An investigation into the trainability, capacity, and the generalization of quantum neural networks with IQP encoding using information geometry

Aydın Utting

10127947

School of Physics and Astronomy

The University of Manchester

MPhys Report

May 2021

This project was carried out in collaboration with *Mohammad Kordzanganeh*

Abstract

Quantum neural networks (QNNs) are a type quantum machine learning (QML) algorithm, which is an emergent field of research formed from the combination of machine learning and quantum computing. Abbas et al. in the paper "the power of neural networks" [1] show that some QNNs display better generalizability and trainability than classical machine learning models. This is measured through the effective dimension, a capacity measure for neural networks derived from information geometry using the Fisher information matrix. In this work we review these results and extend them by investigating the effect of increasing IQP encoding depth on QNN performance. We also investigate the efficacy of the effective dimension as a capacity measure and generalization bound by comparing the training performance of QNNs and classical neural networks to their effective dimensions. We find that the performance of QNNs does not improve as the encoding depth is increased, even though the effective dimension does increase. We also find that (in contrast to [1]) classical networks outperform the QNNs despite having much lower effective dimension. This leads us to conclude that the effective dimension is not an effective generalization bound.

Contents

1 Introduction

Quantum machine learning is an emergent field at the cross section between quantum computing and machine learning theory. Due to the high profile of both fields, quantum machine learning has had significant interest in recent years [2]. Quantum information theory gives a new paradigm for learning algorithms and this has been taken advantage of by many different approaches to quantum machine learning [3–5].

This work mainly focusses on hybrid quantum-classical algorithms where a quantum computer is only used for one part of the overall algorithm. A parametrised quantum circuit performs a calculation which is passed into a classical computer, which can optimise the parameters of the circuit to train the model.

These algorthims are usually used to analyse classical data. How data is encoded (or embedded) into a quantum state is an open problem, and may profoundly effect the performance of quantum algorithms [6]. In this work we discuss the IQP (Instantaneous Quantum Polynomial time) encoding scheme, first discussed in [7], which is designed to take advantage of the large feature space produced through qubit entanglement. While it is suggested that increasing the depth of the IQP encoding (the number of times the circuit is repeated) increasing the richness of the encoding, a full analysis of IQP encoding depth has not been carried out. In [1] it is suggested that more complex encoding increases the trainability and generalizability of QNNs, and gives them an advantage over classical neural networks. In this work we analyse this result by comparing QNNs of different IQP encoding depths and classical neural networks.

A key concept in machine learning is that of generalization: the ability for a model to perform well on data outside the training dataset. Statistical learning theory attempts to explain this by bounding the generalization error: the difference between the (unknown) performance of a model on the full set of all data, and the (known) performance of the model on the training subset [8, p. 447]. Generalization bounds often rely on a measure of the model complexity; the capacity of a model to fit functions. The first attempt to quantify the complexity of a model for generalization was suggested by Vapnik and Chervonenkis [9]. The measure they created was later named the VC dimension. Further work by Valiant developed the probably approximately correct (PAC) theory of learning [10]. These ideas were combined by Blumer et al, forming the basis of statistical learning theory [11]. In modern machine learning it has proved difficult to formulate and test complexity measures and generalization bounds [12].

In this report we use the effective dimension of the model space derived in [13] to compare the capacity of the models used. We also use the generalization bound based on effective dimension derived in [1] to compare the generalization error of the models. This effective dimension is based on the Fisher information matrix which can be shown to be the metric of model space [14] and quantifies the curvature of the loss landscape of the model. We analyse the effectiveness of this bound in Sections 6.4 and 7.2. We estimate the generalization of the models using a validation dataset and compare this to the results for effective dimension to test its ability to bound generalization. We find in practice that as effective dimension increases the generalization of the network decreases, which suggests the effective dimension is not capturing the generalizability of the models well.

The eigenspectra of the Fisher information matrix is also used to compare models. It is suggested in [1] non-degenerate, flat Fisher eigenspectra should lead to faster training as the loss landscape of the model is more favourable to gradient descent. We quantify the flatness of the Fisher eigenspectra using the entropy of the distribution of eigenvalues, and investigate the effect of adding encoding layers on the entropy. We then compare the entropy of the eigenspectra to the training performance of classical and quantum neural networks.

2 Machine Learning

In machine learning the input data $\mathcal{X}$ can be any digital input. All input data is associated with a label in the space $\mathcal{Y}$, which can be a discrete class label (e.g. $\mathcal{Y} = \{0, 1\}$) or continuous

variable (e.g. $\mathcal{Y} = \mathbb{R}^d$). The goal of machine learning is to produce a function $f : \mathcal{X} \to \mathcal{Y}$ which maps the input to it's corresponding label. For a given input x the output of the neural network is $f(x, \theta)$, where $\theta \in \Theta$ is a set of parameters belonging to the parameter space Θ. This is done by minimising a loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ which gives a value for how far the output $f(x, \theta)$ is from the associated label y. Machine learning algorithms are *trained* using a training data set $\mathcal{S}_n = \{(x_1, y_1), \cdots, (x_n, y_n)\}$ by optimising the average loss of all data points in the set

$$L(\theta) = \frac{1}{n} \sum_{i=1}^{n} l(f(x_i, \theta), y_i).$$

The loss function can be minimised in many ways. A common method is using gradient descent, where the parameters of the loss function are updated such that the function moves down its gradient with respect to the parameters. The ith parameter is updated by

$$\theta_i \to \theta_i' = \theta_i - \eta \frac{\partial L}{\partial \theta_i}$$

where η is known as the *learning rate*. The loss function is usually a measure of distance between two numbers, such as the mean squared error (MSE) $l(f, y) = \frac{1}{C} \sum_{c=1}^{C} (y_c - f_c)^2$, where C is the number of dimensions of the output f and the label y, and y_c is the cth component of y.

For classification the output of the neural network $f(x, \theta)$ is a vector, with each element associated with a possible class of the input. This vector can be transformed into a probability vector using the softmax function

$$p_c = \frac{e^{f_c}}{\sum_c e^{f_c}} \tag{1}$$

where p_c is the model's predicted probability of the output being in the cth class and f_c is the cth element of the output. The cross-entropy loss function is often used to compare predicted probability distribution to the true label vector y, given by

$$l(f, y) = -\sum_c y_c \log p_c. \tag{2}$$

In this work the classical neural networks we are concerned with are fully-connected feed-forward neural networks, where input vectors are multiplied by weight matrices to calculate the values at the next 'layer' of the network. Layers are usually separated by non-linear functions which restrict the values at each layer to a small range (normally [0,1],[-1,1], or [0,∞)) which stop overflow errors and make the layers of the network independent [15].

More details on the operation of neural networks can be found in my report from the first semester [16].

3 Quantum neural networks

Quantum neural networks are a subset of parametrised, or variational quantum circuits. Variational circuits allow for the preparation of arbitrary quantum states and are therefore useful for encoding data into qubits for a quantum computer, to encode Hamiltonians for quantum chemistry [17] and high energy physics [18]. If the parameters of the circuit are optimised to maximise or minimise some function, this is a quantum neural network.

The quantum neural networks (QNNs) used in this work are hybrid quantum-classical neural networks. They have two main components: the quantum parametrised circuit which calculates some function (or part of a larger function) and a classical computer which runs an optimisation algorithm on this function. The quantum circuit can further be broken down into three distinct parts: the data encoding circuit, the parametrised circuit, and the measurement circuit. These are shown in Figure 1 parts (a),(b),(c) respectively. In this section we will briefly introduce the key concepts of quantum computing before introducing the concepts of each part of the quantum circuit in turn, with a focus on the specific circuits used in this work.

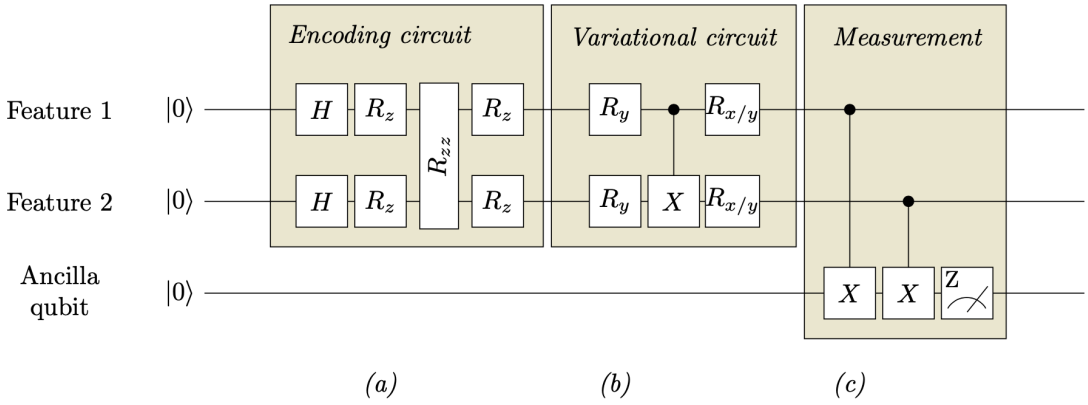


Figure 1: A circuit digram of a 2 qubit (plus 1 ancilla qubit) version of our Quantum neural networks showing (a) The encoding circuit (b) the variational circuit and (c) the parity circuit for measurement. Each horizontal line represent a qubit, and the objects on the line are gates acting on the qubits.

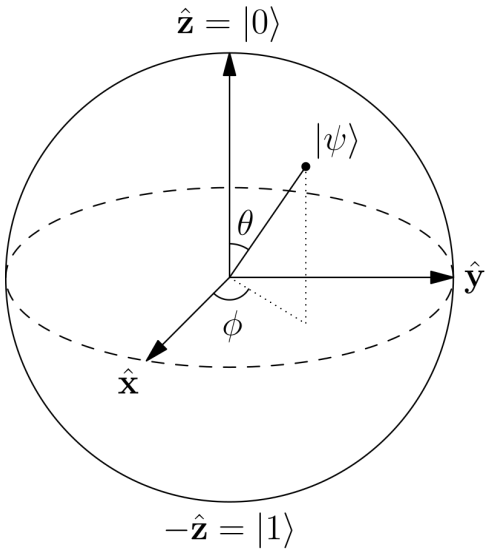


Figure 2: Bloch sphere representing the qubit $|\psi\rangle$ showing the azimuthal angle ϕ and the polar angle θ.

3.1 Introduction to quantum computing

Quantum computing is a field of research mainly focussed on using quantum information theory to complete calculations, simulations and other classically difficult computational problems. The fundamental unit of the quantum computer is the quantum bit or qubit. A qubit is a two-state quantum system, where the two orthogonal, basis states can be denoted (by analogy to classical computers) as $|0\rangle$ and $|1\rangle$. The qubit can exist in a superposition of the two states therefore its state can be written as

$$|\psi\rangle = a\,|0\rangle + b\,|1\rangle = \begin{pmatrix} a \\ b \end{pmatrix} \tag{3}$$

where a and b are complex numbers representing the quantum amplitudes of $|0\rangle$ and $|1\rangle$ respectively. Representing a qubit in this way is known as the computational basis. The coefficients a and b satisfy the condition that $|a|^2 + |b|^2 = 1$. This limitation allows the state of a qubit to be rewritten as

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + \sin\frac{\theta}{2}e^{i\phi/2}\,|1\rangle \tag{4}$$

and therefore the $|\psi\rangle$ can be represented in the Bloch sphere, as shown in Figure 2.

Operations on qubits take the form of unitary matrices. Any unitary matrix can be a single qubit gate, and all single qubit gates can be viewed as rotations in the Bloch sphere. Some examples of important single qubit gates (with their circuit diagram representations) are:

- The Pauli-X gate, X, which is equivalent to a π rotation around the x-axis of the Bloch sphere and analogous to the classical NOT gate as it reverses the coefficients of $|0\rangle$ and $|1\rangle$.

$$-\boxed{X}- \equiv \mathrm{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

- The Hadamard gate, H, which puts the qubit into an equal superposition of $|0\rangle$ and $|1\rangle$, with a phase shift depending on which basis it acts on: $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

$$-\boxed{H}- \equiv \mathrm{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

- The rotation gates RX, RY, and RZ which rotate the qubit some angle θ around the x,y, or z axis respectively. They are very imporant to quantum parametrised circuits, as they allow for arbitrary rotations.

$$\mathrm{RX}(\theta) = \begin{pmatrix} \cos\theta & -i\sin\theta \\ -i\sin\theta & \cos\theta \end{pmatrix}, \mathrm{RY}(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \mathrm{RZ}(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}$$

Adding additional qubits is mathematically equivalent to taking the tensor product of the two qubit states. For example if we have two states given by $|\mu\rangle = a|0\rangle + b|1\rangle$ and $|\nu\rangle = c|0\rangle + d|1\rangle$

$$\begin{aligned} |\psi\rangle &= |\mu\rangle \otimes |\nu\rangle && (5) \\ &= (a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle) && (6) \\ &= ac|0\rangle \otimes |0\rangle + ad|0\rangle \otimes |1\rangle + && (7) \\ &\quad bc|1\rangle \otimes |0\rangle + bd|1\rangle \otimes |1\rangle \\ &= ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle && (8) \end{aligned}$$

where between (7) and (8), we have use the shorthand convention for the basis states. The quantum amplitudes still satisfy $|ac|^2 + |ad|^2 + |bc|^2 + |bd|^2 = 1$. Therefore the two qubit state $|\psi\rangle$ has four orthogonal, basis states. For a state with N qubits, there are 2^N basis states. Logic gates acting on multiple qubits are given by the tensor product of the operators for example the 2-qubit Hadamard gates

$$\begin{matrix} -\boxed{H}- \\ -\boxed{H}- \end{matrix} = H^{\otimes 2} = H \otimes H = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}. \quad (9)$$

The most important 2-qubit gate is the Controlled-NOT (CNOT) gate. A CNOT gate applies a Pauli-X gate to a qubit if a second qubit is in the state, and does nothing if the second qubit is $|0\rangle$. Any multi-qubit quantum logic gate can be created with CNOT and single-qubit gates [19].

$$\equiv \mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (10)$$

In this work an important set of gates will be the 2-qubit rotation gate RZZ. This is a symmetric (independent of the order of the qubits) gate which rotates and entangles two qubits with respect to an angle. It can be constructed from two CNOTs and an RZ gate as

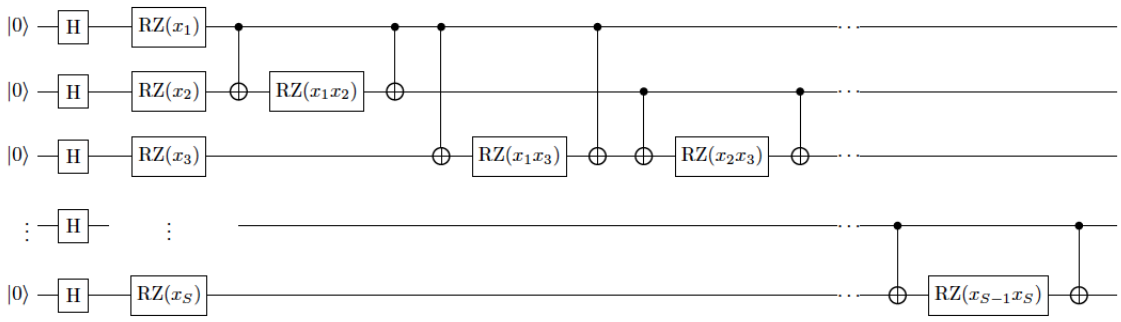
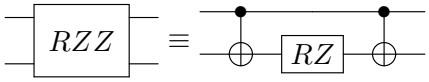


Figure 3: Figure of 1 layer of the IQP encoding scheme for S qubits [1]. Increasing the circuit depth refers to repeating everything after the Hadamard gates.



In order to extract the results from a quantum computation, the state of the qubit or qubits is measured with respect to some basis. When the quantum state is measured it collapses into the eigenstate measured. Quantum algorithms must therefore be designed such that useful information can still be extracted. Quantum measurements often taken the form of an expectation value of some physical observable. For example if we measure want to find the expectation value of an observable B of a quantum state $|\psi\rangle$, we measure $\langle\psi| B |\psi\rangle$. In practice this involves rotating the state such that measuring in the computational basis is equivalent to measuring in the B basis, and repeating this for a large number of calculations (or 'shots') to get an empirical value representing $\langle\psi| B |\psi\rangle$.

3.2 Encoding data into quantum states

Machine learning algorithms are mostly concerned with fitting complex function to some input data, such as splitting two classes of data or giving a numerical value given some input features. In quantum machine learning there are many ways in which the classical information can be stored in a quantum state known as encoding schemes. The method used is highly dependent on the task and many methods have been discussed in literature [6, 20]. Most encoding schemes attempt to produce an advantage over classical algorithms using superposition and entanglement.

A simple encoding scheme is basis encoding, which represents a binary string with the equivalent computational basis state, i.e. $0101 \rightarrow |0101\rangle$. This allows for parallel computation of functions on multiple bit strings simultaneously, and is used in fundamental quantum algorithms such as the Deutsch-Josza algorithm and Grover search [19].

Another possible encoding scheme is amplitude encoding, where a continuous classical value is encoded into the quantum amplitude for a given basis state. This means that a 2^n-dimensional vector can be encoded using only n qubits.

In this work we investigate the popular IQP (Instantaneous Quantum Polynomial time) encoding scheme, first proposed in [7] to take advantage of the high-dimensional feature space produced by quantum states. A diagram of the encoding scheme is presented in Figure 3. Repeating this circuit multiple time is referred to as increasing the depth of the encoding. Encoding depths ≥ 2 are suggested to be difficult to classically simulate. By entangling the qubits using the RZZ gates the data is encoded into the full 2^n-dimensional Hilbert space produced by n qubits. We also use the terminology encoding depth 0 to refer an encoding scheme of only the Hadamard and first RZ gates (referred to as easy quantum in [1]).

Previous works have shown that increasing the encoding depth of the model can lead to an increase in the generalizability of the network [1] through more complex encoding. We investigate this in Section 6. Also it becomes more difficult to implement quantum circuits as they become deeper (as the number gates in series increases), so finding the optimal or minimum encoding depth is useful when applying quantum machine learning to a problem.

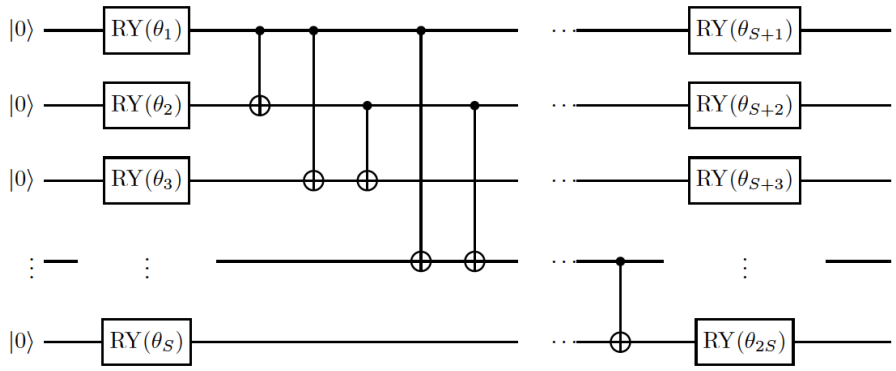


Figure 4: Structure of the RYRY variational circuit with $D = 1$. The CNOT gates connect all the qubits together exactly once, with the upper qubit as the control qubit. The final layer of RY can be replaced with RX to get the RYRX network structure. [1]

3.3 Variational circuits

The two variational circuits used in this work are shown in Figure 4. They are designed to be hardware-efficient (easy to use in practice on real quantum computers), while also giving high expressibility [20]. The first S parameters are encoded by rotating the qubits with the first set of RY gates, before all the qubits are entangled using all-to-all CNOTs, and finally the second set of parameters is encoded into the states by a second set of rotation gates, RY or RX. This structure can be repeated to increase the number of trainable parameters. The number of trainable parameters d is calculated as $d = (D+1)S$ where D is the number of times the all-to-all CNOT gates and second set of RY/RX gates are repeated, and S is the number of qubits. In Section 6 we compare the performance of the two possible structures. We refer to variational circuits ending in RY gates as RYRY circuits, and those ending in RX gates as RYRX.

3.4 Parity measurement circuit

In binary classification tasks where the desired output of the circuit is the probabilities of the input belonging to either class 0 or class 1, the post-processing-parity circuit is commonly used [1].

The parity function outputs 1 if an odd number of qubits are in the $|1\rangle$ state and 0 if an even number of qubits are in the $|1\rangle$ state. The circuit to implement this is shown in Figure 1c. The ancilla qubit is flipped each time a calculation qubit is in the $|1\rangle$ state. Therefore in the simple case of all calculation qubits being in either $|0\rangle$ or $|1\rangle$ it is simple to see that the ancilla qubit will be in $|0\rangle$ if there are an even number of $|1\rangle$ qubits. In most cases the calculation qubit will be in a superposition, and this leads to a complex state in the ancilla qubit. The qubit is measured in the computational basis by getting the expectation value of the Pauli-Z operator: $\langle\psi_{\text{ancil}}| Z |\psi_{\text{ancil}}\rangle$. This returns the -1 if the state is $|1\rangle$ and 1 if the qubit is in $|0\rangle$.

The probability for getting a parity of 0 can be calculated from the expectation value using the equation

$$p(0) = \frac{\langle\psi_{\text{ancil}}| Z |\psi_{\text{ancil}}\rangle + 1}{2} \tag{11}$$

and $p(1)$ can be calculated simply from $1 - p(0)$. These values can be used in loss function such as cross entropy.

3.5 Calculating gradients in quantum neural networks

In neural networks the gradient is calculated through backpropagation, where the chain rule is applied successively to give the partial derivative with respect to each parameter. The partial derivatives with respect to the parameters of the quantum layers must be calculated as part of backpropagation through the hybrid network. Although this can be implemented analytically on

simulated quantum computers it is impossible to perform on real quantum hardware. Therefore to calculate quantum gradients the parameter shift rule is used.

The parameter shift rule Starting with a function $f = \langle 0| U^\dagger(\theta) B U(\theta) |0\rangle$,where B is a physical observable and $U(\theta)$ is a variational circuit, we require the gradient $\partial_\mu f$ where μ is a single parameter that parametrises the unitary gate $\mathcal{G}(\mu)$. The circuit U can be broken down into $U = V\mathcal{G}W$ and therefore f can be rewritten

$$f = \langle \psi| \mathcal{G}^\dagger Q \mathcal{G} |\psi\rangle \tag{12}$$

where $|\psi\rangle = W|0\rangle$, and $Q = V^\dagger B V$. It can be shown then that the gradient of f with respect to the parameter μ is given by

$$\partial_\mu f = r(\langle \psi| \mathcal{G}^\dagger(\mu + s) Q \mathcal{G}(\mu + s) |\psi\rangle - \langle \psi| \mathcal{G}^\dagger(\mu - s) Q \mathcal{G}(\mu - s) |\psi\rangle \tag{13}$$

where where $\pm r$ is the eigenvalue of the Hermitian operator G that generates the unitary operator $\mathcal{G}(\mu) = e^{-i\mu G}$, and $s = \frac{\pi}{4r}$. Therefore the exact gradient of a function with respect to any single-qubit gate can be calculated using two evaluations of the function at $\mu \pm s$ [21]. (Although it should be noted that these are still expectation values, and therefore tend to the exact solution when averaging over many evaluations.)

4 Generalization bounds

As described above, machine learning model gives a function f which is parametrised by θ. The set of all possible functions that a given network structure can produce by changing its parameters is $\mathcal{H}$. Each hypothesis $h \in \mathcal{H}$ is a possible version of f with a different set of parameters. The input-label pairs are drawn from some unknown distribution $p(x, y)$ which the algorithm attempts to approximate.

To test the quality of a hypothesis at predicting p one can define the risk $R(h) = \mathbb{E}_{p(x,y)}[L(h(x), y)]$. However it is often only possible to access a small sample of input-label pairs. The training set $\mathcal{S}_n$ can be used to train a model, finding the hypothesis with the lowest empirical loss $R_n(h) = \frac{1}{n}\sum_{i=1}^{n} L(h(x_i), y_i)$. The goal of machine learning is to choose a hypothesis which performs well on data outside the training set $\mathcal{S}_n$. This is known as generalization. As such the difference in performance of a model on the training set compared to its performance on all possible data drawn from p is known as the generalization error and given by

$$\sup_{h \in \mathcal{H}} |R(h) - R_n(h)| \tag{14}$$

which should tend to 0 as $n \to \infty$. How this is calculated is still an open problem in machine learning theory [22]. Many approaches attempt to bound this quantity using a bound based on a measure of complexity. The complexity (or flexibility or capacity) of a model is a measure of how complex a function the model can fit [9]. Most bounds take the form

$$P\left(\sup_{h \in \mathcal{H}} |R(h) - R_n(h)| > \epsilon(\delta, C)\right) \geq 1 - \delta \tag{15}$$

where δ is a confidence measure, C is the complexity measure, ϵ is a function of confidence and complexity [8, p.447-454].

A commonly employed method to measure the generalizability of a model is to use a validation dataset. This is a set of samples drawn from p which are not in the training set. During training the validation loss is calculated using this set. The difference between the training and validation loss can indicate how well the model is generalizing, with a smaller difference suggesting better generalization. We employ this method in Section 6.4 to assess the generalization of our models.

5 Information Geometry

Information geometry is a field studying how geometry and topology can be applied to information theory and statistics. Central to this study is the Fisher information matrix, or Fisher metric, first discussed in this context by Rao [14]. The Fisher metric allows for the calculation of infinitesimal distances on a *statistical manifold*: a space of statistical models which are parametrised by different sets of parameters θ. The effective dimension, which we use to measure the capacity of machine learning models, is based on the Fisher metric. Therefore here we will discuss the derivation of the Fisher metric from the Kullback-Lieber divergence.

5.1 Fisher information

P and Q are points at θ and θ' respectively on a surface which is parametrised by the set of coordinates $\{\theta_i\}$. $D[P|Q] = D[\theta|\theta']$ is a valid divergence measure on the manifold if it has the following properties

1. $D[\theta|\theta'] \geq 0, \forall \theta, \theta'$. i.e. it is positive definite.
2. $D[\theta|\theta'] = 0$ only for $\theta = \theta'$. The divergence between a point and itself is 0.

If we take P and Q to be infinitesimally close together such that $P = \theta$ and $Q = \theta + \mathrm{d}\theta$, we can Taylor expand D around θ to obtain

$$D[P|Q] \approx D[P|P] + \mathrm{d}\theta_i \frac{\partial}{\partial \theta_i} D[P|Q]\bigg|_{P=Q} + \frac{1}{2}\mathrm{d}\theta_i \mathrm{d}\theta_j \frac{\partial^2}{\partial \theta_i \partial \theta_j} D[P|Q]\bigg|_{P=Q} + O(|\mathrm{d}\theta|^3) \tag{16}$$

where we have used Einstein summation notation for clarity. The first term is zero due to the first property of D above, and the second is zero due to the second. Therefore we are left with

$$D[P|Q] \approx \frac{1}{2}\mathrm{d}\theta_i \mathrm{d}\theta_j \frac{\partial^2}{\partial \theta_i \partial \theta_j} D[P|Q]\bigg|_{P=Q} + O(|\mathrm{d}\theta|^3). \tag{17}$$

This has units of distance squared, and therefore we can define an infinitesimal distance ds^2 as

$$\mathrm{ds}^2 = 2D[P|Q] = \mathrm{d}\theta_i \mathrm{d}\theta_j \frac{\partial^2}{\partial \theta_i \partial \theta_j} D[P|Q]\bigg|_{P=Q} = \mathrm{d}\theta_i \mathrm{d}\theta_j g_{ij} \tag{18}$$

for P, Q infinitesimally separated [23]. In the last equality we have defined the metric of the space g_{ij} as the matrix of second derivatives (the Hessian) of the divergence D. If the metric can be defined to be positive definite the manifold is Riemannian as one can define an inner product. At every point in a Riemannian metric one can define a tangent space.

A statistical manifold is a surface made up of distributions at each point, with coordinates θ. For instance a joint distribution of vectors x and y with parameters θ is given by $P(x, y; \theta)$. On the manifold the divergence between two probability distributions is given by the Kullback-Lieber divergence

$$D_{KL}[P(x, y; \theta)|Q(x, y; \theta')] = \iint \mathrm{d}x \mathrm{d}y P(x, y; \theta) \log\left(\frac{P(x, y; \theta)}{Q(x, y; \theta')}\right). \tag{19}$$

Using Equation 18 one can differentiate D_{KL} twice at the point $P = Q$ and obtain the Fisher metric

$$F_{ij}(\theta) = \partial_i \partial_j D[\theta|\theta']\big|_{\theta=\theta'} = \mathbb{E}_{P(x,y;\theta)}[\partial_i \log P(x, y; \theta) \partial_j \log P(x, y; \theta)] \tag{20}$$

where $\partial_i = \frac{\partial}{\partial \theta^i}$ and $\mathbb{E}$ is the expectation value. So the Fisher information matrix can be used as the metric of the statistical manifold [24].

The calculation of the Fisher information matrix in Equation 20 uses the full joint distribution $P(x, y; \theta)$ and cannot be calculated exactly with finite data. In order to calculate the Fisher

matrix we must use an empirical formulation. Therefore in this report the Fisher matrix is calculated as

$$F_k(\theta) = \frac{1}{k}\sum_{i=1}^{k} \nabla_\theta \log p(y_i, x_i; \theta) \nabla_\theta \log p(y_i, x_i; \theta)^T \tag{21}$$

This approximation is often referred to as empirical Fisher and is used throughout machine learning literature [25]. To calculate $F_k(\theta)$ the vector of gradients with respect to all the parameters is calculated for each data point in the training set. The outer product is taken between this vector and itself, and the resulting square matrix is summed for all data points and divided to find the average. In Section 7.1 we evaluate the accuracy of this approximation to the Fisher matrix.

We also use the Fisher matrix to evaluate the trainability of the neural networks using their eigenspectra in Section 6.2. The distribution of eigenvalues of a matrix is known as its spectrum or eigenspectrum. High degeneracy in eigenvalues of the Fisher matrix, with a few high-value outliers has been shown to decrease the speed at which a model trains [26]. Quantum neural networks have also been shown to have flatter eigenspectra, with less degeneracy than classical neural networks [1]. To quantify the flatness of the eigenspectra, we plot the eigenvalues in a histogram with 10 bins, and calculate the entropy of the distribution. The entropy of the distribution is given by

$$S = -\sum_{i=0}^{10} h_i \log h_i \tag{22}$$

where h_i is the height of the ith bin, which are normalised such that $\sum_i h_i = 1$. The range of the histogram was left unbounded to penalise the entropy if there were high outlying eigenvalues.

5.2 Effective Dimension

The effective-dimension-based generalization bound discussed here was developed in [1], based on the effective dimension complexity measure first discussed in [13].

The effective dimension is derived from the box counting dimension of the model space $\mathcal{M}$. This space is Reimannian with the Fisher information matrix as its metric F_{ij} and has dimensionality equal to the number of trainable parameters in the model d. The volume element of such a space is given by $\sqrt{\det F(\theta)}$. Therefore the effective dimension of a model can be calculated using

$$d_{\text{eff},\gamma} = 2\frac{\log\left(\frac{1}{V_\Theta}\int_\Theta \mathrm{d}\theta \sqrt{\mathrm{Id}_d + \frac{\gamma n}{2\pi\log(n)}\hat{F}(\theta)}\right)}{\log\left(\frac{\gamma n}{2\pi\log(n)}\right)} \tag{23}$$

where n is the number of data points, $V_\Theta = \int_\Theta \mathrm{d}\theta$ is the volume of the parameter space Θ, γ is a constant used in the derivation of the generalization bound which is set to 1 for the remainder of the report, Id_d is the d-dimensional identity matrix, and $\hat{F}$ is the normalised Fisher matrix calculated as

$$\hat{F}(\theta) = d\frac{V_\Theta}{\int_\Theta \mathrm{d}\theta' \ \mathrm{Tr}(F(\theta'))}F(\theta) \tag{24}$$

such that the on average the Fisher matrices satisfy $\mathrm{Tr}(F(\theta)) = d$. The number of data points n gives a 'resolution' at which the model space can be observed, and as such as $n \to \infty$, $d_{\text{eff}} \to d$.

It has been shown in [1] that this can be used to derive a generalization bound which depends on the effective dimension. Therefore by comparing values of the effective dimension we can compare the relative generalizability the models, as calculated by this measure. However, we will later show that the generalization of the model is not well captured by this bound.

In order to calculate integrals over the parameter space Θ the integrals are replaced by Monte Carlo integration using random samples of parameters. The number of samples was chosen to be 100 in all cases to balance accurate values for effective dimension with time and computing constraints when running the experiments. The uncertainty in the effective dimension is discussed in Section 6.3.

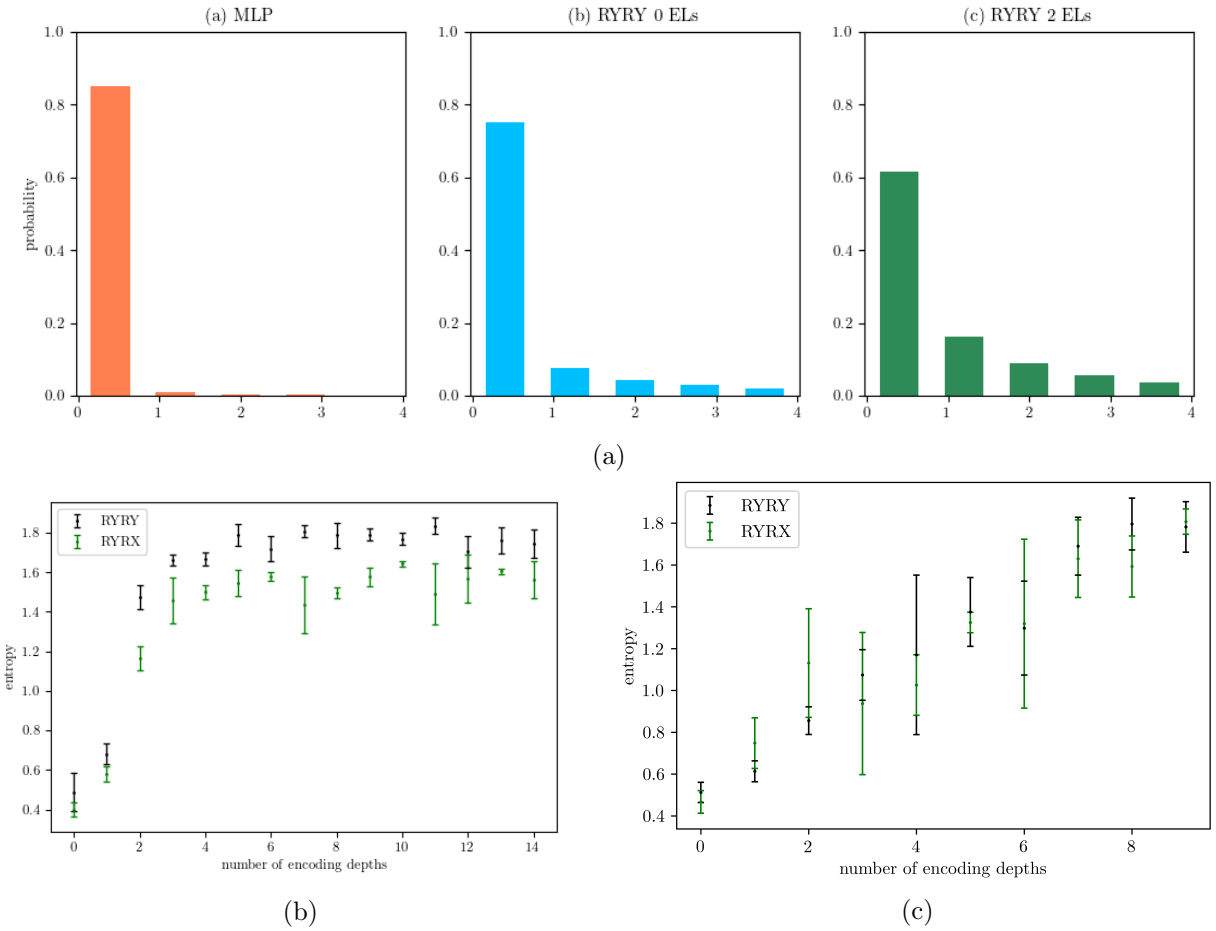


Figure 5: (a) Histograms showing the eigenspectra for the classical neural network and the quantum neural network with encoding depths of 0 and 2, with x axis constrained to the range [0,4] for clarity. (b) Plot of the entropy of the eigenspectra against IQP encoding depth for Gaussian clouds and (c) for IRIS.

6 Numerical Experiments

6.1 Implementation

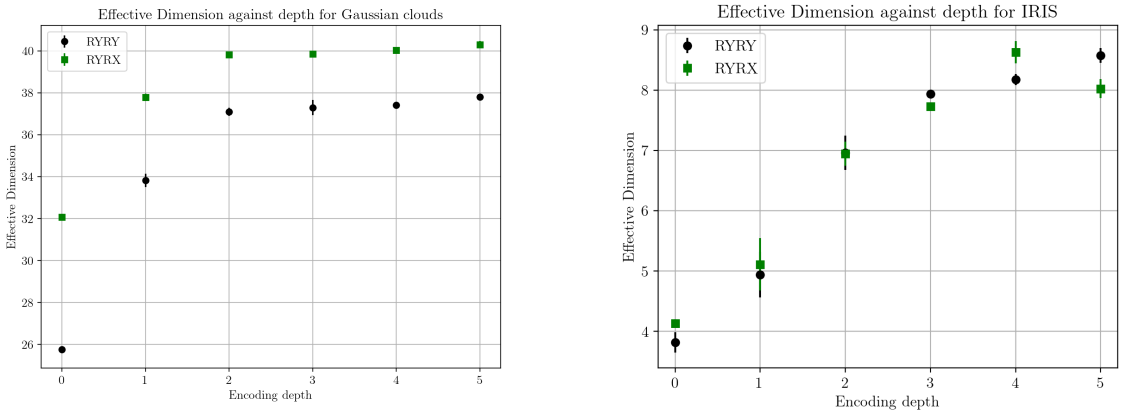
The following experiments were implemented in Python using the Pennylane library for quantum machine learning [27] and the PyTorch library for classical machine learning [28]. The quantum circuits were run using the Pennylane simulator.

For training two datasets where used, which we refer to as the 'Gaussian clouds' dataset and the IRIS dataset. The Gaussian clouds were produced using two 4-dimensional unit Gaussians centred on $(1, -1, 1, -1)$ for class 0 and $(-1, 1, -1, 1)$ for class 1. As the data is produced at runtime it can be of arbitrary size. The IRIS dataset consists 3 types of iris plant (Iris Setosa, Iris Versicolour, and Iris Virginica), each with 50 data points [29, 30]. Each data point consists of 4 features. For our analysis it is we only use the first two of the 3 classes to be consistent with previous work in [1].

6.2 Fisher eigenspectra

The normalised Fisher matrices for a classical network and quantum neural networks with $d = 40$ trainable parameters were calculated at 1000 points in parameter space using the Gaussian clouds dataset. All 40,000 eigenvalues for each were plotted in histograms as shown in Figure 5a.

In agreement with [1], the eigenvalues of the classical neural network are largely degenerate at 0 with a few large eigenvalue outliers, and the quantum neural networks are not degenerate. As the depth of the quantum neural network increases the eigenspectra becomes more evenly spread. This can be quantified in terms of the entropy of the histogram with a fixed number



(a) Gaussian clouds

(b) Iris

Figure 6: Plot showing effective dimension of quantum neural networks at different depths of IQP encoding for (a) Gaussian clouds and (b) IRIS.

of bins. The entropy against encoding depth for the quantum neural networks using IRIS and Gaussian clouds is shown in Figure 5. For Gaussian clouds the entropy increases rapidly from depth 0 to 3, and is slower to increase from then on for both RYRY and RYRX. RYRY has greater entropy for all depths. For IRIS the entropy increases linearly for RYRY and RYRX and does not flatten out.

6.3 Depth and effective dimension

The effective dimension for classical and quantum neural networks was calculated as described in Equation 23 for the IRIS dataset and the Gaussian clouds dataset. Figure 6 shows how the effective dimension increases with depth for both datasets used. In agreement with [1], increasing encoding depth from 0 to 2 increased the effective dimension. For the Gaussian clouds RYRX has a higher effective dimension than RYRY for all depths, suggesting that RYRX should have greater capacity for generalization. For IRIS RYRX is initially slightly higher, but at depth 2 they are indistinguishable. RYRY peaks at depth 4 and RYRX peaks at depth 5. Both datasets show effective dimensions exceeding d for high depths, which is not possible when the effective dimension is calculated analytically, but can occur in our empirical approximations. We discuss this further in Section 7.2.

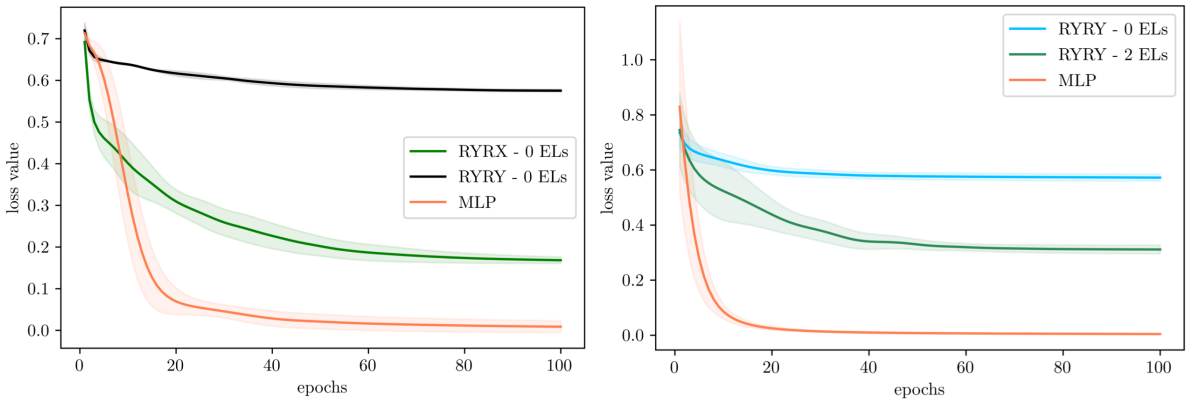
For both datasets the classical neural network had a significantly lower effective dimension. For Gaussian it was calculted to be 7.1 ± 0.1, while for IRIS it was calculated to be 2.83 ± 0.03.

The uncertainty of the effective dimension is shown in Figure 6. This is the error due to initialisations of the weights. The average error is around 3% on average which supports our use of 100 samples of parameters for the integrals in Equation 23.

6.4 Training and depth

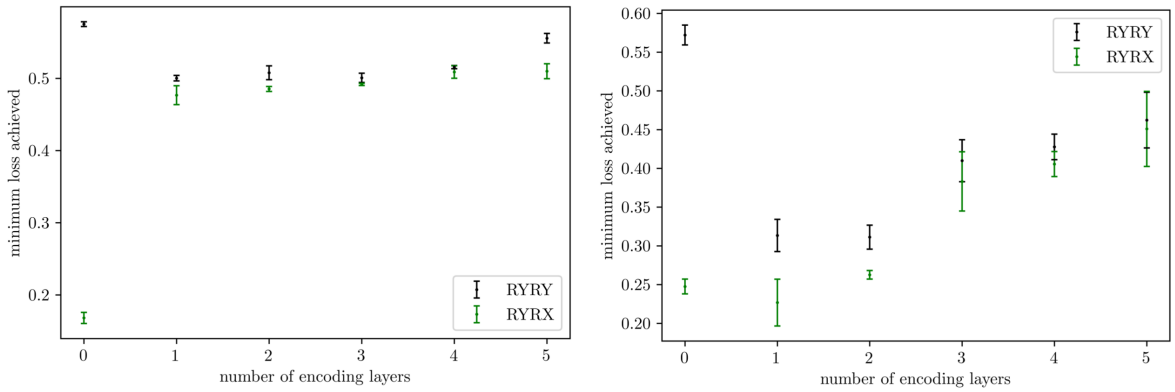
In order to test the effect of depth on the trainability of the quantum neural network, classical and quantum neural networks with were trained on Gaussian clouds and IRIS for 100 epochs. The ADAM optimizer was used with a learning rate of 0.1, and 100 data points were used for both. Key loss curves are shown in Figure 7. The classical network performed significantly better than all quantum models. This could be because of the data being highly classical, and simple to separate with straight lines in some dimensions. This is contrary to the results in [1] and the differences in classical network structure which lead to this difference are discussed in Section 7.4.

The minimum loss did not decrease with number of encoding layers and therefore did not correlate with entropy or effective dimension. The minimum loss achieved against depth is shown in Figure 8. This is contrary to classical results in [31] and suggestions made in [1], where the



(a) Gaussian clouds (b) Iris

Figure 7: Training loss curves on (a) Gaussian clouds and (b) IRIS. In (a) the QNNs with highest and lowest loss are compared to the classical network, and in (b) RYRY depth 0 and 2 are compared against classical to compare with [1].



(a) Gaussian clouds (b) Iris

Figure 8: The minimum loss achieved during training against encoding depth for (a) Gaussian clouds and (b) IRIS. The shaded region represents 1 standard deviation of 5 runs.

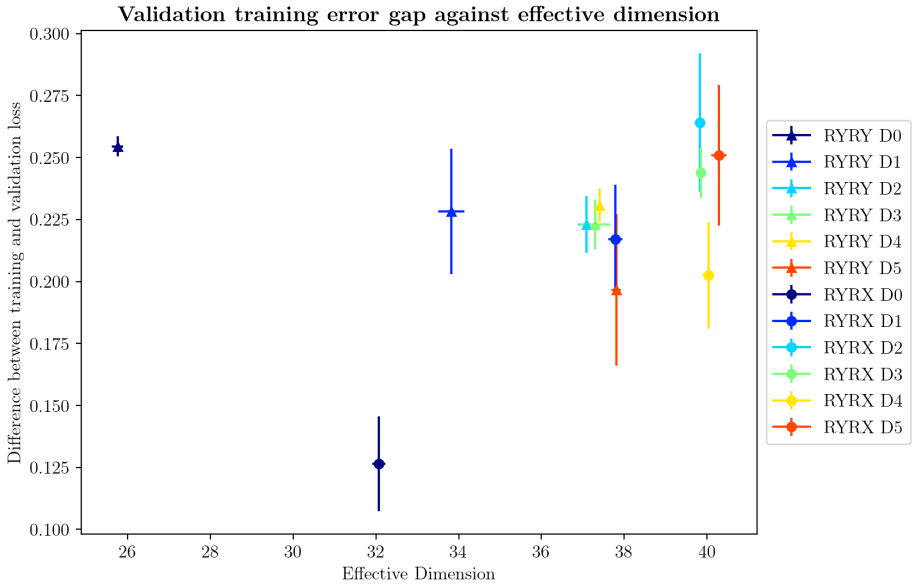


Figure 9: Plot of the gap between training and validation error at the final epoch against effective dimension for Gaussian clouds. The classical network was not included for visibility, as would have been at the coordinates $(7.1 \pm 0.1, 0.13 \pm 0.06)$, having significantly lower effective dimension.

increasing flatness of the Fisher spectrum is shown to correlate with increased trainability. This could suggest that the best encoding depth for a given classification tasks is particular to the dataset used, rather than increasing with effective dimension These results suggest that the effective dimension is not a good measure of trainability, as is suggested in [1].

Using a validation subset of the two datasets, we can estimate the generalizability of the models. A plot of the difference between validation and training loss (generalization gap) against effective dimension for Gaussian clouds can be seen in Figure 9. There is a positive correlation between the generalization gap and effective dimension, with a Pearson correlation coefficient of 0.59. This suggests that the effective dimension may not be a good measure of generalizability, as we would expect the gap to decrease with effective dimension.

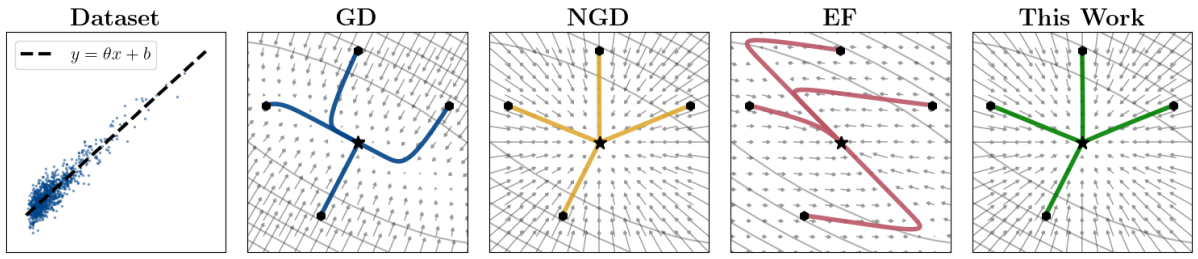
The outliers on the plot, the depth 0 networks and the classical networks, significantly effected the correlation. Without the classical network the Pearson correlation coefficient becomes 0.16, showing almost no correlation between the generalization gap and the effective dimension. Removing the depth 0 networks and considering only the full IQP encoded circuits, the coefficient is 0.29 which suggests a weak positive correlation. Further analysis of more network structures is needed to gain a full picture of the relationship between the generalization gap and the effective dimension.

7 Critical evaluation

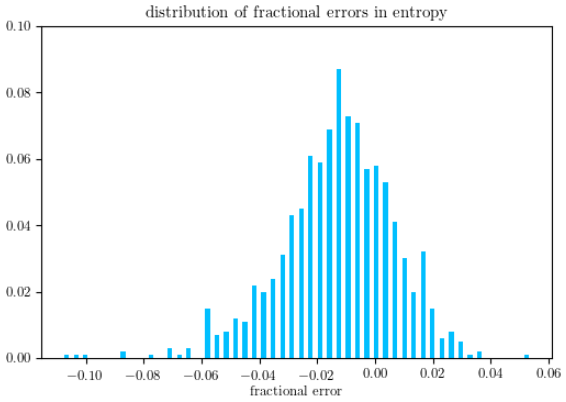
7.1 Approximation of Fisher

In order to ensure that our calculation of the empirical Fisher matrix was correctly approximating the Hessian of the loss, we build on work in [25]. A comparison of results can be seen in Figure 10a. The final plot of the vector field for our approximation matches closely with the NGD result, which supports our approach as a very good approximation to the true Fisher.

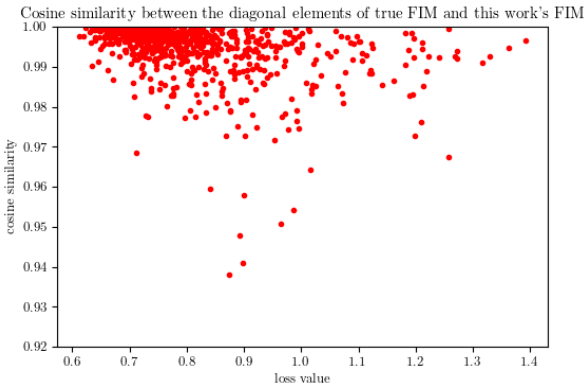
To further analyse the error in our approximation for the Fisher we calculated the entropy of the eigenspectra for the Gaussian clouds with our approximation and the true Fisher calculated in [25, Equation 35]. The fractional difference between the two calculations are plotted in Figure 10b. The mean error is around 1% reflecting that our approximation is very close to the true Fisher.



(a)



(b)



(c)

Figure 10: (a) Figure adapted from [25] showing the result of a gradient descent algorithm using (from left to right) the gradient descent, natural gradient descent, empirical Fisher matrix, and the Fisher calculated in this work. The arrows show the direction of the gradient at that point, and the contours display values of equal loss. The star shows the optimum loss value.(b) Histogram showing the difference between the entropy of the true Fisher spectra and our approximation. (c) Cosine similarity against loss value between the true Fisher and our approximation for a random sampling of parameter space, cosine similarity close to 1 indicated that the matrices are very similar.

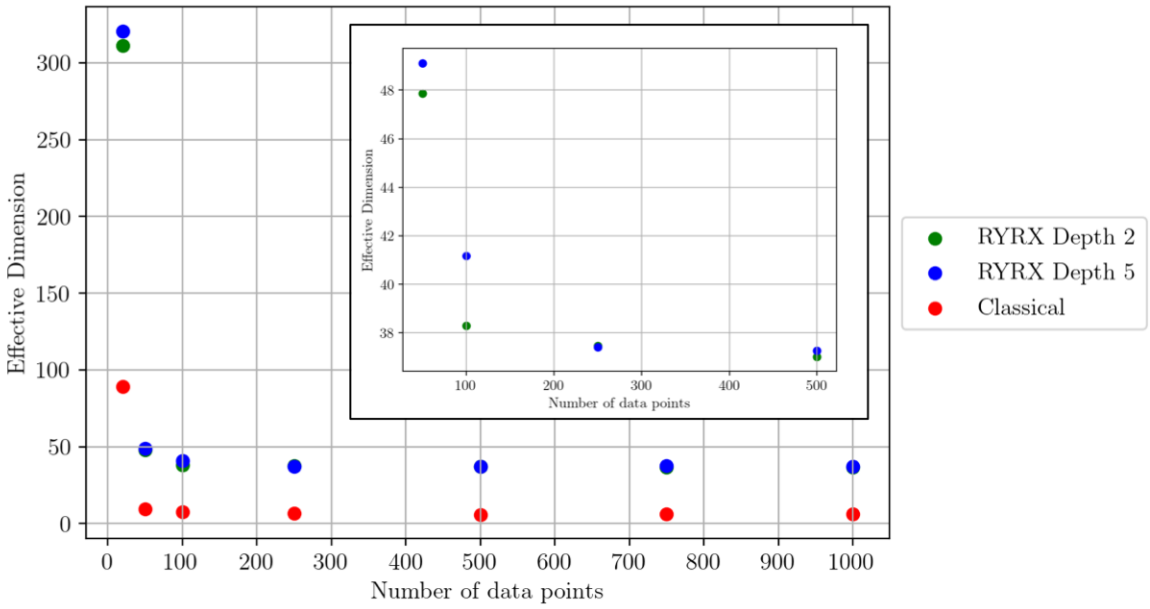


Figure 11: Plot of effective dimension against number of data points for RYRX depths 2 and 5, and the classical network. The number of data points decreases rapidly from 0 to 100 data points. The inset shows a closer view of the QNNs from 50 to 500 data points, and it is clear that at 100 data points the effective dimension has not reached a stable value.

Further comparison between the true Fisher and our approximation was done using the cosine similarity, with a method adapted from [32]. Figure 10c shows that the cosine similarity between the two matrices is close to 1 for a wide range of loss values. This agrees well with the previous two results, and shows the accuracy of the approximation we have used.

7.2 Effect of number of data points on the effective dimension

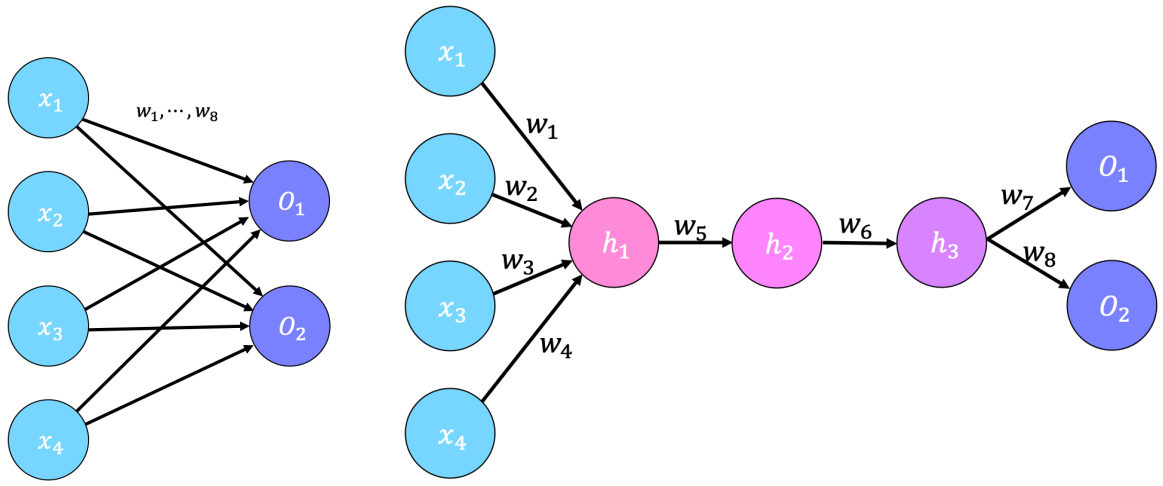
The number of data points used to calculate the Fisher matrix greatly effects the value of effective dimension, due to the Fisher information matrix being approximated through Monte Carlo integration and the $\frac{n}{\log n}$ terms in the effective dimension.

As shown in [1] the effective dimension should approach the maximal rank of the Fisher matrix as the number of data points goes to infinity. For all the quantum neural networks, the rank of the Fisher matrix is equal to the number of parameters d, as there is no degeneracy in the eigenvalues. The number of data points used for calculating the Fisher also greatly increased the time taken for the calculations. Therefore a balance was needed to allow for a wide range of depths to be tested.

In practice we found that the effective dimension was too large, $d_{\text{eff}} > d$, for some encoding depths. Figure 11 shows the effect of increasing number of data points on the effective dimension for Gaussian clouds. For number of data points < 100 the effective dimension is significantly larger than $d = 40$. This suggests that the effective dimension method is less applicable to small datasets. The effective dimension begins to tend to a single value for number of data points $>$100. However at 100 data points the effective dimension is still clearly changing significantly, as shown in the inset. The IRIS dataset has only 100 data points. Therefore Figure 11 suggests that the IRIS dataset is too small for accurate estimation of the effective dimension.

7.3 Analytical calculations of increasing encoding depth

In order to see more clearly the effect of adding IQP encoding layers, we calculated analytically the state of the qubit after m layers for a 2 qubit algorithm. The resulting state is



(a) (b)

Figure 12: Representations of both possible $d = 8$ network structures. (a) 4-2 structure used in this work (b) 4-1-1-1-2 structure used in [1]. x_i represent the 4 input values, w_i represent the 8 weights of the network, h_i are the nodes in hidden layers and O_i are the output nodes

$$\hat{U}^m(x_1, x_2)\,|00\rangle = \frac{1}{2}\exp\left(\frac{im}{2}\begin{bmatrix} -\left(x_1x_2 - (\pi+1)(x_1+x_2) + \pi^2\right) \\ \left(x_1x_2 - \pi(x_1+x_2) + (x_2-x_1) + \pi^2\right) \\ \left(x_1x_2 - \pi(x_1+x_2) + (x_1-x_2) + \pi^2\right) \\ \left(x_1x_2 + \pi(x_1+x_2) + (x_1+x_2) - \pi^2\right) \end{bmatrix}\right),$$

where x_1 and x_2 are the two inputs, and $m > 0$. It is clear that the number of encoding layers simply increases a multiplicative factor in the state. When the state is measured this will become equivalent to increasing the frequency of a sine function by a multiple m. The linearity of the quantum circuits means that the variational circuit can only adjust the amplitudes of these sine functions. This suggests that increasing the circuit depth of the encoding only improves performance when the data happens to be best suited to a given frequency. On the other hand, the effective dimension and entropy increase with encoding depth. Further research is needed to analyse why this occurs.

Trainable encoding schemes, suggested by Schuld et al. [6] have been shown to produce a sum of Fourier terms, and can access more frequencies with increased circuit depth. Further analysis is needed to see how these methods compare to the IQP encoding used in this work.

7.4 Classical network structure

In order to compare the results for the quantum neural networks to those in the literature, a classical network was tested alongside the quantum neural networks. Two classical neural networks were used: the $d = 40$ network used for the Fisher eigenspectra and the $d = 8$ network for training on IRIS.

In [1] when comparing the Fisher eigenspectra of classical and quantum neural networks it was proposed that the matrix with the highest rank would lead to the highest effective dimension and least degenerate Fisher eigenspectra. Therefore all possible $d = 40$ classical feed-forward neural networks were tested by calculating the rank of the Fisher spectrum. The network with the highest rank was used.

The $d = 8$ network had fewer possible network structures. Both possible structures are represented in Figure 12. In [1] the 4-1-1-1-2 network shown in Figure 12b is used for IRIS training. This performs significantly worse that the 4-2 network used in this work. The single node hidden layers significantly reduce performance of neural networks as they collapse the information from the previous layers into 1 node. The use of the 4-1-1-1-2 model handicapped the classical neural network in [1], and lead to the conclusion that the effective dimension and

degenerate eigenspectra lead to worse trainability. However in this work we have shown that the 4-2 classical network performs significantly better than the quantum neural networks which have higher effective dimension and less degenerate Fisher eigenvalues, which suggests the effective dimension or entropy are not good measures of trainability.

8 Conclusion

In this report we have discussed the trainability and generalizability of quantum neural networks with IQP encoding and compared them to classical feedforward networks. In Section 6.2 we used the entropy to show that QNN's had flatter Fisher eigenspectra than classical neural networks and that they are flatter when additional IQP encoding layers are added. Additionally, classical neural networks had very degenerate Fisher matrices, while quantum neural networks had no degeneracy. Although previous works [26] have suggested that this should lead to faster training, in Section 6.4 we found that the classical networks trained to a significantly lower loss than QNNs and that adding IQP encoding layers did not lead to faster training. We also compared the networks' generalizability using the effective dimension as a measure of capacity. We found that the QNNs had higher effective dimension than the classical networks for all encoding depths. Adding encoding layers increases the effective dimension for IRIS and Gaussian clouds. However by estimating the generalization error using a validation data set, we showed that the effective dimension was positively correlated to the generalization error. This suggests that the effective dimension is not capturing the generalizability of the models. We then discussed quality of our empirical approximations for the Fisher matrix and the effective dimension. In Section 7.1 we showed that the Fisher was a good approximation to the true Fisher, and in Section 7.2 we discussed the effect of the number of data points on the effective dimension. We concluded that the effective dimension method is not accurate for small data sets, including IRIS. In Section 7.3 we analytically calculated the effect of IQP encoding in a two qubit network, and showed that the effect of adding addition IQP encoding layers is to increase a multiplicative factor. We concluded that trainability is better for some networks because the frequency of the function after encoding better fit the data. Further investigation is needed to investigate why the effective dimension increases in spite of this. Finally in Section 7.4 we compared our classical network structure to the structure used in [1] and suggested that single-node hidden layers significantly handicapped training performance, leading to the conclusion that classical networks performed worse than QNNs in [1].

There are many avenues for further research from this work. Additional work is need to use QNNs with complex datasets that not specifically made for testing machine learning models. Further analysis of the effective dimension generalization bound is need to test its efficacy, as we have shown it does not capture the generalizability of QNNs.

References

[1] Amira Abbas, David Sutter, Christa Zoufal, Aurélien Lucchi, Alessio Figalli, and Stefan Woerner. The power of quantum neural networks. *arXiv:2011.00027 [quant-ph]*, October 2020.

[2] M. Schuld, I. Sinayskiy, and F. Petruccione. An introduction to quantum machine learning. *arXiv:1409.3097 [quant-ph]*, September 2014.

[3] Iris Cong, Soonwon Choi, and Mikhail D. Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, December 2019.

[4] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum Circuit Learning. *Physical Review A*, 98(3):032309, September 2018.

[5] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. Quantum Generative Adversarial Networks for Learning and Loading Random Distributions. *npj Quantum Information*, 5(1):103, December 2019.

[6] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. The effect of data encoding on the expressive power of variational quantum machine learning models. *Physical Review A*, 103(3):032430, March 2021.

[7] Vojtěch Havlíček, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, March 2019.

[8] Claude Sammut and Geoffrey I. Webb, editors. *Encyclopedia of Machine Learning*. Springer, New York ; London, 2010.

[9] V. N. Vapnik and A. Ya. Chervonenkis. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, January 1971.

[10] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[11] Anselm Blumer, A. Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, October 1989.

[12] Guillermo Valle-Pérez and Ard A. Louis. Generalization bounds for deep learning. *arXiv:2012.04115 [cs, stat]*, December 2020.

[13] Oksana Berezniuk, Alessio Figalli, Raffaele Ghigliazza, and Kharen Musaelian. A scale-dependent notion of effective dimension. *arXiv:2001.10872 [cs, stat]*, January 2020.

[14] C. Radhakrishna Rao. Information and the Accuracy Attainable in the Estimation of Statistical Parameters. In Samuel Kotz and Norman L. Johnson, editors, *Breakthroughs in Statistics: Foundations and Basic Theory*, Springer Series in Statistics, pages 235–247. Springer, New York, NY, 1992.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[16] Aydin Utting. *Understanding Aleatoric Uncertainties in Neural Networks*. Masters, University of Manchester, January 2021.

[17] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1):4213, July 2014.

[18] Samuel Yen-Chi Chen, Tzu-Chieh Wei, Chao Zhang, Haiwang Yu, and Shinjae Yoo. Hybrid Quantum-Classical Graph Convolutional Network. *arXiv:2101.06189 [hep-ex, physics:physics, physics:quant-ph]*, January 2021.

[19] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge ; New York, 10th anniversary ed edition, 2010.

[20] 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, 2019.

[21] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *arXiv:1811.11184 [quant-ph]*, November 2018.

[22] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring Generalization in Deep Learning. *arXiv:1706.08947 [cs]*, July 2017.

[23] Shun'ichi Amari. *Information Geometry and Its Applications*. Number volume 194 in Applied Mathematical Sciences. Springer, Japan, 2016.

[24] Shun'ichi Amari. *Differential-Geometrical Methods in Statistics*. Springer New York, New York, 1985.

[25] Frederik Kunstner, Lukas Balles, and Philipp Hennig. Limitations of the Empirical Fisher Approximation for Natural Gradient Descent. *arXiv:1905.12558 [cs, stat]*, June 2020.

[26] Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient BackProp. In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade*, volume 7700, pages 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[27] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, M. Sohaib Alam, Shahnawaz Ahmed, Juan Miguel Arrazola, Carsten Blank, Alain Delgado, Soran Jahangiri, Keri McKiernan, Johannes Jakob Meyer, Zeyue Niu, Antal Száva, and Nathan Killoran. PennyLane: Automatic differentiation of hybrid quantum-classical computations. *arXiv:1811.04968 [physics, physics:quant-ph]*, February 2020.

[28] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. October 2017.

[29] R. A. Fisher. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[30] Edgar Anderson. The Species Problem in Iris. *Annals of the Missouri Botanical Garden*, 23(3):457–509, 1936.

[31] Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. Universal Statistics of Fisher Information in Deep Neural Networks: Mean Field Approach. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1032–1041. PMLR, April 2019.

[32] Valentin Thomas, Fabian Pedregosa, Bart van Merriënboer, Pierre-Antoine Mangazol, Yoshua Bengio, and Nicolas Le Roux. Information matrices and generalization. June 2019.

Acknowledgements

A huge thank you to our supervisor Prof Anna Scaife, who's guidance and support have made this year incredibly rewarding, interesting and enjoyable. I'd like to thank my lab partner Mohammad, for his boundless enthusiasm and drive, and his support and friendship over the last year. Also thanks to Matthew Bromley and Max Allen, for lending their ideas and listening to ours in our weekly meetings. Thanks to A. Abbas for answering our questions regarding her work.