Introduction to Quantum Machine Learning and Quantum Neural Networks

Andranik Sahakyan
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218
asaakya2@jhu.edu

Abstract

Quantum machine learning is a promising area of research that explores the use of quantum algorithms to find patterns in classical data. Given that linear algebra is at the essence of most machine learning algorithms and there exist exponentially faster quantum analogs for computing the fast Fourier transform (FFT), finding eigenvalues and eigenvectors, and performing matrix inversion, machine learning seems like a natural application for exploiting quantum properties such as superposition, entanglement, and quantum parallelism for solving classically intractable problems. In this paper, we provide an introduction to quantum machine learning (QML) and quantum neural networks (QNN), and discuss a few proposed models, challenges, and applications of QNN.

1 Overview of Classical Machine Learning

Machine learning is a subfield of artificial intelligence that involves building models capable of learning from data without being explicitly programmed to do so. Machine learning algorithms are divided into three main categories: supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, the learner is given a set of input-output pairs (x, y) representing the feature vector and its associated labels, respectively. A supervised learning model assumes there is some true relationship between the input x and the output y, and the goal is to learn a parameterized function $\hat{y} = f(x; \theta)$ that maps the input to the expected output. In unsupervised learning, the learner is given a set of unlabelled inputs and its task is to discover underlying patterns in the data. Perhaps the most popular unsupervised learning task is clustering, in which the learner assigns each data point to a cluster based on some similarity metric. In reinforcement learning, the learner, referred to as an agent, is given neither an input vector nor a set of labels. Rather, the agent receives feedback in the form of rewards by interacting with its environment. The goal of the agent is to learn a policy that maximizes its expected reward. This paper will focus on supervised learning.

The simplest type of model is the linear model $\hat{y} = \theta x + b$. However, many complex real-world relationships are nonlinear in nature. Thus, we can make our linear model more expressive by adding a source of nonlinearity in the form of some activation function. This is the premise behind the McCullough-Pitts perceptron, which is a simplified model of a biological neuron, giving the equation $\hat{y} = \sigma(\theta x + b)$.[13] Extending this idea further, we can create arbitrarily many layers of neurons and inject nonlinearities in between the layers, leading to the feedforward artificial neural network (ANN) or multilayer perceptron (MLP) model. For example, a feedforward ANN with one hidden layer can be written as $\hat{y} = \sigma_2(\theta_2\sigma_1(\theta_1 x + b_1) + b_2)$.

According to the Universal Approximation Theorem, a neural network with a single hidden layer can approximate any continuous function within a finite range. Since most classes of problems can be reduced to functions, artificial neural networks are extremely powerful models that can be applied

to problems in any field of study. We will revisit the expressive power of neural networks when we discuss their quantum analogs.

2 Introduction to Quantum Machine Learning

Quantum machine learning is the use of quantum algorithms within machine learning programs. This field, which is still at its infancy, is trying to explore any potential benefits or advantages from making some part of the machine learning model *quantum*. This can be accomplished in a few different ways, summarized in Figure 1. The quantum component can either be the data processing device, the data generating system, or both. In the first quadrant we have classical data analyzed by a classical computer – this refers to all of classical machine learning. We can also use classical machine learning to analyze quantum data, such as the results of some quantum mechanical experiment. Another approach is to analyze quantum data on a quantum computer. However, the term *quantum machine learning* usually refers to the case of analyzing classical data on a quantum computer. There are also hybrid approaches that involve both classical and quantum processing, where computationally expensive subroutines are outsourced to the quantum computer.[5,8,18]

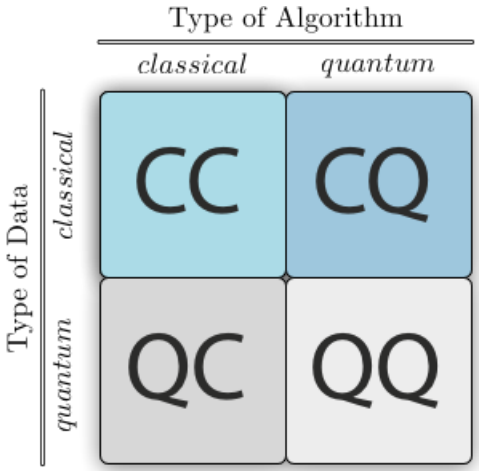

Figure 1: Different approaches of combining quantum computing and machine learning.

3 Why is QML Interesting?

Although the term *quantum machine learning* immediately sounds like a buzzword, there is actually some intuition as to why applying quantum algorithms to machine learning may yield promising results. The mathematics of quantum mechanics is all about linear algebra on very high-dimensional vector spaces. At its core, many machine learning algorithms also reduce to performing linear algebra on high-dimensional vectors. Some of these common linear algebra problems include computing the fast Fourier transform (FFT), finding eigenvalues and eigenvectors, and performing matrix inversion. Table 1 summarizes the time complexity of the classical and quantum versions of these algorithms. There is an important caveat regarding the exponential speed-up of the quantum algorithms which we discuss in the *Future Directions* section.

Table 1: Time Complexity Comparison of Classical and Quantum Linear Algebra Algorithms

Algorithm	Classical	Quantum
FFT	$O(dlogd)$	$O((logd)^2)$
Finding eigenvalues/eigenvectors	$O(d^3)$*	$O((logd)^2)$
Matrix inversion	$O(dlogd)$	$O((logd)^3)$

*$O(sd^2)$ for low-rank/sparse matrices

An exponential speed-up has immense real-world implications for what can be considered computationally feasible. For example, consider an FFT computation on a 1 terabyte dataset. The classical algorithm requires on the order of $10^{12} \times log(10^{12}) \approx 3.99 \times 10^{13}$ operations, while the quantum algorithm only requires $((log(10^{12}))^2) \approx 1600$ operations. A computation requiring 40 qubits and a few thousand operations is feasible with the type of small-scale quantum computers we expect to see in the near future. Additionally, due to the exponentially compact representation of the data, a computation at this scale would not require the type of quantum error correction we would need to factor large integers using Shor's algorithm.

4 Introduction to Quantum Neural Networks

Quantum models can be categorized into *deterministic* and *variational* models. Deterministic models refer to quantum circuits that produce deterministic values – that is, we can be certain what the outcome of the measurement will be. Variational models are stochastic and parameterized, so instead of directly measuring the result, we must repeat the experiment multiple times to gather statistics about the expected measurement outcome. Quantum neural networks are a type of variational model in which the goal is to optimize the weights of the neurons to measure the expected class label with high probability. In this section, we take a deeper look at the various components of QNNs: state preparation, model architecture, training methods, learning rate and runtime, and unique challenges affecting QNNs.

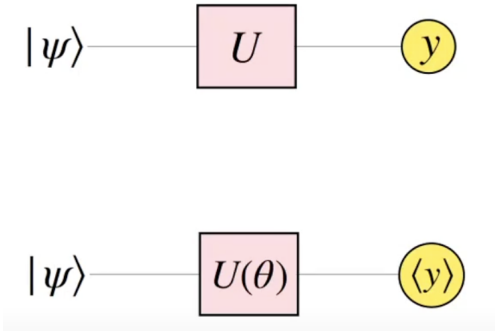

Figure 2: Top – deterministic model, Bottom – variational model

4.1 State Preparation and Input Encoding

Before we can take advantage of quantum algorithms, our classical data must be transformed to be represented as quantum states. This initial encoding step poses a much larger challenge than expected, with state preparation taking exponential time in the worst case. This is important because many quantum algorithms, including the algorithms mentioned in the previous section, assume that data can be loaded in linear or logarithmic time.[22] Both the data and the encoding method influence the runtime of state preparation.[19] Some of the challenges of the loading process include quantum decoherence, error-prone quantum gates, and quantum-mechanical restrictions such as the no-cloning theorem.[1] Existing quantum computers are limited in both the quantity and quality of qubits – the qubits are only stable for a short period of time. Barring the invention of a quantum random access memory (qRAM), this means we need to reduce the number of operations involved in state preparation to achieve faster loading times. The classic time-space trade-off of computer science applies in the quantum world as well, with different encoding strategies trading off the number of required qubits for the runtime complexity of state preparation.

4.1.1 Basis Encoding

For quantum algorithms that need to manipulate real numbers, a simple strategy is to approximate the real number with a bit string and translate it bit-wise to the corresponding basis state in the

[1]The no-cloning theorem states that it is impossible to create an identical and independent copy of an arbitrary unknown quantum state[23]

computational basis $\{|0\rangle, |1\rangle\}$. For example, if we represent the real number 2 with the bit string *10*, then its basis encoding will be $|10\rangle$. Thus, basis encoding requires *n* qubits for an *n* bit number. To implement this encoding, each qubit in the initial state $|0\rangle$ must be flipped to $|1\rangle$, which can be done in linear time.[22]

4.1.2 Angle Encoding

Another encoding strategy that requires a linear number of qubits is angle encoding. For this encoding, we create *n* qubits for an *n*-dimensional input vector. Then, we apply Hadamard gates to all *n* qubits, creating an equally weighted superposition of 2^n states. To encode the data, we apply rotation gates to each qubit, where the angle of rotation of the *i*-th qubit is equal to the value of the *i*-th component of the input vector. Both basis encoding and angle encoding are efficient in runtime complexity but inefficient in the number of required qubits.

$$x^1 = \begin{bmatrix} 0.888 \\ -1.25 \end{bmatrix}$$

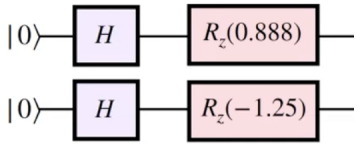

Figure 3: Example of angle encoding

4.1.3 Amplitude Encoding

Amplitude encoding is a clever encoding strategy that encodes the input vector into the amplitudes of the wave function. Every quantum system is described by its wavefunction ψ, and the square modulus of the amplitudes represent the measurement probabilities for each outcome. Therefore, the data values of the input vector must first be normalized to unit length. To associate each amplitude to a component in the input vector, the dimension of the input vector must be a power of two because the vector space of an n qubit register has dimension 2^n. If this is not the case, we can pad the input vector with zeros to increase the dimension.[22] Note that this does not change the result since an amplitude of zero means the probability of observing that outcome is zero. A normalized classical N-dimensional vector x is represented by the amplitudes of an n-qubit quantum state ψ_x as $\psi_x = \sum_i^N x_i |i\rangle$, where $N = 2^n$, x_i is the i-th element of x, and $|i\rangle$ is the i-th computational basis state.[16]

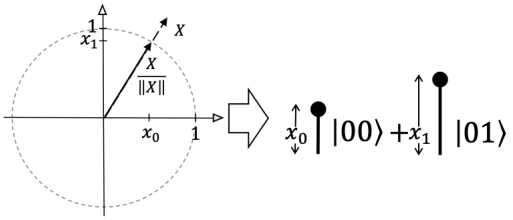

Figure 4: Visualization of amplitude encoding

For example, suppose we wanted to encode $x = [1.0, 0.0, -3.2]^T$ using amplitude encoding. Notice that the dimension of x is not a power of two, so we will increase its dimension to the nearest power of two by padding it with an additional zero. Then, we normalize the values of x so the square modulus of the encoded amplitudes form a probability distribution of the measurement outcomes of $|\psi_x\rangle$: $x_{norm} = \frac{1}{\sqrt{11.24}}[1.0, 0.0, -3.2, 0.0]^T \implies \frac{1}{\sqrt{11.24}}(|00\rangle - 3.2\,|10\rangle)$.

Amplitude encoding is very qubit-efficient, only requiring $\lceil log(NM) \rceil$ qubits to encode a dataset of M inputs with N features each. This is because n qubits can represent 2^n data values. A huge factor in the quantum speed-up is due to this exponential compression in the representation of the classical data. The states of quantum mechanical systems are vectors in very high-dimensional vector spaces, and with a very small number of qubits we are able to represent data residing in an exponentially high vector space. However, amplitude encoding state preparation routines for arbitrary data vectors require an exponential number of operations. Although there are special cases for which we have faster routines, such as sparse data vectors, the theoretical lower-bound of the depth of an arbitrary state preparation circuit is known to be $\frac{1}{n}2^n$, with current approaches taking about 2^n operations.[17] Amplitude encoding is required for many QML algorithms, including the HHL algorithm for solving systems of linear equations.[9] Further research into efficient state preparation routines is therefore a critical area of research for realizing the promised quantum speed-up of QML algorithms.

4.1.4 qRAM Encoding

A classical random access memory (RAM) is a memory device that takes an index and loads the data value stored in that address into an output register. A quantum random access memory (qRAM) is similar, except the address and output registers are quantum registers. This means that the registers can be in a superposition of multiple values. Figure 5 demonstrates the use of a qRAM to load the data value of the first two quantum registers into an empty output register. Note that the index is in a *superposition* of the first two addresses ($\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|01\rangle$) and the result in the output register is in a superposition of the addresses with their corresponding data values: $\frac{1}{\sqrt{2}}|00\rangle|000\rangle + \frac{1}{\sqrt{2}}|01\rangle|110\rangle$. qRAM encoding uses basis encoding for the data values, which we have already seen requires a linear number of qubits, and $\lceil log(n) \rceil$ qubits to represent 2^n addresses. This results in an encoding with very desirable properties – namely, since the encoded data values may be in superposition, we can take advantage of quantum parallelism to operate on all values in the superposition in a single timestep. Additionally, because we use basis encoding for the data values, we can perform arithmetic operations on our data. Currently, we do not have any physical implementation of a qRAM, so the same caveat regarding quantum speed-up applies in this case as well.[21]

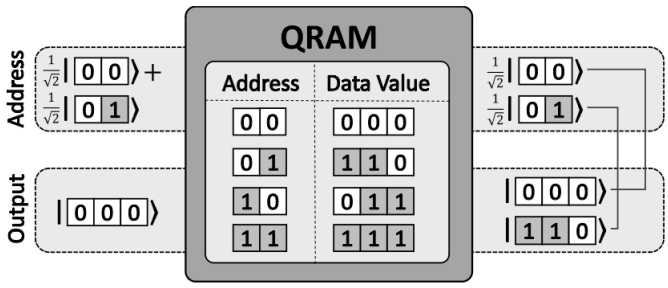

Figure 5: Example of qRAM encoding

4.2 QNN Model Architectures

Before we take a look at QNN model architectures, let us take a step back and understand where this component fits into the overall quantum computing model (summarized in Figure 6). This diagram is the same for all quantum computation of classical data. We begin by pre-processing the classical data – this step might involve applying dimensionality reduction techniques such as principal component analysis (PCA) to make more efficient use of limited qubits. Next, we use some encoding strategy and state preparation subroutine to encode our classical data into quantum states. The next step is where we apply the QNN model. Like all quantum circuits, this component consists of a set of unitary (and therefore linear) operators. This may seem like a disadvantage of QNNs since the expressive power of classical neural networks is largely due to the nonlinear activation functions. However, the measurement operation itself can be a source of nonlinearity in the quantum model. Unlike deterministic quantum circuits, variational circuits have parameterized gate operations which are

optimized in the training process. We will discuss the Quantum M-P Neural Network proposed by Zhou and Ding, and a QNN model by Abbas et al. [13,2]

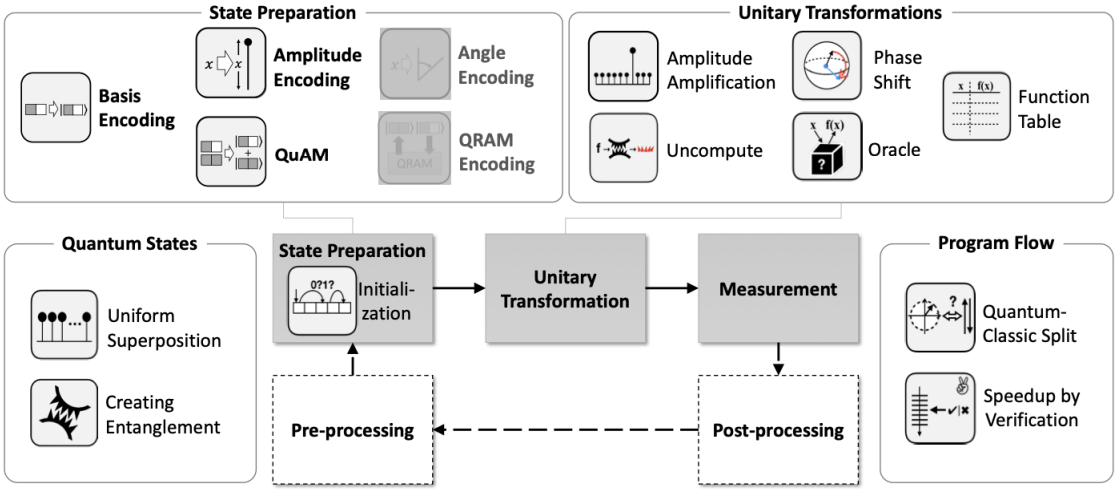

Figure 6: General overview of the quantum computing model

4.2.1 Quantum M-P Neural Network

The quantum M-P neural network is a quantum version of the traditional McCullough-Pitts model, where the output of the k-th neuron can be written as $O_k = \sum_j w_{kj}\phi_j, j = 1, 2, ..., 2^n$ and n is the number of required qubits.[24] For example, a two qubit quantum M-P model has the four possible inputs $|00\rangle, |01\rangle, |10\rangle, |11\rangle$, and its output can be written as $O_k = w_1\phi_{00}(a_1, a_2) + w_2\phi_{01}(a_1, a_2) + w_3\phi_{10}(a_1, a_2) + w_4\phi_{11}(a_1, a_2)$, where a_1, a_2 are the input qubits and $w_k = (w_{k1}, w_{k2}, ..., w_{kj})$ represents a vector.

If the model has orthogonal states, such as $|00\rangle$ and $|01\rangle$, their inner product vanishes. Rewriting our output equation in Dirac notation,[24,10]

$$O_k = \sum_j w_{kj} |a_1, a_2, ..., a_n\rangle, j = 1, 2, ..., 2^n$$

$$\text{i.e. } O_k = w_{k1} |0, 0, ..., 0\rangle + w_{k2} |0, 0, ..., 1\rangle + ...w_{k2^n} |1, 1, ..., 1\rangle$$

If the model has non-orthogonal states, such as $\alpha |0\rangle + \beta |1\rangle$ which is non-orthogonal with basis state $|0\rangle$ or $|1\rangle$, then the output is given by $O_{km} = \sum_j w_{kj}\phi_j \cdot \phi_m, j = 1, 2, ..., 2^n; m = 1, 2, ..., 2^n$, where $\phi_j \cdot \phi_m$ is the inner product of the two states.

Zhou and Ding also discuss a weight updating algorithm which can be used to train the network. [24] The weights of the network can be stored in a square matrix w. Then the relation of the input to the output can be defined as $|O\rangle = w |\phi\rangle$.

Weight Learning Algorithm

(a) Initialize a weight matrix w^0.

(b) Given a set of quantum examples (i.e. input-output pairs of $(|\phi\rangle, |O\rangle)$), calculate the output using $|\gamma\rangle = w^c |\phi\rangle$, where c is the number of iterations with initial value $c = 0$.

(c) Update weights with the formula $w_{kj}^{c+1} = w_{kj}^c + \tau(|O\rangle_k - |\gamma\rangle_k) |\phi\rangle_j$, where w_{kj} are matrix entries indexed by the row k and column j; τ is the learning rate.

(d) Repeat steps (b) and (c) until reaching acceptable errors.

The simple translation of the perceptron algorithm to the framework of quantum mechanics may seem like a good approach and serves as a useful example for introducing QNN model architectures. However, da Silva et al. have shown that the algorithm above does not follow unitary time evolution and that the proposed neuron can be efficiently simulated by a classical single layer neural network.[7]

4.3 QNN (Abbas et al. 2021)[2]

In a model analyzed by Abbas et al. (shown in Figure 7), "the input $x \in \mathbb{R}^{s_{in}}$ is encoded into an S-qubit Hilbert space by applying the feature map $|\psi_x\rangle := \mathcal{U}_x |0\rangle^{\otimes S}$. This state is then evolved via a variational form $|g_\theta(x)\rangle := \mathcal{G}_\theta |\psi_x\rangle$, where $\mathcal{G}$ is a parameterized unitary evolving the state after the feature map to a new state, and the parameters $\theta \in \Theta$ are chosen to minimize a certain loss function. Finally a measurement is performed whose outcome $z = (z_1, ..., z_s)$ is post-processed to extract the output of the model $y := f(z)$".[2] We use this network to this discuss the model complexity, expressibility, and training times of quantum neural networks as compared to their classical counterparts.

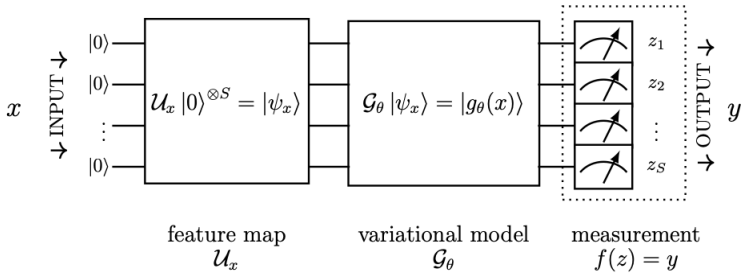

Figure 7: QNN model from Abbas et al.

4.4 Effective Dimension

The effective dimension is a model complexity measure inspired by information geometry and the theory of minimum description length, which is a model selection principle that favors models with the shortest description of the data. This idea is closely related to the Kolmogorov complexity of an object, a generalized measure of complexity defined by the length of the shortest computer program that produces the object as its output. According to Abbas et al., "the goal of the effective dimension is to estimate the size that a model occupies in model space – the space of all possible functions for a particular model class".[2]

A more commonly used model complexity metric is the Vapnik-Chervonenkis (VC) dimension. This metric quantifies the complexity or expressive power of a model by considering the cardinality of the largest set of points that the model can *shatter*, where a model is said to shatter a set of data points if there exists a set of parameters such that the model makes no classification errors on the points *for every assignment* of labels to points. The VC dimension is useful from a statistical theory perspective because one can derive probabilistic error bounds on how well a model generalizes on unseen data. However, it is difficult to compute in practice, because one must essentially show there exists a set of D points that is shattered by the model, but any set of points greater than D is not shattered by the model. Additionally, the VC dimension is criticized for requiring unrealistic assumptions for generating error bounds, scaling with the number of parameters in the model, and ignoring the distribution of the data. This means that generalization bounds for deep neural networks, which are highly overparameterized, are often quantitatively vacuous.

In a later paper, Abbas et al. compare several model complexity (or capacity) measures and analyze the following properties: existence of generalization bound, correlation generalization, scale invariance, data dependence, training dependence, ability to handle finite data, and efficient evaluation.[1] They argue that the effective dimension is a robust model complexity measure as it satisfies all of the aforementioned desirable properties.

4.4.1 Capacity and Trainability of Quantum vs Classical Neural Networks

According to experiments by Abbas et al. (summarized in Figure 8), the QNN "consistently achieved the highest effective dimension over all ranges of finite data".[2] The paper delves deeper into the mathematics of Fisher information matrices, which are used to estimate the effective dimension, and concludes that "QNNs can possess a desirable Fisher information spectrum that enables them to train faster and express more functions than comparable classical and quantum models".[2]

The prospect of higher capacity models with potentially faster training times is very exciting, even in the current NISQ (noisy intermediate-scale quantum) era, where leading quantum processors contain about 50 to a few hundred qubits but are not yet fully fault-tolerant and scalable. Network architectures that are qubit-efficient enable a reduction in the number of required coherent qubits, which increases the practicality of QNNs even in the NISQ era. Beer et al. mention that the decrease in number of qubits in their proposed model resulted in requiring multiple evaluations of the network to estimate the gradient of the cost function. However, many NISQ architectures are able to rapidly repeat executions of a quantum circuit. For example, the "Sycamore" quantum processor was able to execute one instance of a quantum circuit a million times in 200 s. The limiting factor which poses a real challenge is scaling the number of coherent qubits.[4]

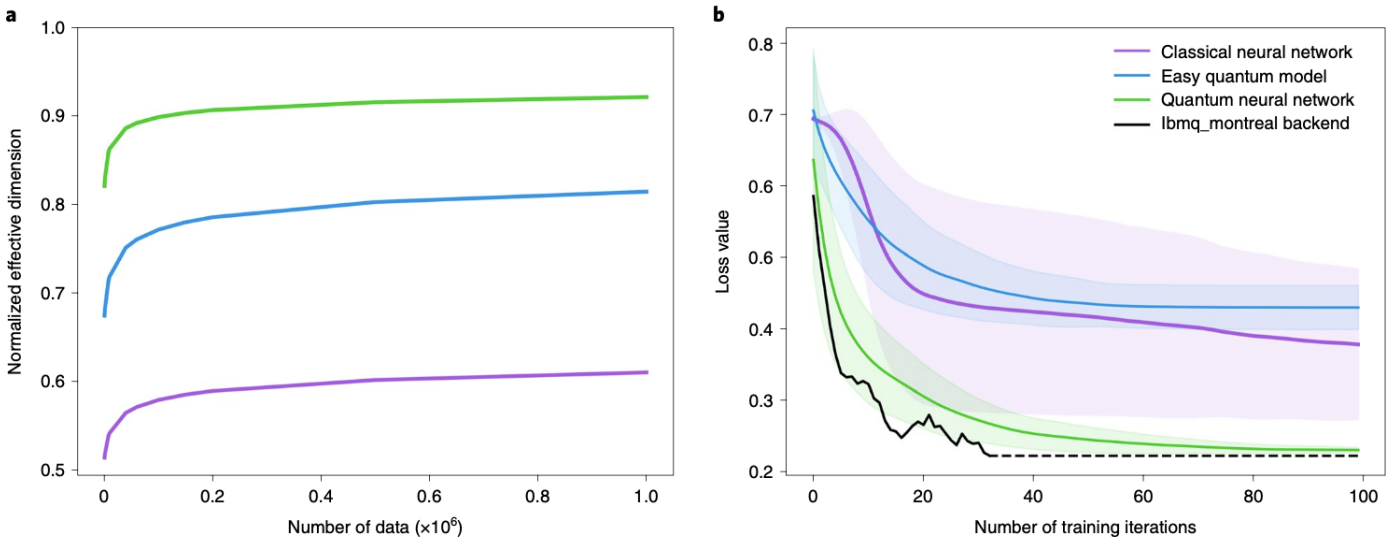

Figure 8: Normalized effective dimension and training loss of models from Abbas et al.

4.5 Challenges of QNN

A significant challenge in training quantum neural networks is the barren plateau phenomenon. This happens when the loss landscape or parameter space is extremely flat, resulting in a vanishing gradient and making parameter optimization very difficult. As shown by Wang et al., barren plateaus can be noise-induced, meaning the noise from the quantum hardware can cause the training landscape to have a barren plateau.[20] Another source of barren plateaus is the circuit design itself, specifically the random parameter initialization component. While there have been some proposed solutions for avoiding circuit-induced barren plateaus, noise-induced barren plateaus are an area requiring further research.[2] Interestingly, according to the experiments discussed above, the data encoding strategy in a quantum neural network affects the likelihood of the model encountering a barren plateau – encoding strategies that are easy to simulate classically are more likely to lead to barren plateaus, while harder encoding strategies are more likely to avoid the phenomenon.[2]

5 Applications of QNN

Given that quantum computing and quantum machine learning are still in early stages of research, applications of QNNs primarily focus on demonstrating feasibility rather than showing significant improvements in accuracy compared to classical models. Even still, QNNs have been applied to a wide range of problems, ranging from time series foresting of financial data to analyzing real-life data from industrial machines.[12,14] In this section, we review applications of QNNs for breast cancer prediction and biosignal processing. The practicality of applying a quantum model over a classical model is discussed in the following section.

5.1 Breast Cancer Prediction

Ali et al. apply a QNN for early-stage breast cancer detection and compare its performance to a classical convolutional neural network. The QNN was built using Google's Cirq software framework for simulating quantum circuits. Due to the limited number of qubits, the original 1080 by 1080 mammogram images were downsampled to 50 by 50 pixels for both the QNN and the classical CNN. The networks were trained on a set of 1200 images with a validation set of 300 images.

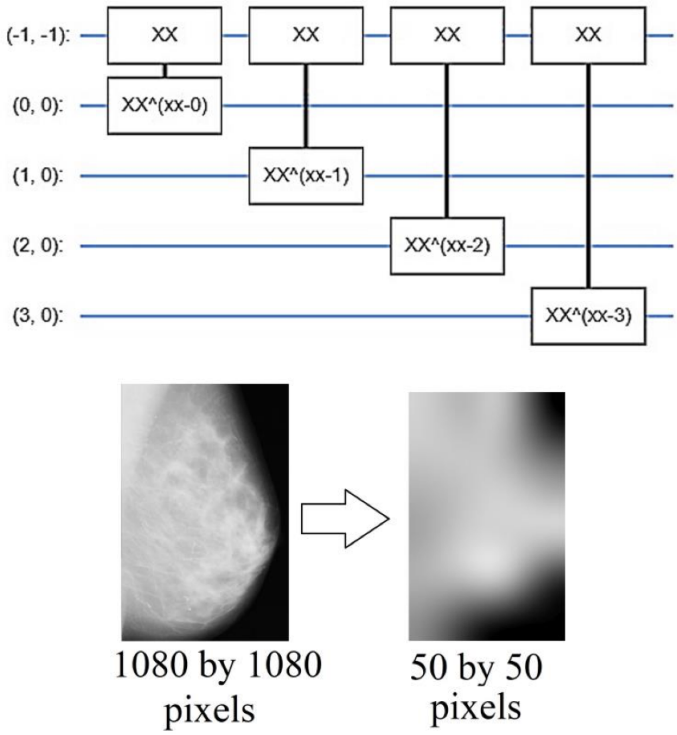

Figure 9: Top – Ising quantum circuit for mammogram image-based cancer detection from Ali et al.; Bottom – Compressed image representation before training.

Figure 10 demonstrates the result of training both networks on the 1200 image dataset. The QNN converges much faster and achieves a higher accuracy on the validation set than the classical CNN. However, it is more interesting that the QNN is able to achieve such performance from relatively few training examples. Ali et al. initially performed the same experiment with a training set of 600 images, but the classical neural network seemed to underfit the data and was unable to converge in its validation curve. The results of Ali et al. demonstrate that the prospect of larger scale quantum computers "make QNNs an anticipated tool for performing mass medical imaging diagnostics with a low false-negative rate".[3]

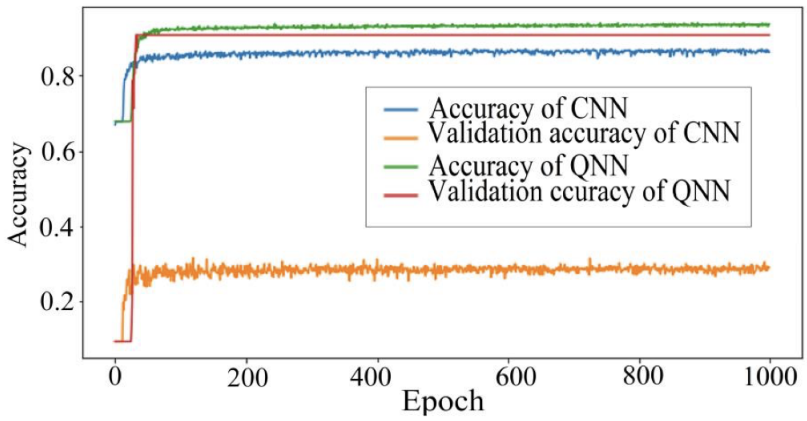

Figure 10: Classical CNN and QNN training accuracy.

5.2 Hybrid Quantum-Classical Model for Biosignal Processing

Akino and Wang propose a hybrid quantum-classical neural network model that integrates a variational quantum circuit (VQC) into a deep neural network (DNN) for electroencephalogram (EEG), electromyogram (EMG), and electrocorticogram (ECoG) analysis.[11] The system takes amplitude-encoded biological waveform arrays as input and predicts a task label on various physiological datasets. The parameters of the QNN and DNN are jointly optimized by leveraging classical stochastic gradient methods. As shown in Figure 11, the VQC component consists of Pauli-Y rotations and staggered controlled Pauli-Z gates. Note that the Pauli-X gate behaves like a classical NOT gate, resulting in a single bit flip. The Pauli-Z gate implements a phase flip operation by performing a single-qubit rotation around the z axis by π radians. The Pauli-Y gate performs both a bit flip and a phase flip by rotating around the y axis by π radians. The QNN component of the hybrid model performs feature extraction while the classical component works as the post-processing layers. The model was implemented using the PennyLane and PyTorch frameworks.

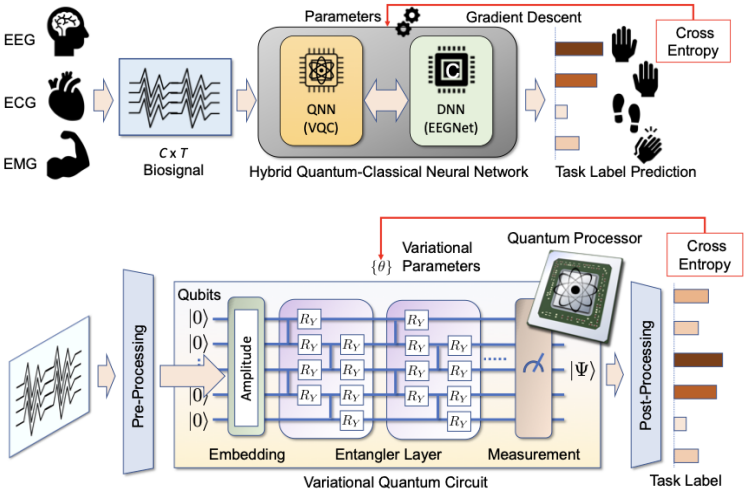

Figure 11: Top – Hybrid quantum-classical neural networks for biosignal processing; Bottom – Variational QNN component.

Figure 12 summarizes the model performance of the classical and hybrid models – the hybrid quantum-classical model outperformed the purely classical model on all seven physiological datasets.[11] This paper provides another successful proof-of-concept for future applications of QNNs.

PERFORMANCE RESULTS IN TEST ACCURACY (%)

Dataset	EEGNet	quEEGNet
Stress	85.87	87.23
RSVP	93.73	95.12
MI	59.61	60.22
ErrP	74.36	75.92
Faces Basic	63.30	64.92
Faces Noisy	75.94	78.01
ASL	23.64	25.16

Figure 12: Classical and hybrid model test accuracy.

6 Quantum Advantage

Quantum neural networks, similar to their classical counterparts, are essentially universal function approximators. Salinas et al. actually proved that the Universal Approximation Theorem holds for QNNs as well.[15] Therefore, assuming the existence of an appropriate data encoding and availability of sufficient qubits, QNNs can be used to solve any problems that classical neural networks can solve. The more interesting question is what advantages, if any, do QNNs provide over classical neural networks. As discussed in previous sections, there is some evidence that QNNs can both train to lower losses faster and can have higher effective dimensions.[2] The idea of a quantum advantage is still an active area of research, but assuming future improvements to quantum hardware, QNNs may one day become another tool in the data scientist's toolset.

The practicality of using a quantum model over a classical model will depend on the cost and availability of quantum computers, as well as the potential room for improvement that a quantum model may provide. For example, although we may prove that quantum models are able to approximate more functions and are therefore a more powerful class of models, is applying a QNN to a problem where the leading classical model already achieves 98% accuracy practical? This cost-benefit analysis is highly context-dependent. If we are examining a machine learning model for noise-cancellation headphones, an incremental increase in effectiveness above 98% will probably not be very noticeable and therefore not practical. However, if we consider models for early-stage medical diagnosis, even a fractional percent of improvement can impact thousands of lives when deployed at scale. Another example is model lift – consider a company like Meta conducting an ad campaign. Using similar reasoning, a new model that improves the lift [2] even by less than a percent can have significant impact on revenue for a company operating at Meta's scale.

Although these cases may be of interest from a theoretical perspective, quantum models will need use cases that are both cost efficient and clearly advantageous to achieve wider adoption in industry. In a recent paper, Caro et al. proved impressive generalization bounds for QML – the generalization error of a QML model with T trainable gates scales at worst $\sqrt{T/N}$, and improves to $\sqrt{K/N}$ when only $K \ll T$ gates have undergone substantial change in the optimization process.[6] This result affirms our observations from the breast cancer prediction model, where the QNN model was able to converge from a much smaller set of training data. According to Caro et al., "QML models can outperform classical methods, assuming both achieve small training error, only in scenarios in which QML models generalize well, but classical ML methods do not. We therefore consider our results a guide in the search for quantum advantage of QML: We need to identify a task in which QML models with few trainable gates achieve small training error, but classical models need substantially higher model complexity to achieve the same goal. Then, our bounds guarantee that the QML model performs well also on unseen data, but we expect the classical model to generalize poorly due to the high model complexity".[6]

[2] Lift is a measure of the effectiveness of a predictive model calculated as the ratio between the results obtained with and without the predictive model.

7 Future Directions

Future research directions in quantum machine learning and quantum computing as a whole can be divided into near-term and long-term categories as they represent different challenges. In the near-term NISQ era, we have noisy, small-scale quantum computers. The most important problem we should solve is the input problem – specifically, we need to build a qRAM. As discussed previously, many of the quantum algorithms that promise an exponential speed-up rely on qRAM, but currently the idea of a quantum memory is purely theoretical. Furthermore, we should continue research into finding more efficient (linear or logarithmic time) state preparation routines to ensure that the quantum speed-up is not dominated by the inordinate cost of encoding classical data into quantum states. Next, we should do more research into barren plateaus – particularly noise-induced barren plateaus – and methods of avoiding them during parameter optimization.

Long-term problems in quantum computing pose more significant challenges – namely, scaling up the number qubits while maintaining coherence. The more interacting quantum subsystems, the more difficult it is for the system to stay resilient to noise and decoherence. Additionally, while small-scale NISQ devices may not require quantum error correction, scalable quantum computers will need further research into error correction to address the problems mentioned above. The current state-of-the-art quantum processor, announced in November 2022, is IBM's 433-qubit Osprey quantum computer. IBM claims to be on track to announce the 1,121-qubit Condor and 1,386-qubit Flamingo processors in 2023 and 2024, before it hits the 4,000-qubit stage with its Kookaburra processor in 2025.

8 Discussion

In this paper, we provided an introduction to the growing field of quantum machine learning and a powerful class of variational models called quantum neural networks. The motivation for applying quantum computing to machine learning problems comes from the fact that both quantum mechanics and machine learning algorithms involve performing linear algebraic operations on high-dimensional vectors. Given a qRAM, we will be able to achieve exponential compression in the representation of classical data, leading to exponentially faster execution of the basic linear algebra subroutines. Thus, in the long term, scalable quantum computers may be able to solve classically intractable problems. While the results discussed in this paper are active areas of research and require further research, current evidence suggests quantum models may be more expressive, consistently having a higher effective dimension in experiments compared to their classical counterparts. Additionally, we have evidence that quantum models may be able to learn from fewer training examples while maintaining high generalizability as validated both experimentally and by theoretical upper bounds on generalization error. This property of learning from fewer training examples may be useful in cases where data collection is expensive or difficult, resulting in limited access to training data. In addressing these challenges, we hope to build novel quantum computers and models capable of solving problems that are currently considered computationally intractable, which will undoubtedly have far-reaching impact across all disciplines.

References

[1] Abbas, A., Sutter, D., Figalli, A., & Woerner, S. (2021). Effective dimension of machine learning models. https://doi.org/https://doi.org/10.48550/arXiv.2112.04807

[2] Abbas, A., Sutter, D., Zoufal, C., Lucchi, A., Figalli, A., & Woerner, S. (2021). The power of Quantum Neural Networks. *Nature Computational Science, 1*(6), 403–409. https://doi.org/10.1038/s43588-021-00084-1

[3] Al Ali, M., Sahib, A., & Al Ali, M. (2022). Investigation of early-stage breast cancer detection using Quantum Neural Network. https://doi.org/10.20944/preprints202210.0208.v1

[4] Beer, K., Bondarenko, D., Farrelly, T., Osborne, T. J., Salzmann, R., Scheiermann, D., & Wolf, R. (2020). Training Deep Quantum Neural Networks. *Nature Communications, 11*(1). https://doi.org/10.1038/s41467-020-14454-2

[5] Benedetti, M., Realpe-Gómez, J., Biswas, R., & Perdomo-Ortiz, A. (2017). Quantum-assisted learning of hardware-embedded probabilistic graphical models. *Physical Review X, 7*(4). https://doi.org/10.1103/physrevx.7.041052

[6] Caro, M. C., Huang, H.-Y., Cerezo, M., Sharma, K., Sornborger, A., Cincio, L., & Coles, P. J. (2022). Generalization in quantum machine learning from few training data. *Nature Communications, 13*(1). https://doi.org/10.1038/s41467-022-32550-3

[7] da Silva, A. J., de Oliveira, W. R., & Ludermir, T. B. (2014). Comments on "Quantum M-P neural network." *International Journal of Theoretical Physics, 54*(6), 1878–1881. https://doi.org/10.1007/s10773-014-2393-1

[8] Farhi, E., & Neven, H. (2020). Classification with quantum neural networks on near term processors. https://doi.org/10.37686/qrl.v1i2.80

[9] Harrow, A. W., Hassidim, A., & Lloyd, S. (2009). Quantum algorithm for linear systems of equations. *Physical Review Letters, 103*(15). https://doi.org/10.1103/physrevlett.103.150502

[10] Jeswal, S. K., & Chakraverty, S. (2018). Recent developments and applications in Quantum Neural Network: A Review. *Archives of Computational Methods in Engineering, 26*(4), 793–807. https://doi.org/10.1007/s11831-018-9269-0

[11] Koike-Akino, T., & Wang, Y. (2022). QuEEGNet: Quantum AI for Biosignal processing. *2022 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*. https://doi.org/10.1109/bhi56158.2022.9926814

[12] Mangini, S., Marruzzo, A., Piantanida, M., Gerace, D., Bajoni, D., & Macchiavello, C. (2022). Quantum neural network autoencoder and classifier applied to an industrial case study. *Quantum Machine Intelligence, 4*(2). https://doi.org/10.1007/s42484-022-00070-4

[13] McCulloch, W., & Pitts, W. (2021). A logical calculus of the ideas immanent in nervous activity (1943). Ideas That Created the Future, 79–88. https://doi.org/10.7551/mitpress/12274.003.0011

[14] Paquet, E., & Soleymani, F. (2022). Quantumleap: Hybrid quantum neural network for financial predictions. *Expert Systems with Applications, 195*, 116583. https://doi.org/10.1016/j.eswa.2022.116583

[15] Pérez-Salinas, A., Cervera-Lierta, A., Gil-Fuster, E., & Latorre, J. I. (2020). Data re-uploading for a universal quantum classifier. *Quantum, 4*, 226. https://doi.org/10.22331/q-2020-02-06-226

[16] *Quantum embedding*. Quantum embedding - PennyLane documentation. (n.d.). Retrieved December 8, 2022, from https://pennylane.ai/qml/glossary/quantum_embedding.html

[17] Schuld, M., & Petruccione, F. (2018). *Supervised learning with Quantum Computers*. Springer.

[18] Schuld, M., Bocharov, A., Svore, K. M., & Wiebe, N. (2020). Circuit-centric quantum classifiers. *Physical Review A, 101*(3). https://doi.org/10.1103/physreva.101.032308

[19] Shah, F. (2021, December 18). *Quantum Encoding: An overview*. Quantum Zeitgeist. Retrieved December 8, 2022, from https://quantumzeitgeist.com/quantum-encoding-an-overview/

[20] Wang, S., Fontana, E., Cerezo, M., Sharma, K., Sone, A., Cincio, L., & Coles, P. J. (2021). Noise-induced barren plateaus in variational quantum algorithms. *Nature Communications, 12*(1). https://doi.org/10.1038/s41467-021-27045-6

[21] Weigold, M., Barzen, J., Leymann, F., & Salm, M. (2021). Expanding data encoding patterns for quantum algorithms. *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*. https://doi.org/10.1109/icsa-c52384.2021.00025

[22] Weigold, M., Barzen, J., Leymann, F., & Salm, M. (2022). Data encoding patterns for quantum computing. In Proceedings of the 27th Conference on Pattern Languages of Programs (PLoP '20). https://doi.org/10.5555/3511065.3511068

[23] Wootters, W. K., & Zurek, W. H. (1982). A single quantum cannot be cloned. *Nature, 299*(5886), 802–803. https://doi.org/10.1038/299802a0

[24] Zhou, R., & Ding, Q. (2007). Quantum M-P Neural Network. *International Journal of Theoretical Physics, 46*(12), 3209–3215. https://doi.org/10.1007/s10773-007-9437-8