Adam [28] optimiser with the adjoint differentiation [29] method of calculating gradients. For finite difference approximation [30] and parameter shift methods [31], Pennylane's GradientDescentOptimizer [32] class is utilised. This simple optimiser computes the new value of a parameter by subtracting an amount proportional to the estimate of the gradient of the loss with respect to that parameter. Finite difference approximation estimates the gradient by perturbing parameters by minute positive and negative proportions

and calculating the loss at each of these points. The gradient of the loss with respect to the parameter is estimated by subtracting the small proportion from the difference in the loss at both points.

Parameter shift is tested with the same optimisation technique. This method also calculates the gradient so long as the parameter governs a rotation around the Bloch Sphere. The parameter shift rule [33] states that the derivative of the expectation value of a measurement operator with respect to rotation angle ϑ can be computed by calculating the difference of the expectation values of the operator when ϑ is shifted by $+/-\frac{\pi}{2}$. This method is beneficial because it provides precise values for gradients as opposed to the more common approximation approaches.

After running tests using gradient descent, the final test replaced this with Simultaneous Perturbation Stochastic Approximation (SPSA) [34]. SPSA is an alteration of finite difference approximation whereby instead of evaluating the function twice for isolated changes to each input parameter, it evaluates the function only twice in total. To do this, a vector is randomly generated and observes the local gradient approximation in this direction. This significantly reduces the number of required measurements for large numbers of parameters. It is also guaranteed to converge on a solution, with the only drawback being inconsistency due to the random nature of the chosen vectors.

III. RESULTS

After running all tests and organising the data into graphs, the results can be observed. FIG. 4 demonstrates that basis encoding for 0 and 1 reaches the highest accuracy the fastest but is surpassed by amplitude encoding for 0 and 1 after epoch 12. Both experiments for 3 and 6 were inferior to those of 0 and 1. FRQI ran but failed to train in this instance.

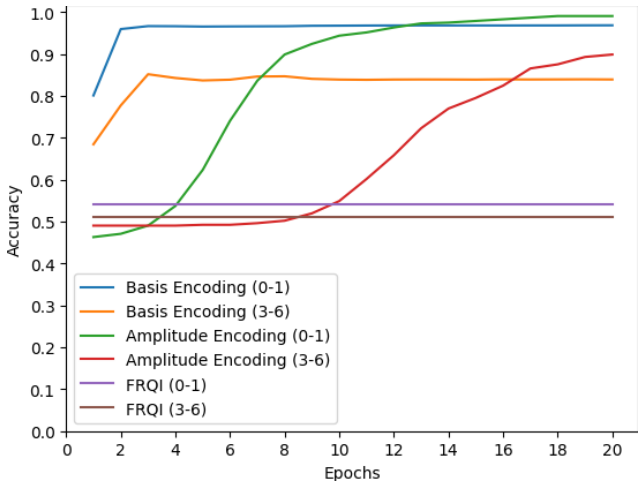

FIG. 4. Image Encoding Accuracy

FIG. 5 shows reasonable loss values very quickly for basis encoding. As expected, basis encoding generally performs the best follows by amplitude encoding for 0 and 1. Loss values across the board appear very high for the remaining methods but steady decreases in loss are maintained.

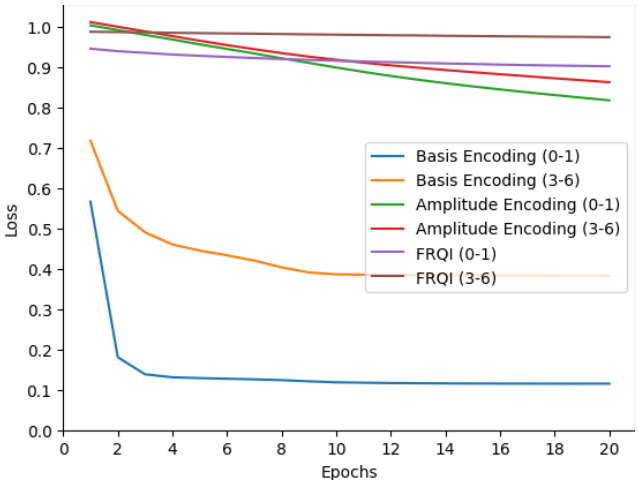

FIG. 5. Image Encoding Loss

FIG. 6 and FIG. 7 both suggest similar results across the layers regardless of the chosen circuit. The higher the number of layers the circuit runs through, the more accurate the model is generally, at least to begin with. The 1-layer models both perform poorly and the second ansatz appears to stabilise after 20 epochs. The 2-layer models perform well for both but reach higher accuracy in the first circuit. The 3-layer and 4-layer models converge on the final accuracy quicker but have much higher training times. For the first ansatz, the 2-layer model surpasses the 3-layer model and matches the 4-layer model by the end of the test. For the second ansatz, the 3-layer model surpasses the fourth. Due to excellent accuracy and relatively low training times, the 2-layer model of the first circuit is retained for further experimentation.

FIG. 8 and FIG. 9 show similar results to the first test where the loss is high but maintains steady decreases across epochs. The higher the layer, the lower the final loss.

FIG. 10 and 11 display the results of testing different optimisation techniques. SPSA was not successfully implemented therefore all three curves represent tests using the Adam optimiser for gradient descent. Each method converges on comparable levels of accuracy and loss. Adjoint differentiation appears to allow the model to learn faster initially although convergence on over 0.9 accuracy occurs at roughly the same time as using parameter shift. The model appears to learn the slowest using the finite difference approximation.

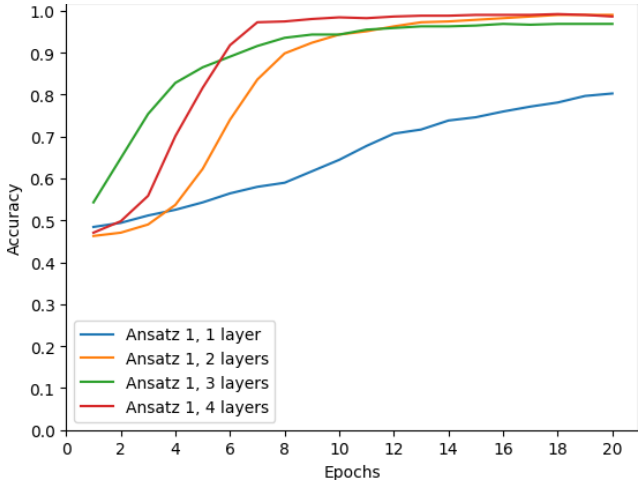

FIG. 6. Ansatz Variation 1 Accuracy

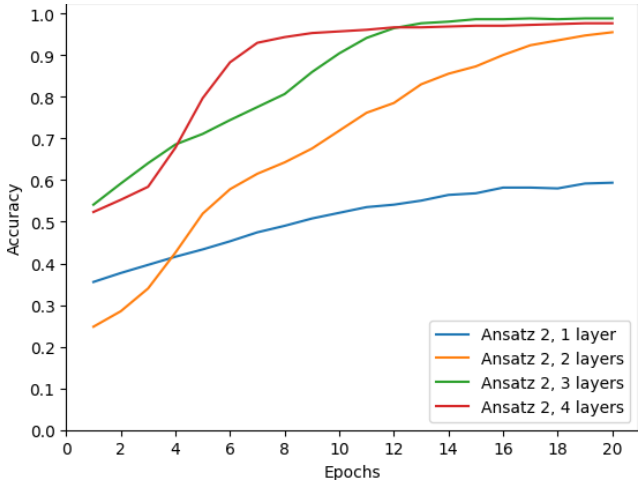

FIG. 7. Ansatz Variation 2 Accuracy

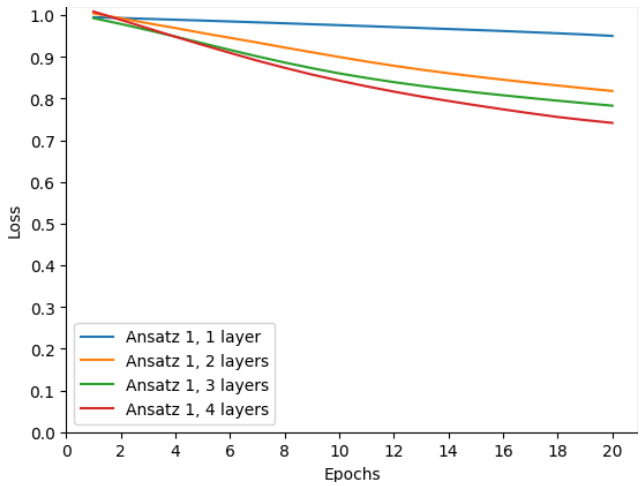

FIG. 8. Ansatz Variation 1 Loss

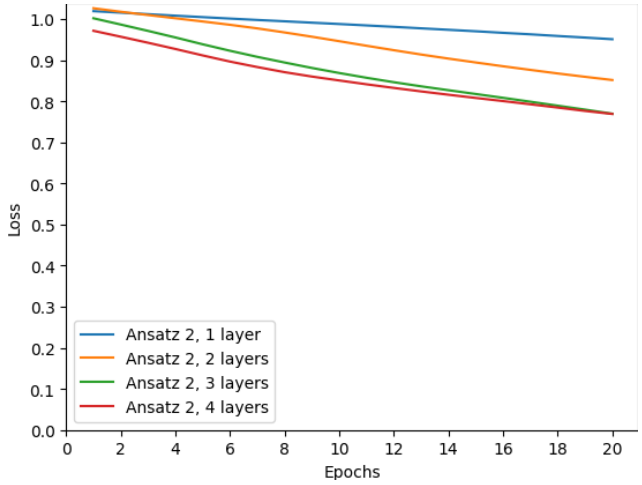

FIG. 9. Ansatz Variation 2 Loss

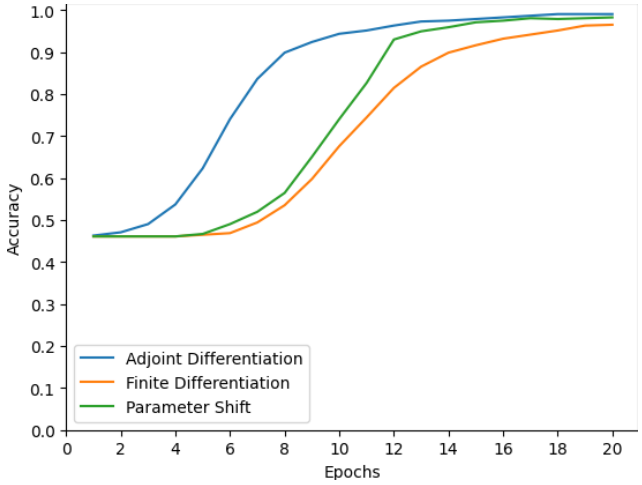

FIG. 10. Optimisation Accuracy

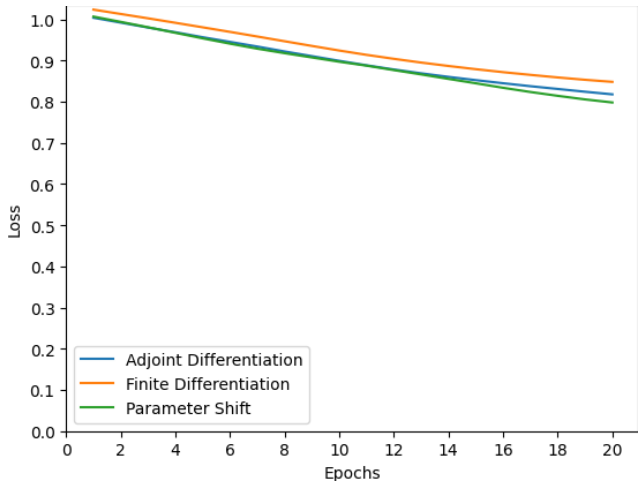

FIG. 11. Optimisation Loss

IV. DISCUSSION

In the first round of experiments, basis encoding for 0 and 1 classification performed the best in terms of both accuracy and loss. However, it is worth keeping in mind the shortfalls of this method. Whilst 0s and 1s are extremely easily distinguishable even with the vast majority of data removed, the same result would likely not be present in a scenario using more complex input data where the subtle information is more crucial. Further to this, due to the nature of the method, this encoding technique took approximately 8 times longer to train than the next best method, amplitude encoding for 0 and 1. For these reasons, this encoding method was maintained throughout the rest of the tests instead. It is unfortunate that FRQI did not yield results as this method was expected to be a serious contender for performance, but there is perhaps an issue in the code that needs addressing. Perhaps this method would have benefited from testing over more epochs. This would provide a more direct insight into the issue. The same can be said for the loss values for most of the methods. It is not understood why loss maintains such a high value despite the accuracy scores rising significantly in the more successful approaches. Again, this could perhaps be due to inaccuracies in the code.

During our exploration of ansatz designs, it was expected that the more complex the circuit the better the model would perform, and distributing the CNOT gates across all wires would lead to better performance with higher epoch counts or more complex data. In this instance, it appears that the complexity of the new circuit is wasted due to the lack of intricacies in the data and the low depth. The second ansatz does reach a higher accuracy using three layers, but the additional complexity comes at a computational cost. Perhaps this difference would be more apparent using data with more complicated features which can be tested in future investigations. For both instances of the circuit, there is a clear advantage to using more than one layer, however, the question of how many more would be optimal is not clear from these results. The models do appear to be learning faster with 3 and 4 layers compared to 2, but whether this corresponds to further benefits in adding further layers is not apparent and would need to be explored further with a more complex dataset.

In the final round of experiments, the optimisation techniques were evaluated. Unfortunately, the GradientDescentOptimizer implementation did not work so the test are run using the Adam optimiser. It was found that using the finite difference approximation and parameter shift to calculate gradients caused slower training compared to adjoint differentiation, although all methods eventually converged. Finite difference approximation achieved a marginally lower accuracy than both other methods. It is potentially the case, however, that the model trained using this technique simply requires more epochs to converge to the same accuracy as the others. The parameter shift method marginally outperformed adjoint eventually in accuracy but took longer initially to demonstrate improvement. Perhaps this margin would increase if the experiment ran for a greater number of epochs. Unfortunately, the SPSA optimiser was not successfully implemented into a working model and so the performance impact of this was not tested. This is unfortunate since it would be interesting to see if this technique allows the model to converge quicker than using the Adam optimiser. We would also expect it to lack the computational runtime drawback that a model using a technique like the finite difference approximation suffered from.

V. EXTRA EXPERIMENT: FASHION MNIST

As an extra test outside of the original scope, the Fashion MNIST dataset [35] was deployed on the best-performing model to measure how the complexity of the input data affected the results against the original. The model used employed amplitude encoding using the first quantum circuit over 2 layers.

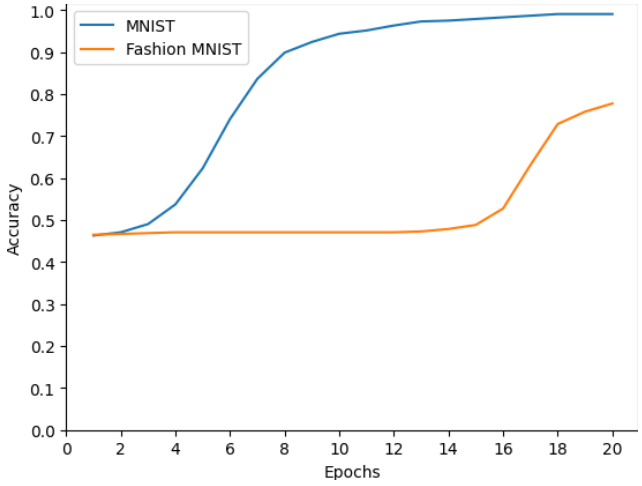

FIG. 12. MNIST vs Fashion MNIST Accuracy

As observed in FIG. 12 and FIG. 13 there is little difference in the loss metrics but the accuracy is drastically different. The MNIST model begins increasing accuracy almost instantaneously but the Fashion MNIST's performance is hindered for most of the test across the 20 epochs. It does however still train and it would be interesting to measure this over 40 or 60 epochs to see if the exact same architecture can reliably produce quality results on completely different data. This does demonstrate that the model is flexible and performs well. An interesting measure would be using FRQI on both instead as this would likely minimise the performance shortfall on the more complex data.

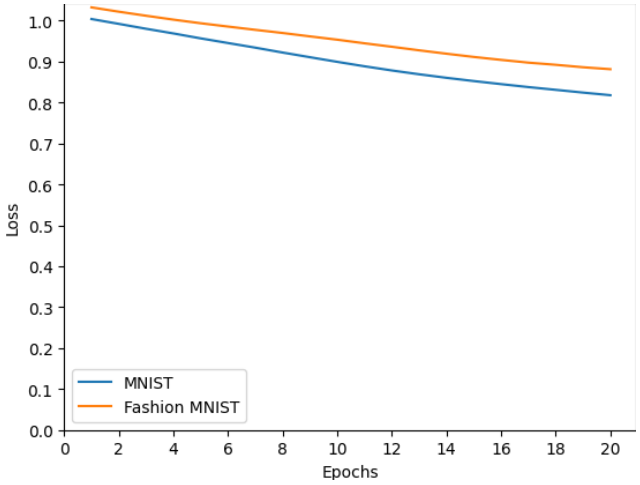

FIG. 13. MNIST vs Fashion MNIST Loss

VI. CONCLUSION

To conclude, this paper met its initial goal of providing a rudimentary analysis of quantum machine learning for binary classification. It set out to demonstrate some experiments with the technology using the MNIST dataset and generated some interesting results worth pondering. The project successfully explores the use of basis encoding, simulated quantum circuits and optimisation techniques in Python and Pennylane. There were a number of difficulties and considerations experienced during the project. Whilst this is a simple demonstration of quantum machine learning, it seems there is a lot more potential for all of the experiments if comparing between simple and complex data over more epochs. This would likely lead to far more definitive results for each technique and support the underlying theories behind them. Some further difficulties also included understanding the coding aspects from the perspective of quantum theory. This was a challenge during debugging, especially using modules in the beta phase of release where resources are restricted. Further to this, the computational power and time required to run some of the simulations are not to be underestimated. Some ideas for further exploration work begin with running the amplitude encoding and FRQI methods on a quantum computer rather than simulating the environment and its operations. It would be interesting to observe whether the results and convergence times are similar or drastically different. Another idea is to use FRQI for colour images and investigate its effectiveness against greyscale images since it is able to capture any amount of information given. Further investigating different variations of quantum circuits and employing some multi-wire measurements since these conditions are where the second ansatz in the paper is likely to perform best. The next natural steps in classification are to implement a multi-class classification model. Finally, there could be some merit to a quantum-classical combination model or some research into some benefits of this where the classical portion handles more broad issues that quantum might struggle with. On this topic, it would also be worth running these same experiments in a classical setting to understand where a quantum solution is better in this scenario.

[1] S. Brown, Machine learning, explained (2021).
[2] P. Johri, J. K. Verma, and S. Paul, Applications of machine learning, , 404 (2020).
[3] J. Ismael, Quantum mechanics, (2020).
[4] A. Dubey, Classical bit vs qubit (2020).
[5] P. Viswanath, Quantum states and the bloch sphere (2021).
[6] Z. Toffano and F. DUBOIS, Interpolating binary and multivalued logical quantum gates, Proceedings 2018, 2(4), 152; https://doi.org/10.3390/ecea-4-05006 **2**, 5006 (2017).
[7] Quantum Inspire, (2020).
[8] J. Emspak, Quantum entanglement: A simple explanation (2022).
[9] C. Orzel, How do you create quantum entanglement? (2017).
[10] Pennylane, (n.d.).
[11] J. McClean and H.-Y. Huang, Quantum machine learning and the power of data (2021).
[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE **86**, 2278–2324 (1998).
[13] Python.org, (2023).
[14] TensorFlow, (2023).
[15] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, and S. Ahmed, PennyLane: Automatic differentiation of hybrid quantum-classical computations https://doi.org/10.48550/arXiv.1811.04968 (2018).
[16] pandas 2.0.1 documentation, (2023).
[17] NumPy v1.24 Manual, (n.d.).
[18] Matplotlib 3.7.1 documentation, (n.d.).
[19] C. Lee, (2021).
[20] TensorFlow, (2023).
[21] M. Mottonen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa, Transformation of quantum states using uniformly controlled rotations (2004), arXiv:quant-ph/0407010 [quant-ph].
[22] D. Mattern, D. Martyniuk, H. Willems, F. Bergmann, and A. Paschke, Variational quanvolutional neural networks with enhanced image encoding (2021).
[23] M. Soeken, D. M. Miller, and R. Drechsler, Quantum circuits employing roots of the pauli matrices, Physical Review A **88**, 10.1103/physreva.88.042322 (2013).
[24] H. Collins and C. Nay, Ibm unveils 400 qubit-plus quantum processor and next-generation ibm quantum system two (2022).
[25] A. Iliyasu, F. yan, and Z. Jiang, Quantum computation-based image representation, processing operations and their applications, Entropy **16**, 5290 (2014).
[26] J. Kim, J. Kim, and D. Rosa, Universal effectiveness of high-depth circuits in variational eigenproblems, Physical Review Research **3** (2021).
[27] M. T. e. a. Maha A. Metawei, Hazem Said, Evaluation of different ansatze designs for quantum neural network binary classifiers, (2022).
[28] A. Ajagekar, Adam (2021).
[29] C. Lee, Adjoint differentiation - pennylane (2021).
[30] Python Numerical Methods, (n.d.).
[31] PennyLane, (n.d.).
[32] qml.GradientDescentOptimizer, (n.d.).
[33] PennyLane documentation, (n.d.).
[34] J. Gacon, C. Zoufal, G. Carleo, and S. Woerner, Simultaneous perturbation stochastic approximation of the quantum fisher information, Quantum **5**, 567 (2021).
[35] (2022).