Impact of Barren Plateaus Countermeasures on the Quantum Neural Network Capacity to Learn

Jacob L. Cybulski[1*] and Thanh Nguyen[1,2†]

[1*]School of IT, Deakin University, Burwood Highway, Burwood, 3125, Victoria, Australia.
[2]Institute of Research and Development, Duy Tan University, Da Nang city, 550000, Vietnam.

*Corresponding author(s). E-mail(s): jacob.cybulski@deakin.edu.au;
Contributing authors: nguyencongthanh@duytan.edu.vn;
†These authors contributed equally to this work.

Abstract

Training of Quantum Neural Networks can be affected by *barren plateaus* - flat areas in the landscape of the cost function, which impede the model optimisation. While there exist methods of dealing with barren plateaus, they could reduce the model's *effective dimension* - the measure of its capacity to learn. This paper therefore reports an investigation of four barren plateaus countermeasures, i.e. restricting the model's circuit depth and relying on the local cost function; layer-by-layer circuit pre-training; relying on the circuit block structure to support its initialisation; as well as, model creation without any constraints. Several experiments were conducted to analyse the impact of each countermeasure on the model training, its subsequent ability to generalise and its effective dimension. The results reveal which of the approaches enhances or impedes the quantum model's capacity to learn, which gives more predictable learning outcomes, and which is more sensitive to training data. Finally, the paper provides some recommendations on how to utilise the effective dimension measurements to assist quantum model development.

Keywords: Quantum Computing, Quantum Information, Quantum Neural Networks, Variational Quantum Algorithms, Barren Plateaus, Effective Dimension

Version 3.0 (21 November 2023): This version of the article has been accepted for publication, after peer review but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. Use of this Accepted Version is subject to the publisher's Accepted Manuscript terms of use https://www.springernature.com/gp/open-research/policies/accepted-manuscript-terms. The Version of Record of this article will be available online at Quantum Information Processing

1 Introduction

Quantum Neural Network (QNN) is a machine learning model that takes advantage of the quantum theory to represent and implement the selected properties of classical artificial neural networks, which could then be deployed for execution on quantum devices [1]. Most commonly, QNN employs a parameterised quantum circuit, to define the structure of a neural network, and a hybrid quantum-classical approach, known as Variational Quantum Algorithms (VQAs), to optimise its parameters [2, ch5]. While quantum techniques can be used to implement different types of neural networks, Multi-Layer Perceptron (MLP) is the closest classical analogue of QNNs, including their layered architecture, as well as, gradient-based and stochastic methods of their optimisation [3, sect 2.5, 5, 7.1].

As VQA is the mainstream method for designing QNNs, QNN inevitably inherits some of the VQA shortcomings, e.g. training large QNNs may suffer from *barren plateaus* (BP) [4, 5], which are large flat areas in the cost landscape of the VQA parameter space, impeding effective optimisation of the QNN circuit. The problem is similar to the *vanishing gradient* phenomenon in classical deep neural networks [6].

This research aims to survey and compare a number of well-known countermeasures to BP phenomena in VQA-based QNN development. In particular, the project focuses on the effectiveness of QNNs treated with those countermeasures, and specifically their capacity to learn, which can be measured as the model *effective dimension* [7, 8], representing a class of functions fitting and generalising training data.

2 Barren Plateaus in QNN Training

2.1 Quantum Neural Networks

When developing a QNN model, we aim to implement a class of functions $f_\theta(x) \to y$ as a variational quantum circuit $U(x, \theta)$, parameterised with θ and taking input data x. When the circuit is executed repeatedly, assuming the initial quantum state $|0\rangle$, it could then generate expectation values of its measurements $\mathcal{M}$, which can be interpreted as output y [2, ch5].

In a typical VQA process of QNN development, the parameters θ are being optimised classically, so that the circuit $U(x, \theta)$ could best approximate a specific function $f_\theta(x) \to y$ defined by the sample of input-output pairs (x_t, y_t), which are given as the model training data. The process is assisted by the cost function measuring the distance between the calculated results and their expected values, and guided by an optimiser able to iteratively vary the parameters θ to gradually reduce the cost.

The QNN considered in this paper is a variational circuit of three blocks (Figure 1), which are responsible for its input, processing and output [9]:

- *feature map*: a circuit $S(x)$ encoding a training example x of classical data into the circuit quantum states;
- *ansatz*: a layered circuit $W(\theta)$ tasked with the processing of quantum information, the accuracy of which is determined by its parameters θ; and,
- *measurement*: a circuit measuring the final quantum state, representing the quantum observable $\mathcal{M}$.

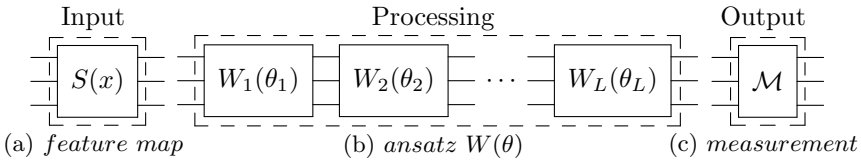


Figure 1: QNN architecture, which includes three blocks: a *feature map* $S(x)$, *ansatz* $W(\theta)$ of L layers $W_l(\theta_l)$ parameterised by $\theta = \{\theta_l\}$, and the *measurement* block $\mathcal{M}$.

The variational quantum circuit can thus be formulated as consisting of a feature map and an ansatz:

$$U(x, \theta) = W(\theta)S(x) \ , \tag{1}$$

and QNN defined as measurement of the state $|\psi(x, \theta)\rangle$ prepared by $U(x, \theta)|0\rangle$, or an equivalent projective measurement in the computational basis of this variational circuit, hence estimating the function $f_\theta(x)$ [2, ch5]:

$$f_\theta(x) = \langle\psi(x, \theta)|\mathcal{M}|\psi(x, \theta)\rangle \ , \tag{2}$$

$$f_\theta(x) = \mathrm{Tr}\big[\mathcal{M}U(x, \theta)^\dagger|0\rangle\langle 0|U(x, \theta)\big] \ . \tag{3}$$

Note that the feature map is constructed for the specific example of training data - this means that QNN is a dynamic learning model, of which parameter values, and potentially its structure, may vary during training. Also, the strategies selected for data encoding by the feature map, quantum information processing by the ansatz, as well as decoding of the resulting quantum states, must all be consistent.

The ansatz $W(\theta)$ is an arbitrary parameterised circuit, however, most commonly it is expressed as a product of L layers:

$$W(\theta) = W(\theta_1, \cdots, \theta_L) = \prod_{l=1}^{L} W_l(\theta_l) \ , \tag{4}$$

with the parameterised operator $W_l(\theta_l)$ defined as:

$$W_l(\theta_l) = e^{-i\theta_l H_l} V_l \ , \tag{5}$$

where the unitary V_l and hermitian H_l operators are unparameterised, and θ_l being the l-th element of the set of parameters $\theta = \{\theta_l\}$. Parameters θ_l define properties of individual ansatz operator gates (such as qubit state rotations), with their values being manipulated by the optimiser to minimise the overall cost function [9].

The *cost function* receives trainable parameters θ and training data $\{(x_t, y_t)\}$. It uses a function $g_{x,y}(m)$ calculating distances between the estimates of the $f_\theta(x_t)$, as obtained from the circuit $U(x, \theta)$, and the expected values y_t. The total cost can thus be expressed as a sum of all distances calculated for the training examples.

$$C(\theta) = \sum_{(x,y)} g_{x,y} \left(\mathrm{Tr}\left[\mathcal{M} U(x,\theta)^{\dagger} |0\rangle\langle 0| U(x,\theta) \right] \right) \ . \tag{6}$$

The cost function can be calculated by relying on either a quantum or classical algorithm, however, the latter is the most common and most convenient way of its calculation in the VQA context.

The *optimiser* is the classical component of the VQA hybrid method. The VQA process is iterative and is commonly controlled by the optimiser itself. It starts with a given training data and an initial set of parameter values. Parameter initialisation may involve a random selection of values, or it may be a separate process, which could be complex and rely on an in-depth knowledge of the problem domain (such as molecular structures in chemistry).

At each optimiser iteration, VQA instantiates the variational quantum circuit $U(x, \theta)$ by encoding input x_t of the selected training example (x_t, y_t) into the circuit's feature map $S(x_t)$, and setting the current values of trainable parameters θ_t for the ansatz $W(\theta_t)$. This results in a static, unparameterised circuit $U(x_t, \theta_t)$, which can be executed on a quantum machine or its simulator. By running such a circuit multiple times, the optimiser generates a distribution of possible observable results, which can then be used to estimate the cost function value $C(\theta_t)$. At each iteration, the estimated cost is relied upon by the optimiser to improve the circuit parameter values. By changing the circuit parameters, the optimiser explores the cost landscape in search of the global minimum, while avoiding any local minima, and thus progressing towards the optimum values for the circuit parameters. This process is repeated until an optimum set of parameter values is found (within an acceptable tolerance) or the process is aborted due to its non-convergence [9].

The optimisation strategy has a significant impact on VQA's accuracy and training performance. While there exist many different classes of optimisation algorithms [10], gradient optimisation is one of the most commonly used. Analysis of gradient is especially important in detecting the presence of barren plateaus, where gradients are near zero but away from the optimum, and so gradient optimisers were of special interest to this project. The method of gradient descent relies on moving the parameters progressively, at some learning rate, in the direction of the steepest slope within the cost landscape, which should lead the optimiser towards the minimum cost, and thus the problem solution. To avoid converging into a local minimum, stochastic gradient descent (SGD) is recommended [11]. A class of adaptive Adam (Adaptive Moment Estimation) stochastic optimisers use both gradients and second momentum information, which improves the speed of algorithm convergence [12].

Nadam (Nesterov-accelerated Adaptive Moment Estimation) [13], a well performing variant of Adam algorithm, was adopted in this project. Nadam replaces a standard momentum calculation with Nesterov-accelerated gradient (NAG) estimation, to further increase training performance, even with gradients of high curvature and some noise.

2.2 Barren Plateaus and Their Mitigation Methods

There exist several well-researched methods of dealing with the emergence of BPs in quantum model development. They can all be grouped into four distinct focus areas, which are explained as follows.

Focus on ansatz size

An approach which aims to control a circuit depth and the number of qubits in the ansatz variational structure, and hence the number of trainable parameters.

McClean et al. [4] conducted a study of large QNNs with the commonly used cost functions and randomly initialised parameters. Subsequently, they found out that increasing the model size (as defined by the number of its circuit parameters, layers and qubits) in the presence of randomly initialised circuit parameters, may lead to statistically independent circuit components generating similar outputs despite of differing parameter values (2-design). This phenomenon was found to be responsible for the flattening of the cost landscape, BP emergence, and hence, ineffective model training.

They also revealed that in randomly initialised variational circuits of L layers, the gradient growth rate is $O(n^{1/L})$, which implies that for large L cost gradients are mostly constant - in this case approximating zero - for the great majority of QNN parameters θ, thus creating large BPs. They noted that variance of the cost gradient shrinks exponentially with the number of qubits n (and the circuit size in general).

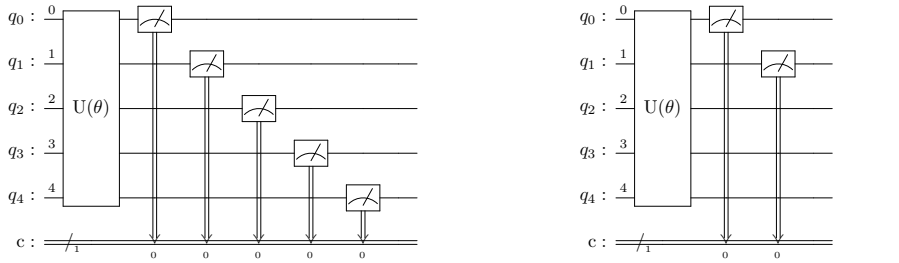
The problem is further amplified when training QNNs on NISQ machines, where it may not be possible to distinguish small gradients from hardware noise [14] or when noise alone is able to induce BPs in VQA-based circuit training [15]. This is a serious issue for the gradient-based optimisation of QNNs in the presence of a near-flat cost landscape.

It is hence commonly recommended to develop a shallow QNN ansatz [16]. An alternative approach is to relax the circuit depth but instead interweave the process of ansatz construction and optimisation. In this way, at each optimisation step it is possible to deal with a shallow ansatz sub-circuit and a small number of trainable parameters [14, 17], thus avoiding BP formation. We expect the latter method, which manipulates more parameters over the entire training process, to facilitate learning of more complex functions.

Focus on observable and cost

An approach which targets the selection of circuit observables responsible for measurements used by the cost function.

It is worth noting that BP could emerge not only in large QNNs but also in shallow circuits, but with an inadequately chosen observable and a cost function [16] (refer to Figure 2). For example, circuit optimisation that relies on a *global cost function*, of which observable aims to measure the state of all qubits and is thus assessing the circuit state in exponentially large Hilbert space, is more prone to the emergence of barren plateaus. Whereas, the use of a *local cost function*, of which observable assesses the states of the selected few qubits, reduces the complexity of the state space, and is less likely to lead to BP development.



(a) *Global Cost Function* measurements (b) *Local Cost Function* measurements

Figure 2: Configuration of measurements in different classes of cost functions

Experimenting with a local cost function and different ansatz architectures could lead to a better control over gradients in training, with the potential for BP avoidance [18]. For example, Cerezo et al. [16] recommended using the local cost with a depth-limited ansatz to ensure non-zero variance of gradient in circuits training.

Having said this, circuit training with a global cost function tends to produce more robust quantum models. In such models, as compared with models trained with a local cost function, all parameters participate equally in training, resulting in parameter changes distributed across the entire circuit. However, this is subject to the successful avoidance of BPs in the process. Therefore, circuit training can be further improved by using a combination of local and global cost functions, the former to avoid BPs and the latter to gain the model quality. This can be accomplished by gradually tuning the circuit and its observable in a training cycle with the series of cost functions of increasing locality, i.e. gradually involving more qubits in measurement [19].

Focus on ansatz parameters initialisation

An approach which seeks to refine the starting point for the optimisation of circuit parameters.

For the vast majority of QNN designs, random parameter initialisation is a default strategy, which as mentioned before in some circumstances may lead to sub-optimal results [4]. Therefore some innovative circuit initialisation methods have been proposed, to pre-train the ansatz while reducing the depth of trained sub-circuits and the scope of random parameter initialisation, if any. Grant et al. [17], for example, suggested to construct an ansatz of shallow circuit blocks, each with two parts - one with random parameter values and another being its inverse, so that each block implements an identity operation (see Figure 3). Thus obtained parameter values serve as the initial point for the second phase of the circuit training. This approach allows avoiding the BP areas in the initial steps of the circuit optimisation.

Another method was proposed by Skolik et al. [14] who also favoured a two-phased training algorithm. In the first phase of layerwise learning, the process starts by training a single layer ansatz with all its parameters initialised to zero. Once optimised, the parameters of the trained layer are frozen, and the ansatz extended with a new

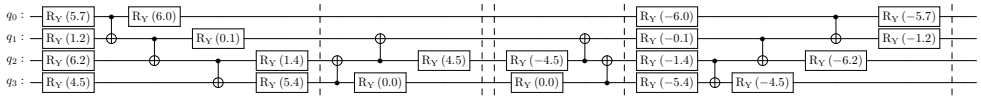


Figure 3: A sample *identity block*, featuring exactly 20 parameters over 4 qubits - the first part consisting of 10 rotational gates with random parameters and the second part the inverse of the first part.

layer of untrained parameters, again initialised to zero and ready for the next step of optimisation. At each step, the algorithm deals with a small extension of the circuit and a small number of untrained parameters, thus reducing the likelihood of developing BPs in the optimisation process. In the second phase, the algorithm retrains the entire multi-layer ansatz, initialised with the previously found non-random parameter values, until convergence. The approach is claimed to minimise the chance of entering BPs.

The idea behind these algorithms is to set the initial ansatz parameters away from BPs so that the optimiser could avoid zero gradients in the early stages of the circuit training. However, these methods cannot guarantee the cost function to run into BPs in further optimisation steps.

Focus on the current practice

A commonly practised approach which accepts the creation of ansatze of arbitrary depth, with random parameter initialisation, and an observable to support a global cost function.

We have explored four distinct focus areas of dealing with BPs. However, any practical BP mitigation strategy needs to address all four foci. Therefore, such a strategy should provide details of the circuit construction or modification. It ought to recommend ways of measuring such a circuit and suggest a suitable cost function. A method of initialising the circuit parameters should also be known in advance. Finally, to warrant our investigation, a BP countermeasure needs to be well-known in the quantum computing community.

We, thus, identified three highly-cited academic publications, each describing some pro-active BP mitigation method [14, 16, 17], and each emphasising one or two of the previously discussed focus areas (see Table 1). This way, we have proposed method #0 as commonly practised, yet passive approach, with no specific BP mitigation strategy and unpredictable results. A pro-active method #1 aimed to eliminate BPs by shortening the QNN ansatz and using a local cost function. An equally purposeful methods #2 and #3 focused on the initialisation of ansatz parameters to avoid BP areas, by relying on layerwise learning or identity blocks, respectively. We designed the last two methods to utilise deep circuits and global cost functions, which was in contrast with method #1, and to focus them on their areas of strength.

A preliminary investigation of the four selected methods was carried out to determine the predisposition of their models to forming BPs in training.

Method / Aspect	**#0: Generic - No BP strategy**	**#1: Shallow circuit with Local cost**	**#2: Layerwise learning**	**#3: Identity blocks**
Ansatz depth	Any	Shallow	Any	Any
Cost Function	Global	Local	Global	Global
Initial Params	Randomised	Randomised	Restricted	Restricted
Reported BP Effect	Unpredictable	Eliminated [16]	Avoided [14]	Avoided [17]

Table 1: BP mitigation strategies investigated in this project

This was achieved by studying geometry of QNN models constructed in accordance with the methods' recommendations and repeated random parameterisation to define their gradient space. Analysis of variance in the gradient space reveals the potential presence of BPs in the landscape of the cost functions, with small variance indicating the possibility of BPs and large variance suggesting their likely absence. By increasing the complexity of the investigated models, e.g. by changing the ansatz depth or the number of qubits, which are known to promote the emergence of BPs, it is possible to determine the model resilience to BP formation.

The preliminary analysis of randomly generated gradients for the four BP mitigation strategies (see Figure 4), indicated that method #1 (shallow circuits with the local cost function) retains high variance across increasingly more complex ansatze, and hence, has the best chance of avoiding development of BPs in QNN training. While the remaining, although more sophisticated, methods #0, #2 and #3 featured significant gradient decay with the number of qubits used in their ansatze.

And yet, in many published cases, similarly to the above-mentioned preliminary investigation of the four BP mitigation strategies, the methods are selected without any consideration of training data, which restricts the range of allowable parameter values, and thus creates a gradient sub-space, with its own unique features, which could enhance or impede the strategy of choice.

Therefore, it is important to examine BP mitigation strategies in QNN training, to determine their suitability for the model architecture (as defined by its ansatz, observable and cost function) and training data, as well as the impact of the selected strategy on the QNN capacity to learn.

2.3 Measurement of QNN Learning Capacity

There are many conceptualisations of the neural network "capacity". For example, the concept of "storage capacity", as measured in bits and bytes of information, dates back to the early investigations of neural networks' associative memory [20, 21]. Other studies considered the network potential for storing input-output patterns [22, 23]. With the advent of larger and more complex neural networks, the focus shifted to practical issues of estimating the network's requisite complexity, measured as the number of layers of densely interconnected "neurons", necessary to approximate a

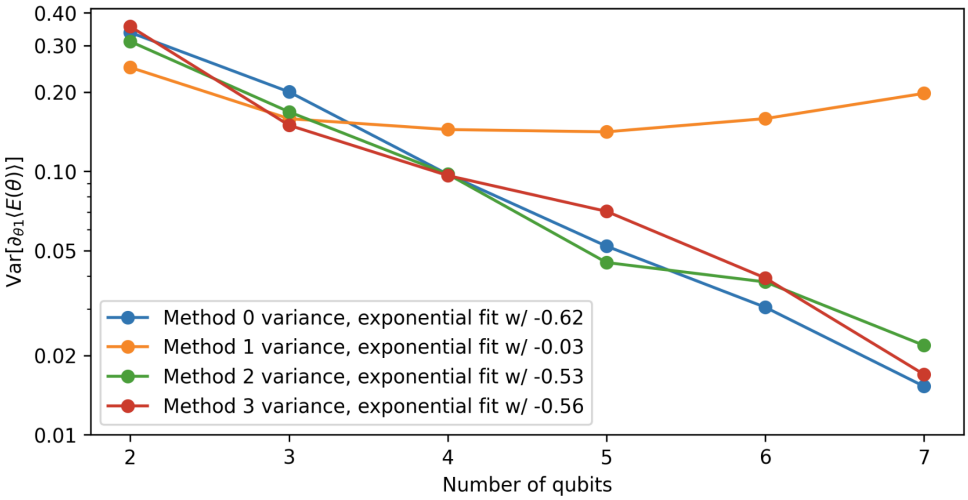


Figure 4: Analysis of gradient for the four BP mitigation strategies (y log scale)

given function (as implied by training data). In a similar vein, others focused on removing unnecessary network parameters without adversely affecting its information contents [24].

Vapnik, Levin and LeCun [25, 26] defined the notion of "VC-dimension" (from Vapnik-Chervonenkis) which is an "empirical measurement of the maximal difference between the error rates on two separate data sets of varying sizes". Such a measure could be interpreted as the set of functions the neural network could represent depending on the size of the training set and the error rate tolerance. In a sense, VC-dimension describes not the network storage or its complexity, but rather a dynamic notion of its "capacity to learn". Baum and Haussler further extended Vapnik and Chervonenkis' VC-dimension to study the network ability to learn and generalise from its training data set [27]. However, the initial formulation of the VC-dimension, which required to "exactly" quantify the lower and upper bounds of measurements, was difficult to calculate for many classes of functions. Hence the authors later replaced their notion of "dimension" with the concept "effective VC-dimension", the new measure of neural network capacity, which took into account not only the size of the training data and the expected error rates, but also the probability distribution of its measurements [25].

A significantly different take on the network capacity to learn was given by Karakida et al. [28] who adopted the "Fisher Information Matrix" (FIM) as a statistical tool linking the neural network learning with the shape of the cost gradient emerging in the network's optimisation. In a similar way, Liang et al. [29] proposed a Fisher-Rao metric as a measure of generalisation and complexity of neural nets.

The notion of model capacity naturally translates to quantum models of neural networks, e.g. Lewenstein et al. [30] are concerned with QNN storage capacity, which is measured by the number of linearily independent pure states the network can represent, and Abbas et al. [7, 8] propose QNN's "effective dimension", which combines "effective VC dimension" with "Fisher Information Matrix" to estimate the number

of functions that a quantum model can learn. Larocca et al [31] also explored the concept of QNN effective quantum dimension and observed that the model learning capacity, which can be enlarged by increasing the number of its parameters, is nevertheless capped by the model's effective dimension. As noted by some scholars [32], a quantum model effective dimension expresses a complex relationship between geometry of the model parameter space, its expressive power, training ability, degree of parameter redundancy and the effectiveness of its initialisation strategy.

To study the impact of BP strategies on QNN learning, we adopted the Abbas et al. [7, 8] concept of *effective dimension* as the most versatile measure of the quantum model's capacity to learn.

2.3.1 Fisher Information

The key tool of the adopted measure of a QNN learning capacity - the effective quantum dimension - relies on the calculation of the Fisher Information Matrix (FIM). In general, Fisher information [33] plays an important role in analysing statistical models. Furthermore, applications of Fisher information include construction of confidence intervals and hypothesis tests, definition of priors for model parameters and measuring model complexity.

A neural network can be considered as a probabilistic model for the conditional distribution $p(x, y; \theta) = p(y|x; \theta)p(x)$ that receives the parameters $\theta \in \Theta \subseteq \mathbb{R}^d$ to generalise the relationship between data pairs (x, y). Moreover, $p(x)$ is the input distribution over the data, and $p(y|x; \theta)$ is the relationship between the input and output for a specific parameter θ in the parameter space Θ. The *Fisher information* can then be presented in the matrix form $F(\theta) \in \mathbb{R}^{d \times d}$ as follows [7, 34]:

$$F(\theta) = \mathbb{E}_{(x,y)\sim p} \left[\nabla_\theta \log p(x, y; \theta) \nabla_\theta \log p(x, y; \theta)^\intercal \right] \tag{7}$$

which is often approximated (with caution [34]) with *empirical Fisher* matrix:

$$\tilde{F}_k(\theta) = \frac{1}{k} \sum_{j=1}^{k} \nabla_\theta \log p(x_j, y_j; \theta) \nabla_\theta \log p(x_j, y_j; \theta)^\intercal \tag{8}$$

In this formulation and its extensions (such as Fisher-Rao norm [29]), Fisher information matrix can be used to explore the model geometry by calculating (or approximating by sampling) its natural gradient, which subsequently can also be used by the stochastic gradient-based neural network optimiser [35]. As Fisher information of classical statistics can be readily adopted to matrix formalism of quantum theory [36], it also finds applications in assessing the complexity of quantum models, and QNNs in particular [7].

2.3.2 Effective Dimension

There are two distinct approaches to measuring the model's effective quantum dimension. The first, called *global effective dimension (GED)* [7], is a measure of the model complexity based on its geometry defined by its entire parameter space (hence the name 'global'), regardless of data and algorithm used in the model training. The

benefit of this approach is that global effective dimension can be calculated from the model structure before its training, while also estimating the bounds on the model generalisation error. The second approach, known as *local effective dimension (LED)* [8], takes into consideration distribution of data and the learning algorithm, which create a geometrical sub-space of the model (hence the name 'local') that cannot be derived from the model structure alone. The LED benefit is that the measure is more sensitive to the dynamic features of the model, which only emerge during its training, and hence can be obtained at each optimisation step. Its main weakness, however, is that it is more costly to calculate.

Abbas et al. [7, 8, 37] calculate GED using Fisher information matrix as its metric. For a model $\mathcal{M}_\Theta = \{p(\cdot,\cdot,\theta) : \theta \in \Theta\}$, a d-dimensional parameter space $\Theta \subset \mathbb{R}^d$, $n \in \mathbb{N}$ and $n > 1$ data observations, and a stabilising parameter $\gamma \in (0, 1]$ ensuring that the formulas are meaningful for sufficiently large $n \to \infty$ (for $n < 20$, the formula may produce incorrect results), the model GED is estimated to be:

$$d_{\gamma,n}(\mathcal{M}_\Theta) = \frac{2\log\left(\frac{1}{V_\Theta}\int_\Theta \sqrt{\det\left(\mathrm{id}_d + \frac{\gamma n}{2\pi\log(n)}\hat{F}(\theta)\right)}\mathrm{d}\theta\right)}{\log\left(\frac{\gamma n}{2\pi\log n}\right)} \tag{9}$$

Where $\hat{F}(\theta) \in \mathbb{R}^{d\times d}$ is the normalised Fisher information matrix:

$$\hat{F}_{i,j}(\theta) = d\frac{V_\Theta}{\int_\Theta \mathrm{tr}(F(\theta))\mathrm{d}\theta}F_{i,j}(\theta), \tag{10}$$

and $V_\Theta \in \mathbb{R}_+$ is the volume of the entire parameter space:

$$V_\Theta = \int_\Theta \mathrm{d}\theta \tag{11}$$

Note that the size of data observations n is used to only define the granularity (or "resolution") of the model space, and does not relate to any specific data set or the model training [7]. However, when optimising QNN parameters with VQA, the optimiser actively explores the parameter values, with the aim of reducing the cost of results obtained from application of the model circuit to sample data. In this way, the optimisation process constrains and reduces the parameter space of the model, alters the cost landscape, and hence changes the QNN learning capacity as well. And yet, these processes are not captured in the GED calculation.

In a later study, Abbas et al. introduced *local effective dimension* [8], which is a measure of learning effectiveness correlated with data and the training algorithm, thus addressing the above issue.

Throughout the QNN model training, at each step its LED can be derived from the Fisher Information Matrix calculated around the evolving parameter values $\theta^\star \in \Theta \subset \mathbb{R}^d$, but within a small d-dimensional Euclidean ball of some radius $\epsilon > 0$:

$$\mathcal{B}_\epsilon(\theta^\star) = \{\theta \in \Theta : ||\theta - \theta^\star|| \leq \epsilon\} \tag{12}$$

with the volume of the Euclidean d-ball $V_\epsilon \in \mathbb{R}_+$

$$V_\epsilon = \int_{\mathcal{B}_\epsilon(\theta^\star)} d\theta = \frac{\pi^{d/2}\epsilon^d}{\Gamma(d/2+1)} \tag{13}$$

where Γ is Euler's Gamma function.

The LED measurement can be calculated around the reduced parameter space $\theta^\star \in \Theta$ with $\gamma \in (\frac{2\pi \log n}{n}, 1]$, and $\epsilon > \frac{1}{\sqrt{n}}$, and thus defined as follows.

$$d_{\gamma,n}(\mathcal{M}_{\mathcal{B}_\epsilon(\theta^\star)}) = \frac{2\log\left(\frac{1}{V_\epsilon}\int_{\mathcal{B}_\epsilon(\theta^\star)} \sqrt{\det\left(\mathrm{id}_d + \frac{\gamma n}{2\pi\log(n)}\hat{F}(\theta)\right)}\mathrm{d}\theta\right)}{\log\left(\frac{\gamma n}{2\pi\log n}\right)} \tag{14}$$

where matrix $\hat{F}(\theta)$ is the normalised Fisher information matrix $F(\theta) \in \mathbb{R}^{d\times d}$:

$$\hat{F}_{i,j}(\theta) = d\frac{V_\epsilon}{\int_{\mathcal{B}_\epsilon(\theta^\star)} \mathrm{tr}(F(\theta))\mathrm{d}\theta} F_{i,j}(\theta), \tag{15}$$

In this way, LED can provide insights into learning effectiveness over the process of model training. Importantly, it has also been shown that LED correlates with the model generalisation errors [8].

The measure can be interpreted as the effectiveness of the model parameter utilisation (in a scale from zero to the number of parameters), which is commonly normalised (by the number of parameters) to determine the utilisation of each parameter in the model (in a scale of 0 to 1, where 0 and 1 represent extremely poor and full parameter utilisation, respectively).

In this investigation of barren plateaus in QNN, as LED is sensitive not only to the model geometry (as defined by the QNN ansatz, its parameters and observable) but also to the dynamic process of model training (influenced by the cost function, the optimiser and data), it was therefore selected as the most suitable measurement of QNN learning capacity in our experiments.

3 Methods

Effectiveness of the selected BP mitigation strategies was determined by analysing their impact on training and testing of their respective QNN models and on their effective dimension, regardless of the BP presence in training.

Two groups of experiments were conducted (see Figure 5), each associated with a distinct data set, i.e. IRIS and MNIST. These two data sets were selected specifically as other authors working in the area of BP mitigation and effective dimension used them for their experimental work as well [7, 8]. The two data sets are fundamentally different as IRIS describes unique characteristics of flowers and MNIST is a collection of images of hand-written digits (each consisting of a grid of 28×28 black and white pixels). Both data sets were transformed into exactly the same structure for encoding in a quantum circuit. After the transformation both had 4 features and supported a binary classification task.

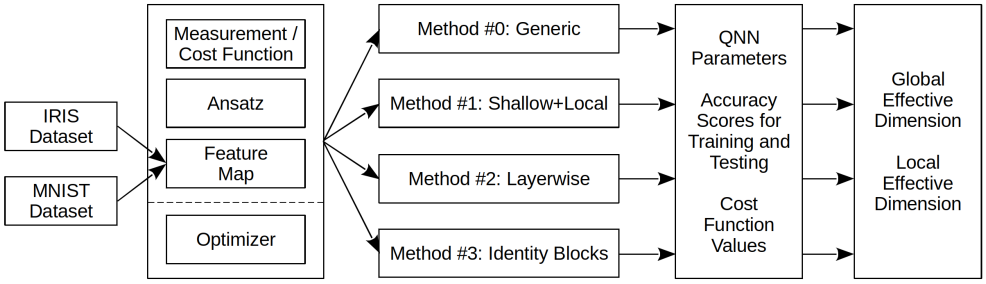


Figure 5: The adopted research framework of data, model components, BP countermeasures, measurements of training performance and learning capacity

A subset of IRIS data included (all) 100 records of 2 selected varieties of iris flowers, i.e. 'setosa' and 'versicolor' (50 each, classified by values -1 and 1), and described by their 4 characteristic features. A subset of MNIST data consisted of 500 randomly selected images of digits 0 and 1 (250 each, classified by values -1 and 1), and described by the 4 top principal components obtained from PCA analysis of the original MNIST data set. Both data sets were split into a training (70%) and testing (30%) data partition of balanced classes (-1 and 1).

In each group of experiments, four BP mitigation strategies (see Table 1) were utilised to implement and test a QNN classifier in Qiskit [38]. Each of the QNN circuits used in the experiments had 4 qubits, and consisted of three blocks representing a ZZ-feature map [39], an ansatz and measurements (as per Figure 1).

What follows are implementation details of the QNN structures and their ansatze for each of the four methods specified by their BP mitigation strategies (cf. Table 1).

Method #0 (Generic - No BP Strategy)

The goal of this QNN configuration was to create a performance benchmark against which the specific BP mitigation strategies could be compared. A general-purpose variational circuit was created to represent a multilayer perceptron. The circuit included a randomly initialised RealAmplitudes Qiskit ansatz. The circuit's measurement block, monitored the state of all qubits, and hence was designed to support a global cost function. The ansatz was designed for 4 qubits (as many as the number of features in both data sets), it had 9 repetitions of 4 rotational parameter blocks each (as well as included entanglement block), plus a default final parameter block of 4 rotations, which resulted in 40 parameters.

Method #1 (Shallow Circuit with Local Cost)

This approach to QNN design aimed to reduce the circuit depth, as well as, its state space by relying on the local cost function [16]. Therefore, the circuit training block was represented by a short RealAmplitudes Qiskit ansatz with 8 parameters only. The measurement block was designed to facilitate the local cost function, which was achieved by monitoring the state of 2 out of 4 qubits.

Method #2 (Layerwise Learning)

This strategy relied on the use of the ansatz identical to that used in Method #0. However, its parameters were zero initialised in a layerwise fashion by following the algorithm proposed by Skolik et al. [14] (as explained previously). The method aimed to avoid training of deep circuit blocks and to eliminate random parameter initialisation, both known to encourage formation of barren plateaus. The circuits created by this methods featured 40 parameters.

Method #3 (Identity Blocks)

This BP mitigation strategy was proposed by Grant et al. [17] (as explained previously). The method aimed to train short ansatz blocks and avoid both zero initialisation, as well as, a completely random parameter initialisation. Hence, the circuit was sub-divided into identity blocks. To this end, a custom multilayer Qiskit ansatz was created (see example in Figure 3). The ansatz was limited to 2 identity blocks of 20 rotation gates over 4 qubits, which resulted in a circuit of 40 parameters.

All experiments were scripted as Jupyter notebooks written in Python, featuring Qiskit APIs with PyTorch optimisers, to support the creation of QNN classifiers.

For each BP mitigation method, its prescribed model was implemented as a Qiskit EstimatorQNN. A Nadam optimiser was used with all models, using an L1 loss function (Mean Absolute Error cost). Ten instances of every EstimatorQNN were then trained with both IRIS and MNIST data, and their circuits executed using a quantum state vector simulator (i.e. simulation of a noise-free quantum machine).

Expectation values from each set of 10 runs were then analysed to obtain a fair observation of the model behaviour, e.g. to obtain the average model performance, account for the optimiser encountering local minima, occasional under- or over-fitting data, etc. Training and testing performance of all models was recorded for further processing.

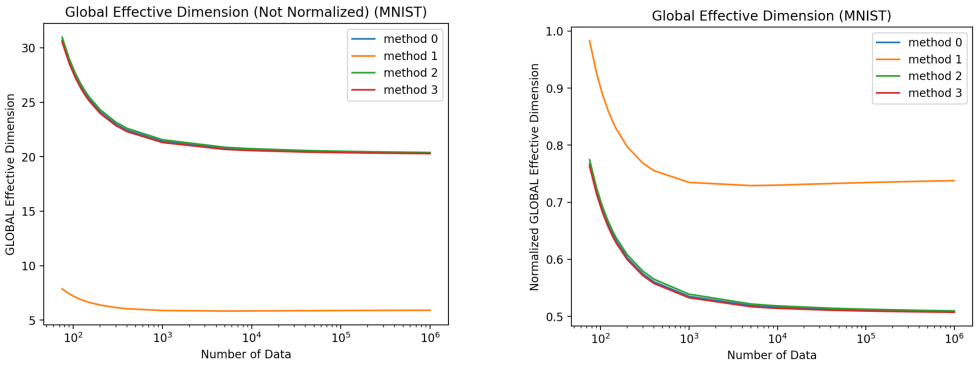
The QNN models and their learning processes were subsequently evaluated based on the records of their parameter values, accuracy scores on training and test data partitions, as well as cost function values (as Mean Absolute Error). Eventually, the models GED and LED were calculated (as given in Eqs. 9 and 14) for a sample of sufficiently large number of data observations (as required $n > 20$, so $n = 75 \ldots 1000000$). Due to the wide range of observation data, the effective dimension was subsequently plotted with the logarithmic x-scale (unless specified otherwise).

4 Results and Discussion

This section initially discusses GEDs characteristic of the QNN models prescribed by the four BP mitigation strategies and their methods. Then it investigates model training for each of the methods and their evolving LEDs.

Global Effective Dimension

The structure of the quantum circuits developed for each of the methods was dictated by the data set features and their respective BP mitigation strategy.



(a) (b)

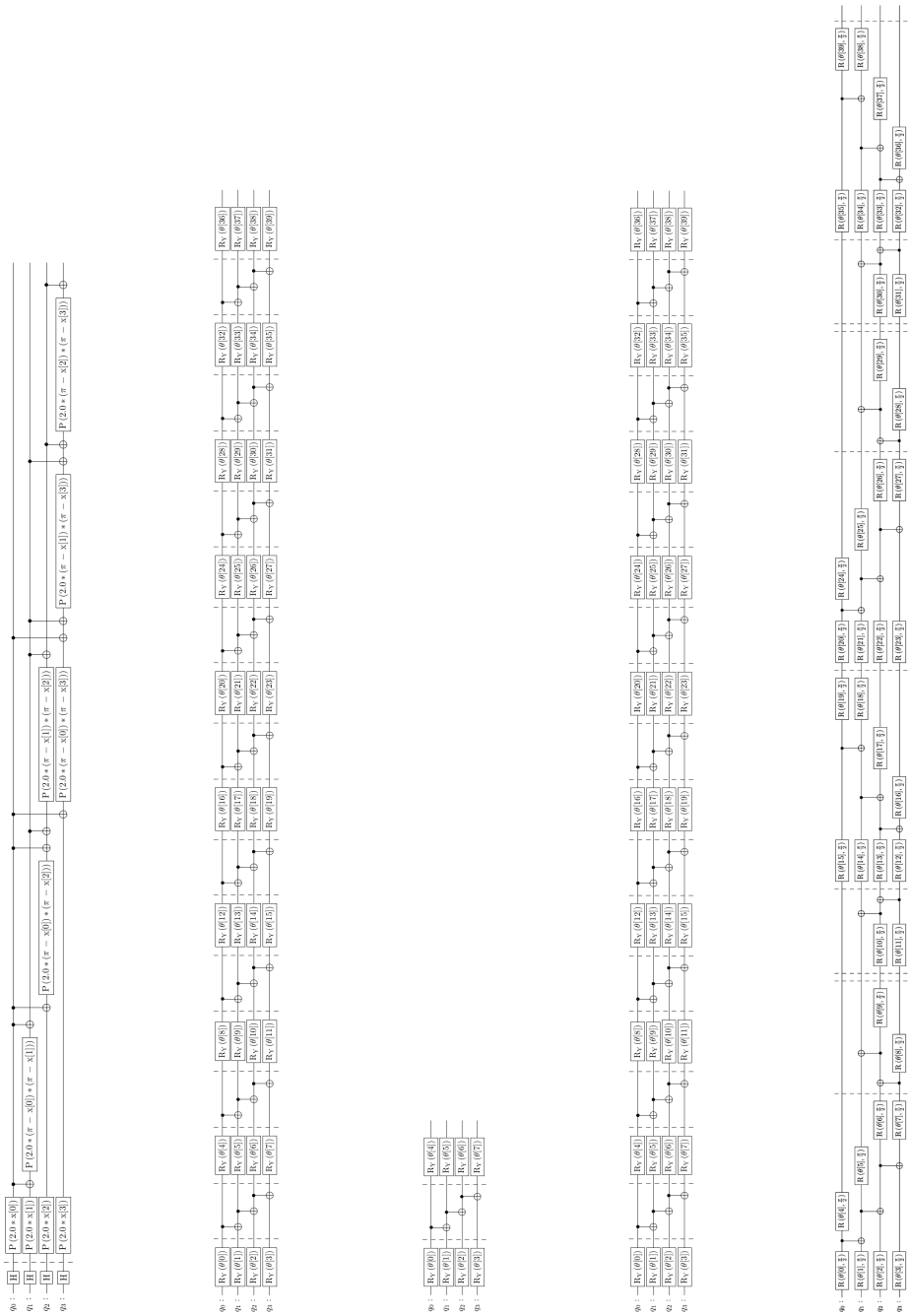
Figure 6: Global Effective Dimension (with log x-scale), which is identical for both IRIS and MNIST: (6a) Not normalised. (6b) Normalised.

As explained above, the subsets of both IRIS and MNIST data were transformed into identically structured data of 4 features and a binary label. Therefore, the QNNs developed for training with either IRIS or MNIST data relied on circuits, consisting of identical feature maps and ansatze. Their models were hence influenced only by the methods prescribed by their BP mitigation strategy. Subsequently, as GED calculation does not rely on data, Figure 6 depicting GED plots for circuits developed to support the methods #0, #1, #2 and #3 is equally representative of experiments associated with either IRIS or MNIST data sets.

The first aspect to note is that learning capacity, as measured with GED (normalised or not), of methods #0, #2 and #3 are nearly identical for all numbers of data observations n. This is not surprising, as methods #0 and #2 rely on the same ansatze (see Figure 7b and 7d) and differ only in their initialisation, hence their total parameter space and geometry are also the same. The circuit of method #3 (see Figure 7e) has exactly the same number of parameters as compared with those used in methods #0 and #2 (40 each). However, it is different in its qubit entanglements and the allocation of rotational parameters. Yet, these differences seemed to have little impact on the calculation of Fisher Information Matrix.

The learning capacity of the ansatz used in method #1 is drastically different. Initially consider the global effective dimension without any normalisation (as depicted in Figure 6a), which represents the learning capacity of the entire ansatz. It is evident that irrespective of the number of observation data, a shallow circuit of only 8 parameters, as associated with method #1 (see Figure 7c), has less than a third of the capacity of deeper circuits of 40 parameters, as used by methods #0, #2 and #3. The more circuit parameters, the more learning capacity.

At the same time, it seems that if we consider the effective use of each individual circuit parameter, as captured by normalised GED (see Figure 6b), to learn the same data complexity method #1 utilises its parameters more extensively than the remaining three methods.



(a) (b) (c) (d) (e)

Figure 7: Circuit components: the common feature map (7a) and the specific ansatze for - Method #0 (7b), Method #1 (7c), Method #2 (7d) and Method #3 (7e).

Method	IRIS Accuracy	MNIST Accuracy
#0	0.90667	0.97
#1	0.79667	0.85067
#2	0.9	0.956
#3	0.84667	0.94267

Table 2: Average peak accuracy for all method's instances in testing

Method	IRIS Variance	MNIST Variance
#0	0.00177	0.00066
#1	0.00351	0.00289
#2	0.0	0.0
#3	0.0024	0.00087

Table 3: Accuracy variance in testing of all method's instances

Normalised effective dimension focuses on the effectiveness of an individual parameter in a circuit. Its analysis is, therefore, better suited comparison of different QNN models, or the same model undergoing changes and improvements, irrespective of the overall number of parameters in use, which can always be increased as needed. For this reason, normalised GED and LED results are given more prominence in this survey of BP mitigation methods, than the results without normalisation.

Model Training for LED Measurement

Measuring LED required training of QNN models. The LED measurements occurred at regular intervals of each model evolution, during its optimisation process. Subsequently, the recorded measurements were saved for later plotting and analysis.

All models trained with IRIS data required up to 150 iterations to converge (see Figures 8a, 8c, 8e). The models trained with MNIST data required only 120 iterations (see Figures 8b, 8d, 8f). Due to the small size of IRIS test data and Nadam's tendency to incur stochastic noise in some circumstances, its testing accuracy suffered from a higher level of volatility (see Figures 9e vs 9f, dark lines) and a higher inter-instance variance (same Figures, light area around dark lines), as compared with MNIST testing.

When analysing the methods' training and testing accuracy (cf Table 2 and Figure 8), regardless of the data set used in training, the shallow ansatze with local cost function performed worst in both. The test scores obtained for methods #0 and method #2 were best. Method #3 score was marginally behind them. In the absence of quantum indeterminacy (due to the use of state vector simulator), within each method, variance in accuracy can be attributed to differences in the initialisation of model instances (cf Table 3).

Local Effective Dimension

LED measurement depends on the number of data observations and the stage of model optimisation, as denoted by the the number of iterations spent in model training. Its shape is therefore three dimensional (see Figures 9a, 9b).

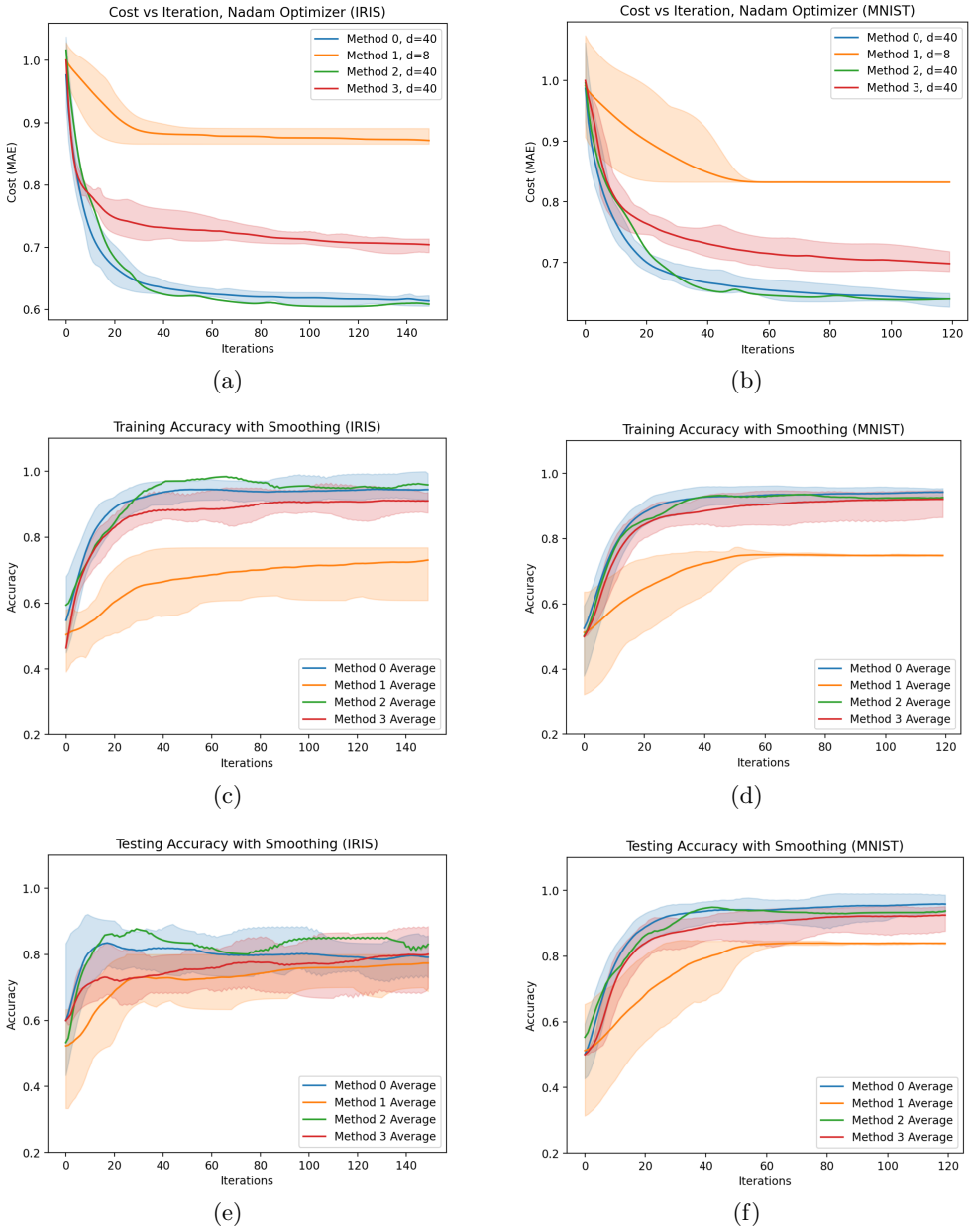


Figure 8: QNN training performance: (8a, 8c, 8e) IRIS data set. (8b, 8d, 8f) MNIST data set. Note that smoothing was applied to all plots.

To increase the chart readability, we represent it in two dimensions by plotting only the selected few iterations (see Figures 9c, 9d). While the chart clarity is thus improved, unfortunately, it loses some of its details.

As evident from Figure 9, and previously reported by Abbas et al [8], LED for an individual QNN closely aligns with the model generalisation errors, which are here represented in terms of test accuracy.

It is important to note that irrespective of the number of observation data n used in calculating the local effective dimension, its measure grows with the model test scores but then degrades when generalisation errors increase. For example, within method #0, model instances 7 for IRIS and 3 for MNIST, iterations 40 and 60 coincide with the peak test accuracy as well as with LED values (which are signified by the green line in Figures 9c and 9d).

When analysing multiple model instances within each method of dealing with BPs (see Figures 10a, 10c, 10e for IRIS and 10b, 10d, 10f for MNIST), we can observe three important stages of the model evolution, which is reflected in their LED measurement, i.e. immediately at their initialisation, at peak testing performance, and at the end of training. First, it can be noted that the mean LED for all of the model instances at their initialisation (before training) is a strong predictor of their LED at the subsequent improvement. With a notable exception of method #2 training with IRIS data, the models' LED does not change at large numbers of data observations (for $n > 10,000$) after the best test performance was reached.

Interestingly, the methods' LED seems to be less impacted by data used in models training (such as drastically different IRIS and MNIST data sets), as it is influenced by the model architecture and its initialisation strategy (especially for large n, as can be observed in pairs of Figures 10a, 10b; Figures 10c, 10d; and Figures 10e, 10f).

In multi-instance training, where experiments were repeated 10 times for each model type, the QNN initialisation strategy had a pronounced effect not only on the model testing performance (as seen in Figure 8) but also its LED (see Figure 10). It is clear that those methods, which allowed any part of their circuit to be randomly initialised (even when counteracted with identity blocks), exhibited differing learning behaviour of their instances and consequently their different LEDs, which can be observed as variance bands around their average instance LED, especially in the early stages of their training (see Figure 10a, 10b). The worst affected method was method #1, which relied on the shallow ansatze, but which rapidly reduced the LED variance once the models converged in training (see Figure 10c-10f). Except for method #3 which maintained high LED variance across it models' evolution, the models of the remaining methods converged and reduced their LED variance. Method #2, which adopted parameter initialisation to zero, resulted in all identical instances of its models (and no variance bands). The normalised LED for the method #3 is consistently lowest of all models at all stages. However, it needs to be recalled that normalised LED identifies an average effectiveness of a single model parameter. This means that LED measures of methods #0, #2 and #3 are penalised by the large number of parameters in use. If we were to compare their raw LEDs denoting the effectiveness of their entire circuits, method #1 would fair worst, while the remaining three methods would display the same relative effectiveness as they use exactly the same number of parameters.

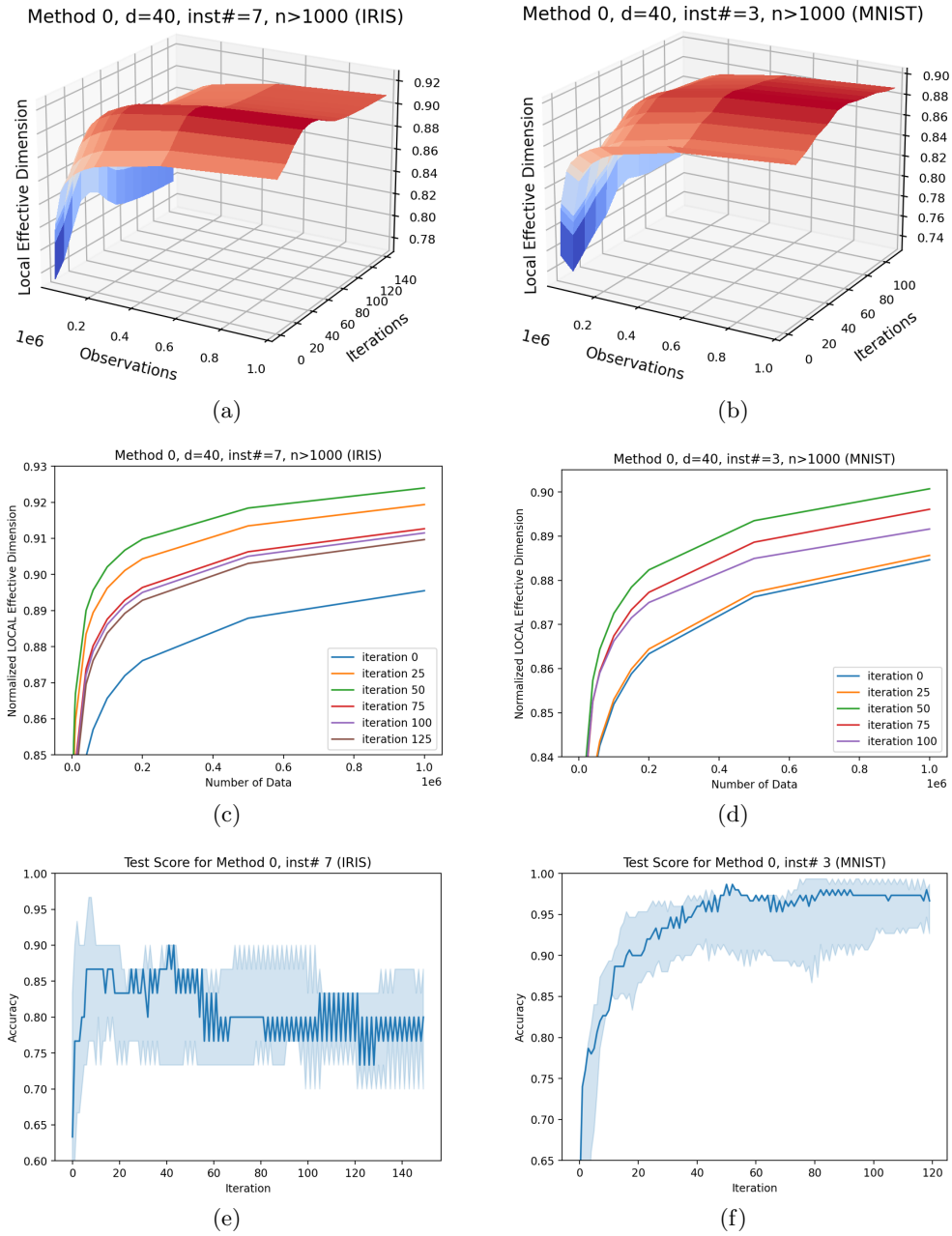


Figure 9: Alignment of the local effective dimension with test scores (linear x-scale, no smoothing): (9e, 9f) dark line denotes the selected instance, light area represents the value range of all instances. (9a, 9c, 9e) IRIS data set. (9b, 9d, 9f) MNIST data set.

Method #0 displayed excellent test accuracy (cf Table 2) and the highest LED in all stages of model training with both IRIS and MNIST data (cf Figure 10). This is in contrast with McClean et al [4] observation that random parameter initialisation could lead to BP development and inferior model training. The results presented here, however, indicate that random parameter initialisation should be included as a viable option and tested, and discarded only when problems in model training emerge.

It is worth noting an important aberration in method #2 training. When models were initialised in a layerwise fashion with IRIS data, as prescribed by method #2, their LED continued to increase beyond the point of their best testing performance (cf Figure 10e vs 10c). This goes against the previously discussed, and reported in the literature [8], alignment between LED and generalisation errors. Such a behaviour, however, was not detected in models trained with method #2 and MNIST data (cf Figure 10f vs 10d). The likely explanation for this occurrence was high volatility in QNN models training with IRIS, which resulted in early peak in test performance, its subsequent drop, and then a repeated increase in test scores in later iterations of the model optimisation (cf Figure 8e).

Finally, note that LED is defined for the "sufficiently large" numbers of data observations [8], such as $n > 20$, however, in practice $n > 5000$ is recommended [40]. Selection of very large n as a starting point in LED calculation results in monotonically increasing LED charts (cf Figures 9c, 9d, with linear x-scale). The notion of what is "sufficiently large" is inadequately defined. As evident from Figure 10 (with log x-scale), for small n LED calculations initially exhibit a downward trend. The LED minimum is reached at different training stages for different methods and their model instances, and only then the LED values assume their upward trend.

5 Conclusion

Many studies report effectiveness of different strategies to mitigate the problem of emerging barren plateaus in QNN training [14, 16–18], which includes altering properties of the circuit ansatz (e.g. shallow or deep), application of different cost functions (e.g. local or global), or reliance on various circuit initialisation strategies (e.g. random, layerwise or identity preserving blocks).

In this work, we compared the methods associated with four distinct BP mitigation strategies, i.e. #0 - common practice (deep ansatz, global cost and random parameter initialisation), #1 - shallow ansatz and local cost function, #2 - utilising layerwise ansatz intialisation, and #3 - reliance on identity blocks in circuit initialisation.

In all four cases, we developed and evaluated a number of QNN classifiers by training them with two distinct data sets (IRIS and MNIST). We analysed their training and test accuracy scores, as well as, their learning capacity by measuring their global and local effective dimension. In our experiments, we did not investigate the methods performance in the face of the actual barren plateaus, but rather whether their application boosts or impedes the model optimisation, its performance and its learning capacity.

We found that the commonly practised approach to QNN development was the most beneficial to the model performance. Its models provided the best test scores

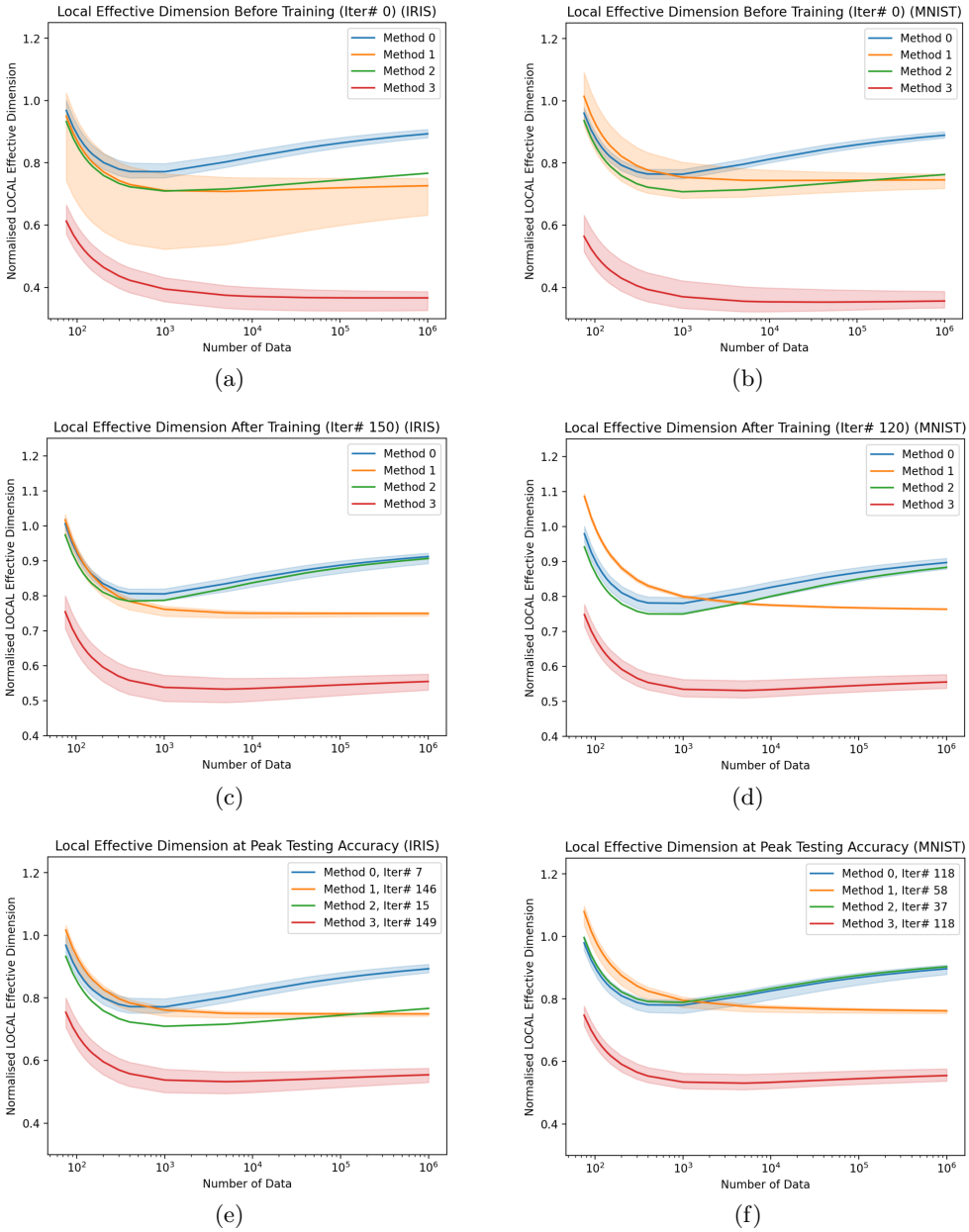


Figure 10: Local effective dimension (with log x-scale): (10a, 10c, 10e) IRIS data set. (10b, 10d, 10f) MNIST data set.

and the highest local effective dimension. However, as observed in prior studies, deep-circuit models with random parameter initialisation promote development of barren plateaus. For this reason we recommend to use this approach with a great deal of caution, in spite of the positive results obtained in this study.

The layerwise parameter initialisation produced the most effective model with high testing accuracy and a consistently high local effective dimension. Although finding the initial parameter values was computationally expensive, the benefits of the layerwise approach were clearly visible in the subsequent model training and its quality.

In all tests, models initialised with identity blocks performed relatively well on accuracy scores, and yet, the worst in their effective dimension measurements. This level of performance could be attributed to the sensitivity of the method to the number and depth of deployed identity blocks - as the method creators remarked [17]: each block must be "sufficiently deep" and "sufficiently shallow" to meet the training criteria. Perhaps our future work could investigate if the identity-block strategy could be improved by combining it with layerwise training block-by-block.

QNN models with shallow circuits and local cost function converged earliest in training and reduced their large initial variance quickest, as predicted by the preliminary analysis of the models' gradient decay. Unfortunately, the model overall capacity was insufficient to learn data of any significant complexity. The lesson learnt in this case is that a QNN circuit must be as short as possible but not shorter. The second lesson is that the analysis of gradient variance and decay alone is insufficient to properly assess the model ability to deal effectively with barren plateaus and more importantly with model learning.

While the shallow circuits were found to be poor learners, their local effect dimension was relatively high. In consequence, we found that local measures need to be used with caution. This is especially when relying on their normalised values, which are useful to compare models, evaluate the model variants and assess its future changes, but which are unable determining the learning capacity of a QNN circuit as a whole, the task which is better handled with raw measurements. It is also a reminder for QNN developers that effective dimension should not be used as a sole measure of a model training success.

Global effective dimension provides a good estimate of the model structural support for learning. However, it fails to distinguish between good and poor models in their dynamic learning environment, which is influenced by the properties of training data, selection of measurement strategy and cost function, model initialisation, as well as, the detailed approach to the model optimisation. The local effective dimension measures are better equipped to do so.

Furthermore, local effective dimension at the time of the model initialisation is a good predictor of its final capacity to learn. The moment of the model convergence, as established by the peak in test scores, is also a good indicator of the final measurements of learning capacity. However, care must be taken to observe any deviations from this behaviour as this may be the sign of unreliable performance scores, e.g. due to volatility in training.

In our experiments we found with some surprise that the local effective dimension seems to be more influenced by the QNN architecture and its optimisation choices,

rather than the nature of data used in the model training. However, as our investigation utilised two data sets only, further work needs to be undertaken to conduct additional tests.

In summary, this research investigated effectiveness of strategies countering the emergence of barren plateaus in training quantum neural networks. We demonstrated usefulness of measuring the effective dimension of quantum models to assess their evolving learning capacity. We trust that our experimental approach to studying quantum neural networks provided some valuable insights into development of such models.

6 Compliance with Ethical Standards

The majority of work presented in this article was conducted when both authors were associated with Deakin University, Melbourne, Australia.

Author Jacob L. Cybulski holds an Honorary Associate Professorship at the School of IT, Deakin University and has no relevant financial or non-financial interests to disclose. Author Thanh Nguyen's contribution to this project was partially funded by 2022 Summer Project Prize from the School of IT, Deakin University. He has no other competing interests to declare that are relevant to the content of this article.

Data used in this project, such as IRIS and MNIST data sets, are in public domain and do not require a separate ethics permission to use. All data and programs which resulted from this project are in public domain.

7 Data Availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

[1] Schuld, M., Sinayskiy, I. & Petruccione, F. The quest for a quantum neural network. *Quantum Information Processing* **13**, 2567–2586 (2014).

[2] Schuld, M. & Petruccione, F. *Machine Learning with Quantum Computers* 2nd ed. 2021 edition edn (Springer, 2021).

[3] Dawid, A. *et al.* Modern applications of machine learning in quantum sciences. *arXiv preprint arXiv:2204.04198* (2022).

[4] McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R. & Neven, H. Barren plateaus in quantum neural network training landscapes. *Nature communications* **9**, 1–6 (2018).

[5] Zhao, C. & Gao, X.-S. Analyzing the barren plateau phenomenon in training quantum neural networks with the ZX-calculus. *Quantum* **5**, 466 (2021).

[6] Brownlee, J. *Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions* (Machine Learning Mastery, 2018).

[7] Abbas, A. *et al.* The power of quantum neural networks. *Nature Computational Science* **1**, 403–409 (2021).

[8] Abbas, A., Sutter, D., Figalli, A. & Woerner, S. Effective dimension of machine learning models. *arXiv preprint arXiv:2112.04807* (2021).

[9] Cerezo, M. *et al.* Variational quantum algorithms. *Nature Reviews Physics* 1–20 (2021).

[10] Zaheer, R. & Shaziya, H. A study of the optimization algorithms in deep learning. *Third International Conference on Inventive Systems and Control (ICISC)* 536–539 (2019).

[11] Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint* (2017).

[12] Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[13] Dozat, T. Incorporating nesterov momentum into Adam. *4th International Conference on Learning Representations (ICLR)* **Workshop Track, paper 43** (2016).

[14] Skolik, A., McClean, J. R., Mohseni, M., van der Smagt, P. & Leib, M. Layerwise learning for quantum neural networks. *Quantum Machine Intelligence* **3**, 1–11 (2021).

[15] Wang, S. *et al.* Noise-induced barren plateaus in variational quantum algorithms. *Nature communications* **12**, 1–11 (2021).

[16] Cerezo, M., Sone, A., Volkoff, T., Cincio, L. & Coles, P. J. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature Communications* **12**, 1791 (2021).

[17] Grant, E., Wossnig, L., Ostaszewski, M. & Benedetti, M. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum* **3**, 214 (2019).

[18] Uvarov, A. & Biamonte, J. On barren plateaus and cost function locality in variational quantum algorithms. *Journal of Physics A: Mathematical and Theoretical* **54**, 245301 (2021).

[19] Storwick, T. Alleviating barren plateaus with local cost functions — PennyLane (2021).

[20] Little, W. A. & Shaw, G. L. Analytic study of the memory storage capacity of a neural network. *Mathematical Biosciences* **39**, 281–290 (1978).

[21] Newman, C. M. Memory capacity in neural network models: Rigorous lower bounds. *Neural Networks* **1**, 223–238 (1988).

[22] Gardner, E. & Derrida, B. Optimal storage properties of neural network models. *Journal of Physics A: Mathematical and General* **21**, 271 (1988).

[23] Gardner, E. The space of interactions in neural network models. *Journal of Physics A: Mathematical and General* **21**, 257 (1988).

[24] LeCun, Y., Denker, J. & Solla, S. Optimal brain damage. *Advances in Neural Information Processing Systems (NIPS)* **2** (1989).

[25] Vapnik, V., Levin, E. & Cun, Y. L. Measuring the VC-Dimension of a Learning Machine. *Neural Computation* **6**, 851–876 (1994).

[26] Vapnik, V. N. & Chervonenkis, A. Y. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Applications* **16**, 264–280 (1971).

[27] Baum, E. & Haussler, D. What size net gives valid generalization? *Advances in neural information processing systems* **1** (1988).

[28] Karakida, R., Akaho, S. & Amari, S.-i. Universal statistics of Fisher information in deep neural networks: Mean field approach. *Journal of Statistical Mechanics: Theory and Experiment* **2020**, 124005 (2020).

[29] Liang, T., Poggio, T., Rakhlin, A. & Stokes, J. Fisher-Rao Metric, Geometry, and Complexity of Neural Networks. *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics* 888–896 (2019).

[30] Lewenstein, M. *et al.* Storage capacity and learning capability of quantum neural networks. *Quantum Science and Technology* **6**, 045002 (2021).

[31] Larocca, M., Ju, N., García-Martín, D., Coles, P. J. & Cerezo, M. Theory of overparametrization in quantum neural networks (2021). arXiv:2109.11676.

[32] Haug, T., Bharti, K. & Kim, M. S. Capacity and quantum geometry of parametrized quantum circuits. *PRX Quantum* **2**, 040309 (2021).

[33] Ly, A., Marsman, M., Verhagen, J., Grasman, R. & Wagenmakers, E.-J. A Tutorial on Fisher Information (2017). arXiv:1705.01064.

[34] Kunstner, F., Hennig, P. & Balles, L. Limitations of the empirical Fisher approximation for natural gradient descent. *Advances in neural information processing systems* **32** (2019).

[35] Amari, S.-i., Karakida, R. & Oizumi, M. Fisher Information and Natural Gradient Learning of Random Deep Networks. *The 22nd International Conference on Artificial Intelligence and Statistics* 694–702 (2019, April).

[36] Petz, D. & Ghinea, C. Introduction to quantum Fisher information. *Quantum Probability and Related Topics* 261–281 (2011).

[37] Berezniuk, O., Figalli, A., Ghigliazza, R. & Musaelian, K. A scale-dependent notion of effective dimension. *arXiv preprint arXiv:2001.10872* (2020).

[38] Qiskit. Qiskit Textbook. https://qiskit.org/learn/ (2023).

[39] Havlíček, V. *et al.* Supervised learning with quantum-enhanced feature spaces. *Nature* **567**, 209–212 (2019).

[40] Qiskit. ML Tutorial: Effective Dimension. https://qiskit.org/ecosystem/machine-learning/tutorials/10_effective_dimension.html (2023).