# Quantum Machine Learning Review and Technical Demonstration

Milan Leonard u6661755

2020

**Abstract**

Quantum Computing and Machine Learning promise to provide solutions to problems that were previously intractable. However, quantum algorithm development is a severe challenge, with many algorithms requiring quantum technologies that are decades away, similarly, classical machine learning tasks are approaching the boundary of computational limits, with computational demands exhibiting exponential growth. In this paper, I will explore the interdisciplinary field of Quantum Machine Learning, and its promises to solve both of these problems, with the purpose of identifying fertile areas for future research. Furthermore, I demonstrate a personal implementation of a specific recent quantum machine learning technique, which illuminates the key technical decisions and considerations that are necessary when designing these algorithms. I conclude that the most exciting near-term applications in Quantum Machine Learning fall within the class of Variational Hybrid Quantum-Classical algorithms, and that the assumptions required in many hand-crafted algorithms are outside of the reach of near-term quantum technologies. For quantum computers to solve real problems leveraging quantum advantage in the near term, we will need to understand how to create and assess quantum machine learning algorithms.[1]

# Contents



[1] Code can be found at `https://github.com/milanleonard/QNN-implementation`

# 1 Introduction

Quantum Computing and Machine Learning are two of the most exciting technologies of the early 21st century. Quantum computers, in theory, allow for exponentially better use of computational resources, whilst concurrently providing an exponential speed-up in the ability to perform certain computations [1]. In October 2019, researchers at Google claimed to have achieved quantum supremacy [2]; where a quantum computing device is able to perform a computation task that would be intractable (due to time or memory constraints) on a classical computer. Despite the fact that this claim was countered by other members of the community, it represented a major milestone for the field. However, as is true for many of the quantum algorithms that are able to be performed on existing quantum computers, the algorithm has no known practical real-world applications. The lack of a known application, and the conflicting opinions are, taken together, are indicative of the difficulties in the development and assessment of novel quantum algorithms..

Machine learning is, in its broadest definition, a set of methodologies for autonomously creating algorithms by performing computation on pre-existing or generated data [3]. The empirical utility of machine learning is becoming more established with its ability to perform novel, previously impossible tasks, such as computer vision and speech recognition, highly useful for industry and private problems, to quantum state preparation and solving many-body quantum simulations. Machine learning tasks out-perform classical algorithms on all of these tasks. One particular machine learning algorithm, the neural network (and more broadly, deep learning) has demonstrated an ability to perform well across domains, and to scale well with the amount of training data [4]. Some insight into the essential cause for their success can be gleaned from the universal approximation theorem [5]. In essence, this theorem states that a neural network of sufficient depth with sufficient non-linearity can approximate any continuous function with equivalent domain and codomain. However, the computational resources involved in training machine learning models are increasing as the neural network architectures become deeper and more complicated, and as the world collects more data. To exacerbate this problem, simply increasing the scale of pre-existing architectures and collecting more data, while training for longer is a proven method for

improving the success of many of these models. This computational bottleneck in the ability to quickly iterate various models make machine learning a great target area for quantum computing.

Quantum Machine Learning (QML) is the emerging inter-disciplinary field of using quantum computing to perform machine learning tasks and of using quantum computers as learners [6]. This field is broadly interested in solving the mentioned problems of the difficulty of algorithm development and the computational resource requirement of classical machine learning. In fact, there are competing definitions for the role of QML [6][7], with reviews of the field disagreeing on even the most basic definitions. The possibility of automatically producing useful quantum algorithms that could provide exponential speed-up on a vast array of problem domains without requiring a specific moment of ideation, and requiring domain specialization, is a tantalizing prospect. Equally tantalizing, is the possibility for a dramatic reduction in costs for training the existing landscape of machine learning algorithms, or for allowing these algorithms to exist at a scale previously impossible.

The aim of this paper is to disentangle the confusion, to critically review the field as it stands, and to make an assessment of the applications of quantum machine learning, and to orient towards prosperous near-term research directions. Before analyzing the field in specifics, section 2 aims to bring a machine learning expert up to speed in the relevant aspects of quantum computing, and to bring a quantum researcher up to speed in machine learning. Then, partly due to the current lack of a standard view of the field by experts, a deep understanding of the current questions that must be answered and landscape of current algorithms will be explored in section 3. Furthermore, this paper will explore, in section 4, the formalism of a specific algorithm that attempts to enable quantum computer vision. In section 5, I explore my instance of this formalism, and discuss results and ways forward. In doing so, it should provide a lens and methodology through which to critically assess papers in this field, and demonstrate how the practical considerations must be taken into account when attempting to produce new work in this field. Finally, the key considerations, opportunities, and a path forward will be synthesised.

# 2 Background

Due to the interdisciplinary nature of Quantum Machine Learning, an understanding of both Quantum Computing and classical Machine Learning is required to assess the field. The purpose of this section is to bring a machine learning researcher up to speed in the relevant quantum computing domains, and vice versa.

## 2.1 Quantum Computing

The fundamental principle surrounding current attempts at producing quantum computers is to use leverage superposition and entanglement of coupled 2-qubit systems to perform information processing. There are many different ways to physically build 2-qubit systems, each of which come with its unique set of advantages and disadvantages. For example, the polarization state of a photon or the spin state of an atom. Much of the discussion around quantum algorithm development will be informed by the specific requirements of the hard-

ware. Since the mathematical description of all two-level quantum systems is identical, the following discussion will define an orthonormal basis as $|0\rangle$ and $|1\rangle$. A linear algebra representation of the Dirac notation I will commonly be used is, choosing what will be defined as the *computational basis*, the state $|0\rangle = (1,0)^T$, and therefore by orthonormality $|1\rangle = (0,1)^T$. The following is relevant material from Nielsen and Chuang's Quantum Computation and Quantum Information [1].

A qubit is defined by the state of a single two-level system. Clearly, a qubit, like a traditional bit, could be in either $|0\rangle$ or $|1\rangle$. However, due to superposition of quantum states, the state of a qubit is most generally defined (with respect to the previously defined basis) as

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

where $\alpha, \beta$ are complex numbers. The probability of measuring $|0\rangle$ is $|\alpha|^2$ and of measuring $|1\rangle$ is $|\beta|^2$. Since these are complex values, this general state is parameterized by 4 scalars $(\alpha_{real}, \alpha_{im}, \beta_{real}, \beta_{im})$. However, it is possible to remove two of these parameters. First, the global phase has zero physical effect (i.e. has no effect on the outcome of any real measurements), and as such it is possible (through arbitrary convention) to define that $\alpha_{im} = 0$ and to absorb the relative phase into $\beta$. Furthermore, the total state $|\Psi\rangle$ has to be normalized, meaning that $|\alpha|^2 + |\beta|^2 = 1$. This means that it is possible to reduce to two parameters $(\theta, \phi)$, and re-write the general state vector as

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

Interpreting $(\theta, \phi)$ as a polar and azimuthal angle, this can be interpreted as a point on a sphere. This sphere is called the Bloch sphere, and allows for a useful geometric interpretation as in Fig. 1. This makes clear one of the quantum advantages, which is a linear increase in encoding density. This is due to the continuous nature of complex numbers, such that you can encode any real number into the amplitude, alongside a scaling factor to put it into the right units. Concretely, in order to store an 8-bit unsigned integer $\lambda$, which can take on values from 0-255, a requirement of 8-bits is required. However, an amplitude encoding could write this into a single qubit as $|\lambda\rangle = \sqrt{\lambda}\,|0\rangle + \sqrt{1-\lambda}\,|1\rangle$ such that the probability of measuring $|0\rangle$ is $\lambda$.

Transitioning from single qubits to multiple qubits follows from the tensor product of many two-level qubit systems. Moving from a single qubit to two qubits, the possible state of the two-qubits are given by the tensor product of the basis states as in equation (3)

$$\{|0\rangle\,, |1\rangle\} \otimes \{|0\rangle\,, |1\rangle\} = \{|0,0\rangle\,, |0,1\rangle\,, |1,0\rangle\,, |1,1\rangle\}. \tag{3}$$

An arbitrary 2-qubit state is given by $|\Psi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle$. Noticing that in order to specify a quantum state there is a complex number for every possible entry in the basis, and that the basis is given by every possible combination of 0 and 1 inside of a ket, to specify an n-qubit system requires $2^n$ complex numbers. If it is possible to set those complex numbers, the *amplitudes* of the quantum state, to any desired value, then it is possible to encode exponentially more information with the same number of (qu)bits.

Broadly, there are two known methods for performing computation using qubits. Firstly, through quantum annealing, which exploits the adiabatic theorem to solve optimization

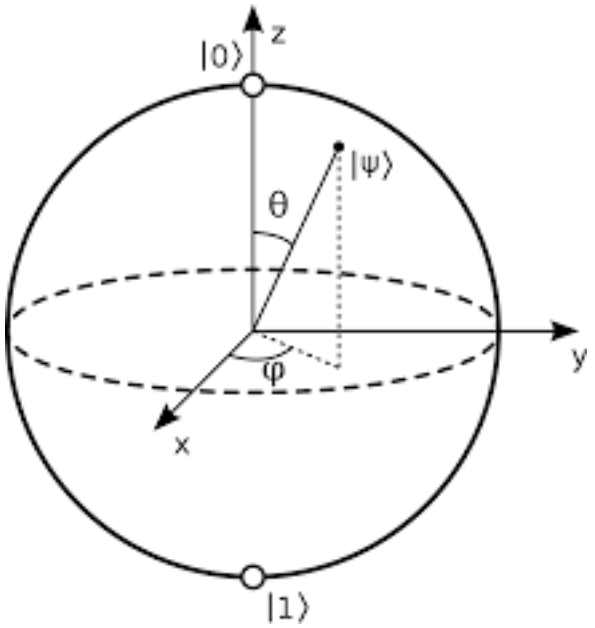


Figure 1: The poles of the sphere represent the basis states, and the quantum superposition states can exist anywhere on the sphere. This represents all of the extra information that quantum bits can hold. Image source [1]

problems. It has been shown that this is not a Universal Turing Machine, and is therefore not able to perform any possible quantum computation. Due to the desire to produce new algorithms, or to optimize arbitrary algorithms, this paper will focus its discussions on gate based quantum computation.

Gate based quantum computation is the closest analogue to classical computation in that it uses a set of possible logic operations to transforms qubits. There are two classes of gates, single qubit gates, and multiple qubits gates. Some of the most useful single qubit gates, which appear in a vast majority of known algorithms, are the Hadamard gate H, the Pauli X, Y, and Z gate, and a continuous rotation $R_\phi$ gate around an axis (in the below case, the z-axis of the Bloch Sphere). Since these quantum gates are all implemented as the action of a Hamiltonian over some time, they correspond to time evolution under a unitary operator, and as such, are all unitary. These gates, in the $\{|0\rangle, |1\rangle\}$ basis, are written in matrix form

as

$$\hat{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{4}$$

$$\hat{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{5}$$

$$\hat{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{6}$$

$$\hat{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{7}$$

$$\hat{R}_\phi = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}. \tag{8}$$

Common multi-qubit operations are the controlled-not, swap, and Toffoli gates. Of particular interest is the 3-qubit operation, the Toffoli gate, since it can be shown quite simply that using only the Pauli gates, the Hadamard, and the Toffoli gate can perform any computation [8], making quantum computers universal Turing machines; where any classical computation can be performed on a quantum computer.

Any arbitrary quantum computation can be written as a unitary operator, by multiplying the chain of operators that compose the computation. For example, an algorithm $\hat{A}$ characterised by applying Hadamard to the first qubit and an X gate to the third qubit, followed by a Y gate to the first qubit could be written as the matrix product

$$(\hat{Y} \otimes \hat{I}_2^{\otimes 2})(\hat{H} \otimes \hat{I}_2 \otimes \hat{H}) \tag{9}$$

An algorithm written in this format is hard to decipher, and hard to analyse. Instead, it it better written using quantum circuit notation as in Fig. 2. Notice that the order seems to be 'flipped' relative to (9), since matrix multiplication is applied right-to-left on an input vector. Conversely, the figure is read left to right, applying the operations in the given sequence where required. If no computation occurs on the qubit in that step, this means that there is an identity gate $I$ in the slot

Circuit notation allows the introduction of the second main category of quantum advantage, quantum parallelism. Quantum parallelism is the ability for a quantum computer to apply a function concurrently to many states at once. A specific example of this is the **Deutsch's algorithm**.

Deutsch's algorithm attempts to determine whether a function $f : \{0, 1\} \to \{0, 1\}$ is either constant or balanced. If $f$ is constant, then $f(0) = f(1)$. If $f$ is balanced, then $f(0) \neq f(1)$. Clearly, to determine this classically this requires computation of both $f(0)$ and $f(1)$. On a quantum computer, this requires only one application of a circuit performing $f$. This procedure is done by the circuit shown in Fig. 3. Here $U_f$ is an *oracle* which maps $|x\rangle |y\rangle \to |x\rangle |f(x) \oplus y\rangle$ where $\oplus$ is additon modulo 2.

To show that this implements the desired algorithm, the calculations are readily per-

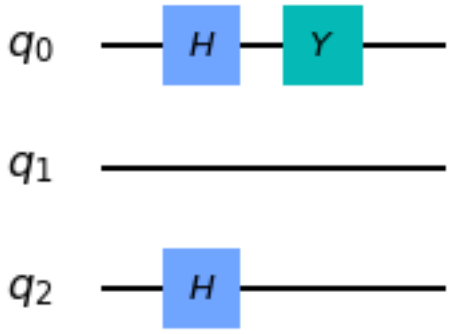


Figure 2: Quantum circuit diagrams such as this one are an efficient way to write quantum algorithms

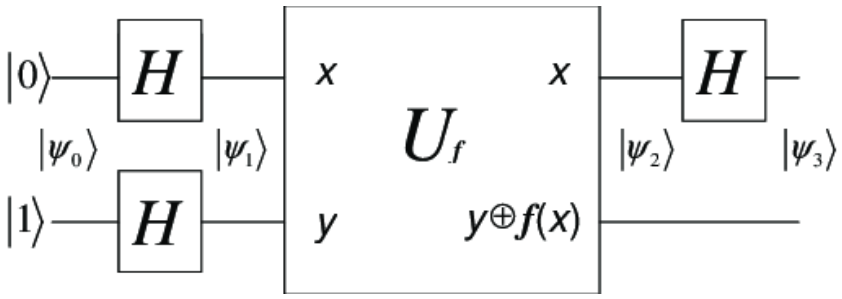


Figure 3: A circuit implementing Deutch's algorithm.

formed

$$
\begin{aligned}
|\psi_0\rangle &= |0\rangle\,|1\rangle \\
(\hat{H}\otimes\hat{H})\,|\psi_0\rangle &= \frac{1}{2}(|0\rangle+|1\rangle)(|0\rangle-|1\rangle) = |\psi_2\rangle \\
\hat{U}_f\,|\psi_2\rangle &= \frac{1}{2}\,|0\rangle\,(|f(0)\oplus 0\rangle - |f(0)\oplus 1\rangle) + |1\rangle\,(|f(1)\oplus 0\rangle - |f(1)\oplus 1\rangle) \\
&= (-1)^{f(0)}\frac{1}{2}(|0\rangle + (-1)^{f(0)\oplus f(1)}\,|1\rangle)(|0\rangle-|1\rangle) = |\psi_2\rangle\,(|0\rangle-|1\rangle).
\end{aligned}
$$

Since we will only measure the state of qubit 1, we can ignore the second qubit and focus solely on $|\psi_2\rangle$

$$
\hat{H}\,|\psi_2\rangle = \frac{1}{2}((1+(-1)^{f(0)\oplus f(1)})\,|0\rangle + (1-(-1)^{f(0)\oplus f(1)})\,|1\rangle) \tag{10}
$$

Analysing equation (10), we notice that if $f(0) = f(1) \implies f(0)\oplus f(1) = 0$ then the probability of finding it the qubit in state $|0\rangle$ is 100% and as such the function is constant,

and similarly to show that if it is in state $|1\rangle$ then the function is balanced. Notice that the question at hand has been solved with fewer evaluations of the function $f$ than in the classical state, at the cost of not knowing the actual identity of the function.

Quantum algorithms in general, unlike in Deutsch's algorithm, do not allow for a 'one shot' computation. Instead, since the results of measurement in quantum mechanics are, unlike classical determinism, probabilistic and non-deterministic, many shots must be performed in order build up a statistical picture of the final distribution over the states. This places another restriction on algorithm design, in that sampling the desired answer must be efficient. For example, if the information is spread across all of the qubits, as is the case in the quantum Fourier transform, then a very large number of computational shots must be performed. Comparatively, Deutsch's algorithm represents the ideal case, where there is a 100% probability to be in particular states at the end of the algorithm.

Quantum advantage is generated therefore in two shown ways. Firstly, an exponentially more dense encoding of classical information through amplitude encoding, and secondly, through quantum parallelism performing computation on many inputs simultaneously. However, there are significant limitations to our current ability to leverage these effects. The following limitations will define the characteristic devices and quantum algorithms that are technologically feasible in the near future, the 'noisy intermediate scale quantum' (NISQ) era. The first major limitation is in the decoherence time of a quantum system. Since there will inevitably be some noise decohering the desired quantum state, this places a time limit on the length of computation before the probability of obtaining the correct result becomes tiny. This places a limit on both gate depth and gate complexity, and signifies an important consideration in the construction of quantum-amenable hardware. Since it is difficult, in general, to be able to perform any desired gate using a single control loop, more complicated quantum gates need to be decomposed into ones that are simpler to execute. This can dramatically increase the depth of proposed algorithms since the decomposition can be expensive. Secondly, hardware devices need to be able to couple arbitrary qubits, but in practise this is difficult. Instead, there are tricks to imitate qubit coupling through exploiting other qubits as an intermediary couple (for example, A is coupled to B, and B is coupled to C), however this further increases gate depth. This means that algorithms suitable for the NISQ era need to be simple and short. Thirdly, despite the theoretical exponentially dense encoding there is no known method of efficiently preparing an arbitrary state $|\Psi\rangle$. The preparation and storage of easily retrievable quantum states is known as qRAM, and is an unsolved problem. Therefore, algorithms that make use of an exponentially dense amplitude encoding, such as the quantum Fourier transform for arbitrary input data, cannot be implemented on quantum hardware. As such, there are a series of compromises to be made in the design choices of a quantum algorithm that limit its potential in favour of being able to implement it in now or in the near future.

Encoding a quantum state is one place where these choices become important. As demonstrated, an *amplitude encoding* encodes classical data $(x_1, ..., x_{2^n})$ into the amplitudes of n qubits. However, since a feasible method for doing so is currently unknown, requiring this for an algorithm means that it may not find use in the near future. Alternatively, *basis encoding* is the current most feasible approach. This only achieves a linear resource efficiency, which scales with the number of bits required to store the traditional piece of data. However, a continuous basis encoding, which achieves this linear increase, comes at the expense

of increased gate depth, and since arbitrary rotations in general require decomposition to produce the correct state, exacerbated by the fact that rotating more precisely will require even more decomposition, then the linear increase in resource efficiency can be outweighed by the increase in gate depth, and as such shorter useful time to implement an algorithm. Then, there is a binary encoding, which will be encountered later, which encodes an arbitrary bit-string into its quantum representation. This is trivial to implement, by starting with the $|0...0\rangle$ state, and applying the $\hat{X}$ gate to the qubits which should be in the $|1\rangle$ state. These encoding schemes are shown in Table 2.1, noting that the amplitude encoding assumes a normalized vector $||(y_1, ...y_n)|| = 1$. Lastly, there is Hamiltonian encoding, in which case a quantum annealing approach is most suitable. However, this cannot store most types of classical data.

| Encoding type | Data | Quantum State |
|---|---|---|
| Binary basis encoding | Bitstring $x_1...x_n$, $x_i \in \{0, 1\}$ | $\|x1...xn\rangle$ |
| Continuous encoding | N-bit numbers $y_1, ..., y_n$ | $\frac{1}{2^N} \bigotimes_i^n y_i \|0\rangle + \sqrt{(1 - y_i^2)} \|1\rangle$ |
| Amplitude encoding | Normalized vector of N-bit numbers $y_1, ..., y_n$ | $\sum_i^n y_i \|y_i\rangle$ where $\|y_i\rangle$ is the $\mathrm{i}^{\mathrm{th}}$ computational basis state |

All of these considerations, alongside a shortage of people with backgrounds in both quantum mechanics and computer science, makes the ideation process of quantum algorithms extremely difficult. Therefore, taking an automated approach to algorithm development is exciting.

## 2.2 Machine Learning

Machine learning is the field of trying to autonomously recognize useful patterns through data. Broadly, there are three methods for approaching machine learning tasks. They are supervised learning, unsupervised learning, and reinforcement learning. Whilst quantum reinforcement learning research is rapidly progressing [9], the goal of reinforcement learning is most often distinct from other machine learning methods, and as such is not discussed here.

(a) *Supervised Machine Learning*
Supervised machine learning is the process of optimizing a model using labelled examples to improve a performance metric. For example, if there was a two class (for example, the canonical cats and dogs) classifier $h : \mathbb{R}^{N \times M} \times \mathbb{R}^W \to \mathbb{R}^2$, where the input to h was an image, and the function h is parameterised by a weight vector $\mathbf{w}$ of dimension $W$, and the output is interpreted as $\begin{bmatrix} prob(class1) \\ prob(class2) \end{bmatrix}$ then a possible loss function for a single example might be

$$\mathcal{L}(image, \mathbf{target}, \mathbf{w}) = ||\mathbf{h}(image, \mathbf{w}) - \ \mathbf{target}||^2 \tag{11}$$

where equation (11) represents the squared error. Then, by iteratively using new training samples and updating the weight as in equation (12), this problem can be

seen purely as a gradient descent problem with step size (*learning rate*) $\eta$ to a local minimum of the loss function given by $\mathcal{L}$.

$$\mathbf{w}^{(\tau)} = \mathbf{w}^{(\tau-1)} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{w}} \tag{12}$$

Clearly, there are some choices in the *hyperparameters*, to be made. Firstly, there is a choice of loss function. Instead of using the $L_2$ (Euclidean) norm, this loss could have used any $L_p$ norm instead. Furthermore, this loss term could have taken on a completely different form, with the most common for a classification task being the *cross-entropy loss*. Furthermore, a ridge, or $L_2$ regularisation term could be introduced to the traditional loss function as in equation (13).

$$\mathcal{L}_{reg}(\mathbf{w}, * = \mathcal{L}(\mathbf{w}, *) + \lambda ||\mathbf{w}||^2 \tag{13}$$

Lastly, the exact specification of the function $\mathbf{h}$, called either the algorithm for non-neural network based approaches, or the architecture for network approaches, is available.

Overall, supervised machine learning is a highly productive methodology for producing empirically useful models. Furthermore, the capacity to test an idea depends only on the differentiability of the loss function with respect to the weights. Even if the loss function is not analytically differentiable, numerical gradients have been demonstrated to perform well. This will become important in the discussion of variational quantum circuits.

(b) *Unsupervised Machine Learning*
Unsupervised machine learning, by contrast, does not require labelled examples, and finds patterns through another method. A typical unsupervised algorithm is that of clustering, as shown in Fig. 4, where the K-means algorithm has been applied to separate the data points. One important, and commonly used, method of improving

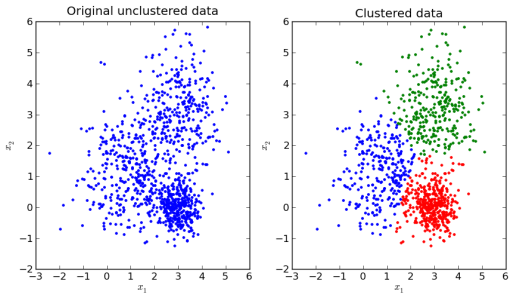


Figure 4: The k-means clustering algorithm relies on computing the distance between points.

the performance of unsupervised algorithms in particular, is through the use of a *feature map*. For example, a clustering algorithm like k-means, which groups data by proximity, would not be able to cluster something as in 5, since the actual distribution is circular.

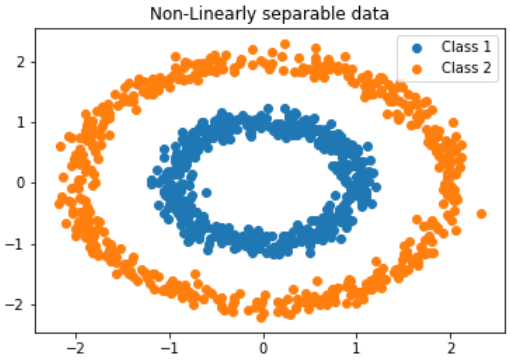


Figure 5: K-means would not be able to cluster data of this form.

However, introducing a feature map $\phi(x_1, x_2) = (x_1, x_2, x_1^2+x_2^2)$ then a machine learning algorithm would be able to separate out these two clusters significantly more easily, as shown in Fig. 6.

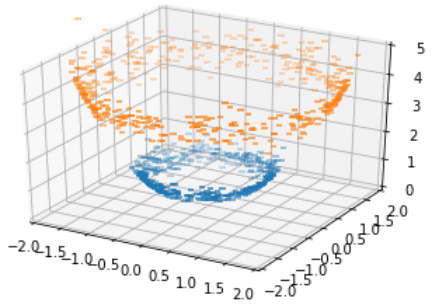

Figure 6: By using a feature map the dataset is amenable to many more machine learning algorithms.

Therefore, feature maps are an incredibly useful tool for improving the capacity of machine learning models. However, this form of feature map comes at the cost of an increase in dimensionality. Increasing the dimensionality creates two problems. Firstly, computing the feature map $\phi$ can be expensive and sometimes impossible, and secondly, when repeated norm calculations are required, computing the inner product in a high-dimensional space can be prohibitively expensive.

Computational cost is one of the biggest factors in choosing machine learning algorithms. As seen in Fig. 7, between 2014 and 2020 the computational cost of training the state-of-the-art machine learning algorithms has grown exponentially, doubling every 3-4 months [10]. This means that machine learning is now right up against the edge of possibility with the current computational power, and progress will start to slow if the computational resources cannot match the requirements. That being said, full model training happens only once,

and whilst models need to be updated as the data distribution drifts, this requires only *fine-tuning* pre-trained models onto the newly acquired data. In fact, this method of using pre-trained models extend beyond fine-tuning for new data, but can also be applied to train new models entirely. For example, if I wanted to perform a classification task in natural language processing, I could use a pre-trained text-model that already understood many of the problem specific features, such as the structure of long-term dependencies in English grammar. This method of fine-tuning pre-existing model weights is called transfer learning [11]. On the other hand, whilst training is expensive, the cost of *inference* is usually relatively cheap. That is, the infeasibly large computational costs arise from calculating gradients of petabytes of data with respect to consistently growing network architectures, however performing one forward pass of an example to provide the output is easy in comparison. This means that neural network models trained using thousands of compute hours on expensive hardware by major companies can be deployed into production on relatively computationally weak devices like mobile phones.

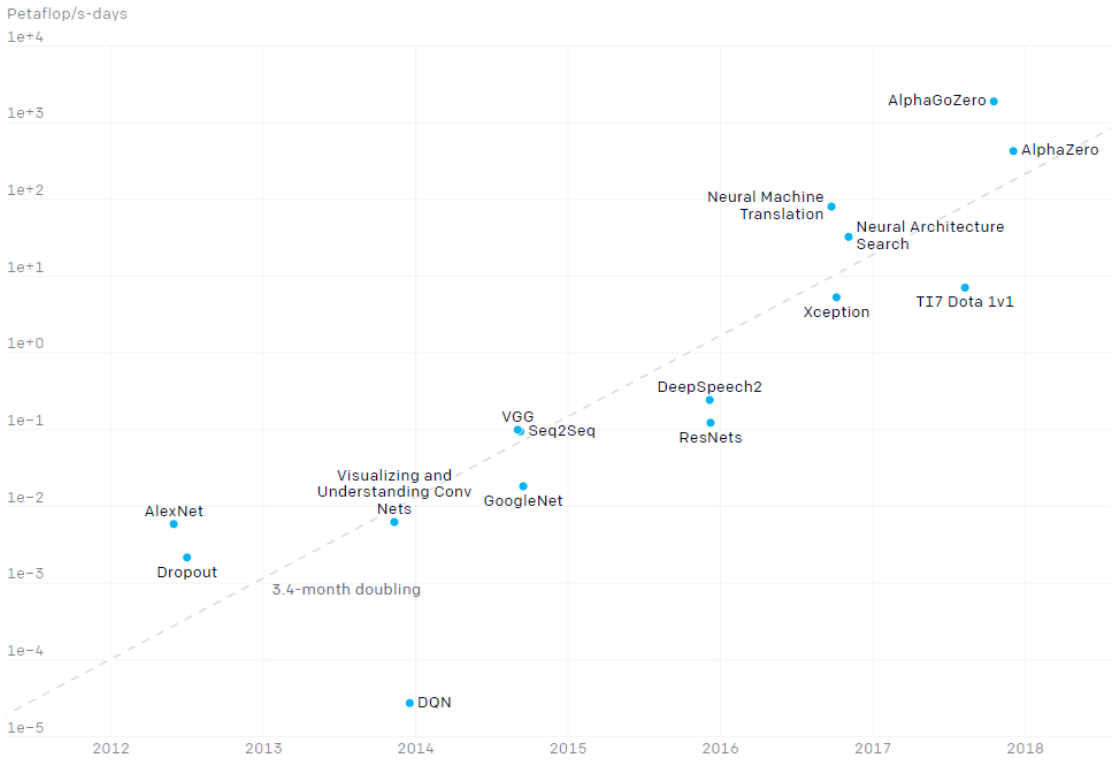


Figure 7: The computational cost of training deep neural networks has been scaling exponentially since 2014, doubling ever 3-4 months (image source [10]).

# 3 Quantum Machine Learning

Quantum machine learning is the emerging interdisciplinary field of utilising quantum computers to either aid, or perform, machine learning tasks. In this section, I will motivate quantum machine learning, provide an analysis of the the current field, and a method for

which to categorize, assess, and interpret the results of quantum machine learning tasks. I will also critically analyze a recent QML paper in order to demonstrate the necessary considerations when designing an algorithm, and use this to derive a potential methodology for producing novel QML research.

There are two major reasons that quantum machine learning is an exciting research space; the ability to generate quantum algorithms autonomously, and the expensiveness of classical machine learning. We are currently aware of few useful quantum algorithms that have the potential to be used on existing quantum computers, showing the difficulty in generating quantum algorithms. A universal quantum estimator, in theory, should be able to learn an arbitrary quantum algorithm from data. While it is unknown exactly which, if any, of these learned algorithms can give consistent quantum advantage, recent examples such as the Variational Quantum Linear Solver [12] (VQLS) provide a positive outlook for the study in general. Many classical machine learning algorithms rely on linear algebra sub-routines. In 2009, the HHL algorithm was introduced as a method for solving a subset of linear algebra problems with exponential speed-up, leading to the production of many quantum-enhanced machine learning algorithms such as QPCA and QSVM [13]. Since then, doubt has been cast into whether the HHL algorithm still provides an advantage [14], however new and improved methods such as the hybrid HHL [15] and VQLS mean that these algorithms still could provide benefits. In fact, accelerating the machine learning pipeline in this way is considered by some to be the 'holy grail' of quantum computing [16].

Quantum machine learning is best understood through the agent-environment (AE) paradigm.The agent environment paradigm is more commonly used to describe reinforcement learning algorithms as opposed to more general machine learning, however, its application to the categorization of QML algorithms allows for the most specific delineation. In the AE paradigm, the options for the agent and environment are classical or quantum. This naturally leads to four possibilities - Classical-Classical (CC), Classical-Quantum (CQ), Quantum-Classical (QC), Quantum-Quantum (QQ) [17]. Then, we can define 'standard' ML approaches to be CC, approaches for using standard machine learning on quantum systems, such as in quantum control in CQ, and call these 'classical machine learning'. Classical machine learning is also often used to describe non-neural network based approaches, but here it means not quantum. It is important to note that other papers use a different decomposition, the data-algorithm paradigm, where "CC" is still fully classical, but "CQ" becomes any quantum algorithm that uses classical data. This then relegates quantum data to be, in fact, classical data that is generated from a quantum system. Due to the fact that quantum data in this paradigm is a purely meaningless distinction, and obfuscates the interesting differences between algorithms, the agent-environment paradigm is preferred.

Algorithms that use a quantum agent are clearly the candidates for a definition of 'quantum machine learning.' The quantum environment is defined by having access to quantum data, and as such, any algorithms that require quantum ram will be in this category (QQ). The quantum agent operating in a classical environment is for those types of algorithms that do not require arbitrary state preparation, such as annealing techniques. Since the interest of this paper is in universal quantum estimators, the focus of the discussion will be of those algorithms situated in QQ.

The last distinction to draw is to separate those algorithms which aim to automatically generate novel quantum algorithms, and those designed specifically to obtain a quantum

speedup through specific quantum subroutines. I define the latter, using quantum subroutines to accelerate existing machine learning algorithms as quantum-enhanced machine learning, and define finally quantum machine learning to be the research of machine learning methodologies that can be executed on a quantum computer.

## 3.1 Quantum-enhanced machine learning

Being able to efficiently compute basic operations in linear algebra is incredibly important in a vast number of machine learning algorithms, and is of great interest for quantum computers since their action is described exactly through linear algebra. Algorithms such as HHL [13] have provided methods for gaining an exponential increase in the runtime of many linear algebra subroutines, leading to the development of the quantum basic linear algebra subroutines [18], which can improve the speed of machine learning algorithms. One such example is in matrix inversion for square matrices (or the Moore-Penrose Pseudo-inverse) which is an important task in many machine learning algorithms, with the fastest known classical algorithm being $O(n^{2.372})$. However, the (ommited) constant term out the front of this algorithm is so large that the most commonly implemented runtime complexity of matrix inversion is $O(n^{2.807})$. Matrix inversion, via phase estimation, can be computed exponentially more quickly using the HHL algorithm, which has led to the development of Quantum Principal Component Analysis [19] and Quantum Support Vector Machines [20]. These algorithms posit the existence of qRAM, and would require orders of magnitude more qubits than are currently available. Furthermore, the gate depth of these algorithms is often well beyond the best currently expected decoherence times, and so would also require robust quantum error correction methods. Furthermore, whilst these approaches are useful demonstrations of the abilities of HHL, they both invoke strict assumptions on the type of the data, such as the sparsity of the data matrix. The most common place for sparse matrices are in solving partial differential equations, and despite how much more effective they can make training even classicaly, they are a rarity in the problem spaces that machine learning is most interested in.

Despite being provably more effective, due to the limitations these algorithms are only mildly exciting for dramatically impacting the way that machine learning is done. Firstly, the specific machine learning algorithms that have been targeted for speed-up, are usually not the computationally expensive algorithms that people are nowadays training. The main reason for this is that all of these algorithms reach their learning capacity after a (relatively) small number of data points compared to the more common modern algorithms in neural networks. Furthermore, these algorithms have little hope for being implemented on a quantum computer at any point in the NISQ era.

## 3.2 Variational quantum circuits

Variational quantum circuits use gradient descent in classical optimization routines to update internal parameters such that the outcome of measurements best mirror a desired distribution. The trade-off for variational quantum circuits is the reduction in quantum circuit depth at the expense of additional classical optimization. Specifically, almost all of these methods are described as *variational hybrid quantum-classical algorithms* (VHQCAs),

as they use classical computation in the optimization step, and they can be placed into greater network-based architectures. These quantum algorithms are the ones most likely to be useful in the NISQ era for two reasons. Firstly, despite the fact that any implementation on a real quantum computer will introduce noise, there are generic gradient descent optimizers that are (empirically) spectacular in their ability to find an optimum through noise [21]. Secondly, the gate depth can be exactly as much as desired by the researcher. However, in this case, it is not feasible to reproduce neural network type advantages. If a parameterised gate matrix was multiplied by another parameterised gate matrix, then it would just return a new parameter matrix corresponding to a linear transformation, which is (clearly) linear. Since there is no introduction of non-linearity, the Universal Approximation Theorem cannot be applied. It is easy however to introduce classical non-linearity between linear transformations, which are exactly the structure of neural networks.

One of the most exciting variational quantum circuits is the Variational Quantum Linear Solver [12], which utilises variational quantum circuits in order to solve systems of linear equations. This is a particularly important step in the field for two reasons. Firstly, they successfully implement their quantum linear solver on a real quantum computer, which was impossible with previous linear solver techniques. Secondly, the computational complexity class lies in Deterministic Quantum Computing with 1 Clean Qubit (DQC1), which is believed to be infeasible to classically simulate [22][23].

In fact, variational algorithms have been proposed that allow for quantum factoring far sooner than expected [24], for quantum state tomography [25], as well as across many other disciplines [26]. Due to their applicability in the NISQ era, with few qubits and short decoherence times, useful noise resistant small-depth algorithms must be created. Variatonal quantum circuits are demonstrating their ability to dramatically reduce the requirements required in quantum algorithm creation, and demonstrating that optimization methods can learn how to exploit the quantum nature of quantum computing. Therefore, I think that variational hybrid quantum-classical circuits are going to take up a substantial portion of the useful algorithms produced in the near-term.

# 4 Quantum convolutional neural networks

## 4.1 Convolutional Neural Networks

Convolutional neural networks employ the use of convolutional kernels to extract meaningful features from images, and use those features to perform statistical inference. A convolutional kernel $\mathcal{F}$ takes the form of an $n \times n$ matrix, and is applied to an $n \times n$ subsection of a matrix $\mathcal{I} \in \mathbb{R}^{h \times w}$, representing an image, and performs a discrete correlation of $\mathcal{F}$ and $\mathcal{I}$ to produce a scalar output. In general, a convolutional kernel is a function $\mathcal{K} : \mathbb{R}^{n \times n} \times \mathbb{R}^{n \times n} \to \mathbb{R}$. In practise, it is defined as in equation (14), where the row and column numbers determining the top left corner of the image subsection are given by $r, c$ respectively,

$$\mathcal{K}(\mathcal{F}, \mathcal{I}_{r,c}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathcal{F}_{i,j} \mathcal{I}_{r+i,c+j} \tag{14}$$

Computing this function varying $r, c$ to all allowed values, while padding the image with

zeros such that this operation is well defined at the boundaries, another matrix of the same size as $\mathcal{I}$ is produced. This process is called *convolving* the image $\mathcal{I}$ with filter $\mathcal{F}$. It is worth noting that this is a misnomer in that it is not in fact a traditional convolution operation, but a correlation. One commonly used kernel is the Sobel-y filter ((15)), which is a horizontal edge detector.

$$S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{15}$$

Applying this filter to the image of a face produces Figure 8. This transformation is useful in computer vision since it allows for simple statements to be made about the content of the image, and programatically detect features of an image.

Figure 8: The application of the Sobel-y filter highlights areas of the image where there are large changes in horizontal colour.

Convolutional neural networks, however, do not use pre-defined convolutional kernels like the Sobel-y kernel. Instead, they parameterise $N$ kernels, of (in general) variable size, and use a supervised gradient descent to determine parameter updates in order to minimize the relevant loss function. The relevant aspect to this paper is that there are many different convolutional kernels, each of which produces an output matrix, called a feature map, which can be stacked into a tensor. This means that for each convolutional layer in the network, the 'depth' of the feature tensor increases as in Figure 9.

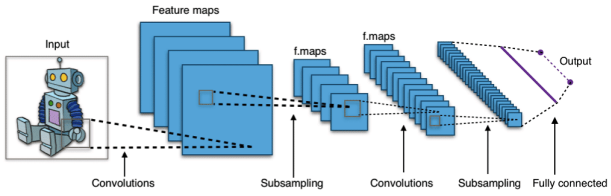


Figure 9: The general structure of a convolutional neural network

## 4.2 The Quanvolutional Layer

Henderson et al. (2019) [27] proposed a pre-processing method inspired by the process of a convolutional layer that used quantum circuits. A pre-processing method is fully decoupled

from the learning algorithm, and impacts the effectiveness of the learner. For example, a ubiquitous pre-processing step is data standardization, where you divide the data by its variance along each dimension, and subtract the mean. This is done so that the data is unitless, and the chosen measurement unit of data generation isn't considered.

Since a convolutional network operates on a small subsection of an image at a time, and many of the images can be computed in parallel if the hardware allows it, this is a high-value target for NISQ era devices, characterized by a small number of qubits, but with the ability to produce many of these quantum devices.

Formally, this pre-processing step is broken into 3 sections

(1) An encoding function $\mathbf{e}$ which takes in an image subsection $I$ of size $N$ and maps it either into an amplitude encoding of size $ceil(\log_2 N)$ or into a basis encoding of size $N$. Ancillary qubits could be added such that non-linearities could be better approximated.

(2) A quantum circuit $U$ that performs computation on the qubits.

(3) A decoding function $\mathbf{d}$ that converts from a probability distribution received from measurement to a scalar value.

Therefore, the total quantum convolutional layer $h(I)$ can be written as in 16

$$h(I) = \mathbf{e}(U(\mathbf{d}(I))). \tag{16}$$

As can be seen in Fig. 10, the quanvolutional layer takes in an image subsection of arbitrary size, and turns that into a scalar output. The representation in this figure importantly does not represent the most general quanvolutional layer. Instead, it implicitly defines a form of basis encoding, where each pixel value is mapped onto a single qubit. It also demonstrates measurement each qubit, which is not a requirement in general. One of the main reasons for the success of convolutional neural networks is that there are many different convolutional kernels at each layer. To mimic this property, many quantum circuits are used so that a separate feature matrix can be produced by each circuit.

This quanvolutional layer algorithm is summarized as in Algorithm 1.

**Algorithm 1:** The Quanvolutional Layer algorithm.

```
Result: A feature tensor of shape (num_filters,height,width)
filters ← get_qcircuits(num_filters) ;
foreach filter ∈ filters do
    feature_map ← apply_qcircuit(filter,image) ;
    yield feature_map ;
end
```

# 5 Technical implementation of a Quanvolutional Layer

## 5.1 Experimental setup

In order to reproduce results and discuss the practical implications of quantum algorithms for machine learning, I will discuss a specific instantiation of the general quanvolutional layer

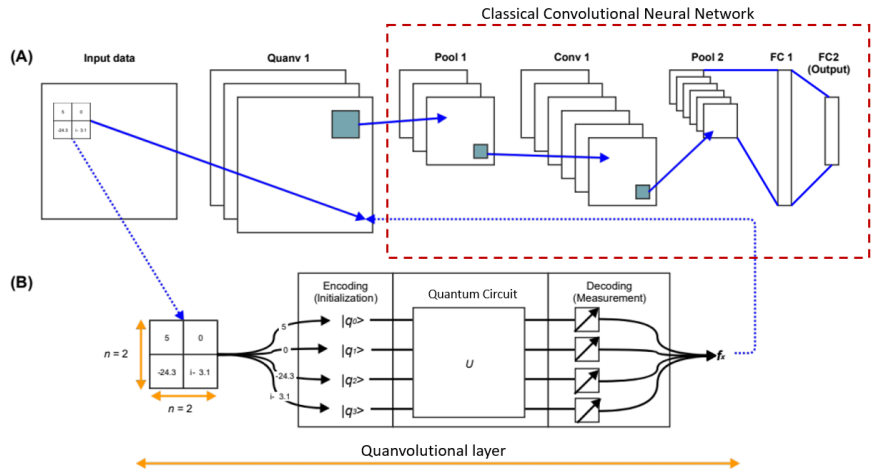


Figure 10: How the quanvolutional layer fits into the convolutional neural network pipeline, modified image from [27].

defined above.

The specific implementation we will investigate, as seen in the reference paper [27], is to convert the image subsection into a binary string through a thresholding function, and then use binary basis encoding to initialize this quantum state for each desired kernel. This state is then passed through a random quantum circuit of fixed gate depth, with the outcome of measurements determining the value of the filter through a decoding step. The decoding method used was max-state sum, which finds the most probable output state through running many computational measurement shots, and counts the number of individual qubits that are in the $|1\rangle$ state to produce an integer.

The random circuit was generated by randomly sampling gates, and deciding whether to apply those gates with a specific probability. The back-end for simulating the action of these quantum gates was provided by qiskit, using the 'qasm_simulator'. The gate sets were chosen to be native to as many different quantum architectures as possible, and as such a small basis was chosen. The possible single gates were the Hadamard; Pauli X, Y and Z; and a rotation around the Z by random angle gates. The only multiple quibt gate was the 2-qubit gate Controlled-NOT. Fig. 11 shows an example of a random quantum circuit. One thing to note is that, due to randomness, there is a useless computation on the 4th qubit, where two Hadamards are applied in a row, which does no computation, since the Hadamard gate is represented by an involutory matrix.

One hyperparameter of the quanvolutional layer is the number of shots. Since our decoding strategy of max state sum relies on knowing the probability distribution over all of the possible $2^n$ output states, and since I was using $2 \times 2$ filters, and therefore had 16 possible output states, I decided that doing 1000 computational shots would be sufficient. Furthermore, since I was caching the results on every unique $2 \times 2$ grid that I encountered, performing more computational shots only incurred a once-off penalty.

The binary thresholding was done by simply rounding the standardized pixel value. However, the choice of how to perform this thresholding impacts the models capacity to learn,

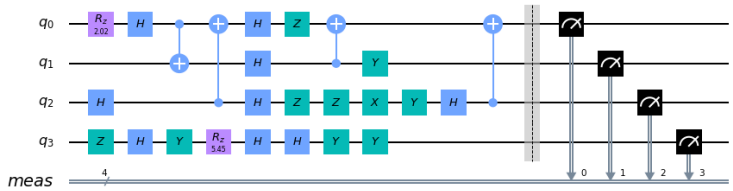


Figure 11: An example of a random quantum circuit pulled from the gate set with probability of applying a CNOT=0.2, a random single-qubit gate of 0.3.

since it can destroy information. One immediate alternative, that I hypothesise would perform better on this particular dataset, would be to say that any non-zero value becomes fully white. However, since this would not work for real images, or any more complicated datasets, I decided to use the rounding thresholding. It is also important to note that, in comparison to the dataset used in the original paper, binary thresholding will create a much more dramatic impact in this work, since many images in the dataset have brightness gradients within the image that aid in classification, in comparison to the MNIST handwritten digits dataset. The effect of binary thresholding on two examples from fashion-MNIST can be seen in Fig. 12.

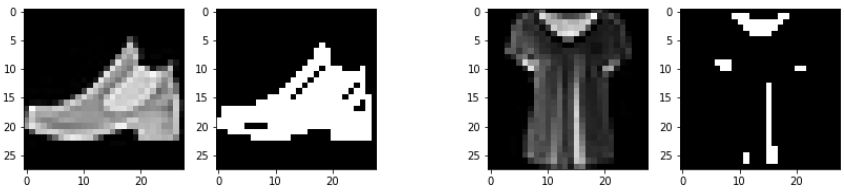

Figure 12: The effect of binary thresholding is dramatically different depending on the image.

The dataset that I will be using is the fashion-MNIST dataset. This dataset is of $28 \times 28$ grasycale images, and is significantly more complex than the dataset of the reference paper. Fashion-MNIST is comprised of 60000 training images and 10000 test images, where each item is one of 10 classes. The 10 classes are t-shirt, trouser, jumper, dress, coat, sandal, shirt, sneaker, bag and ankle boot [28]. An example of a training example from MNIST is shown in Fig. 13, which is of handwritten digits. The reason for choosing a harder dataset, is that even extremely simple convolutional network architectures perform extremely well on the MNIST dataset, and as such it is hard to analyze whether there is a statistically significant change in performance, since after relatively few iterations all reasonable network architectures achieve an accuracy of over 99%.

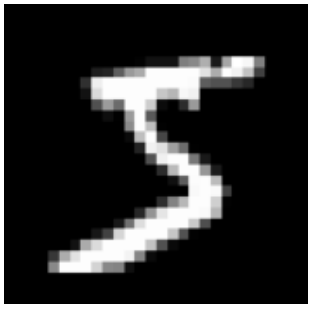

Figure 13: MNIST examples do not suffer much under binary thresholding, and training on MNIST is significantly easier than fashion-MNIST.

## 5.2 Results

The training results are shown in Fig. 14, and they indicate that the quantum model was better able to generalize to unseen training examples since it had better performance on the test set. Both models were subject to $L_2$ weight regularization with regularization parameter $\lambda = 0.01$ and were trained with a learning rate of $\eta = 0.01$. There is clear evidence of overfitting for both the quantum and classical model, however the overfitting was worse for the quantum model, as demonstrated by the fact that the test-loss had a local minimum earlier, and also grew much more quickly. These results are consistent with the reference paper, and demonstrate that the quantum layer is providing value in the learning process.

In order to explain the reason for the improvement in performance, I plot the actual action of the quanvolutional layer for a single example. As shown in Fig. 15

This plot reveals that the quanvolutional layer is performing the exact function as a series of random convolutional filters, in it's ability to highlight similar regions of the image. This is unsurprising since the quanvolutional layer was designed to mimic a convolutional layer. Somewhat surprising is that in each image there is a clear difference in regions of the image, since the randomness of the circuit did not ensure that its output would change based upon the input.

This also informs both the improved performance, and the increased tendency to overfit. Data augmentation is a common technique for improving the ability of image recognition tasks. For example, if we are classifying cats and dogs, it doesn't matter whether the image is mirrored, since the data distribution of the features that is considered 'dog-like' is symmetrical. However, the arrangement of pixels is different, especially if the un-transformed image itself is highly unsymmetrical (e.g. the dog is tilting its head). Data augmentation techniques need to be specific to the problem, since clearly the mirror image of the MNIST dataset of handwritten digits is no longer a valid training example. That is, that a horizontally mirrored '3' is not a three. In this case, since the reference images are thresholded (i.e. either 1 or 0) this method allows for more variation to be introduced in the color scheme depending upon, as an example, whether it is an edge or not. As for the tendency to overfit, since there are 5 images to memorise, there are 5 times as many opportunities to use a convolutional kernel, that happens to have small values (since the regularisation will still have a large impact) that memorizes the input-output combination.

Therefore, I have shown that this approach does provide a benefit. However, the benefit

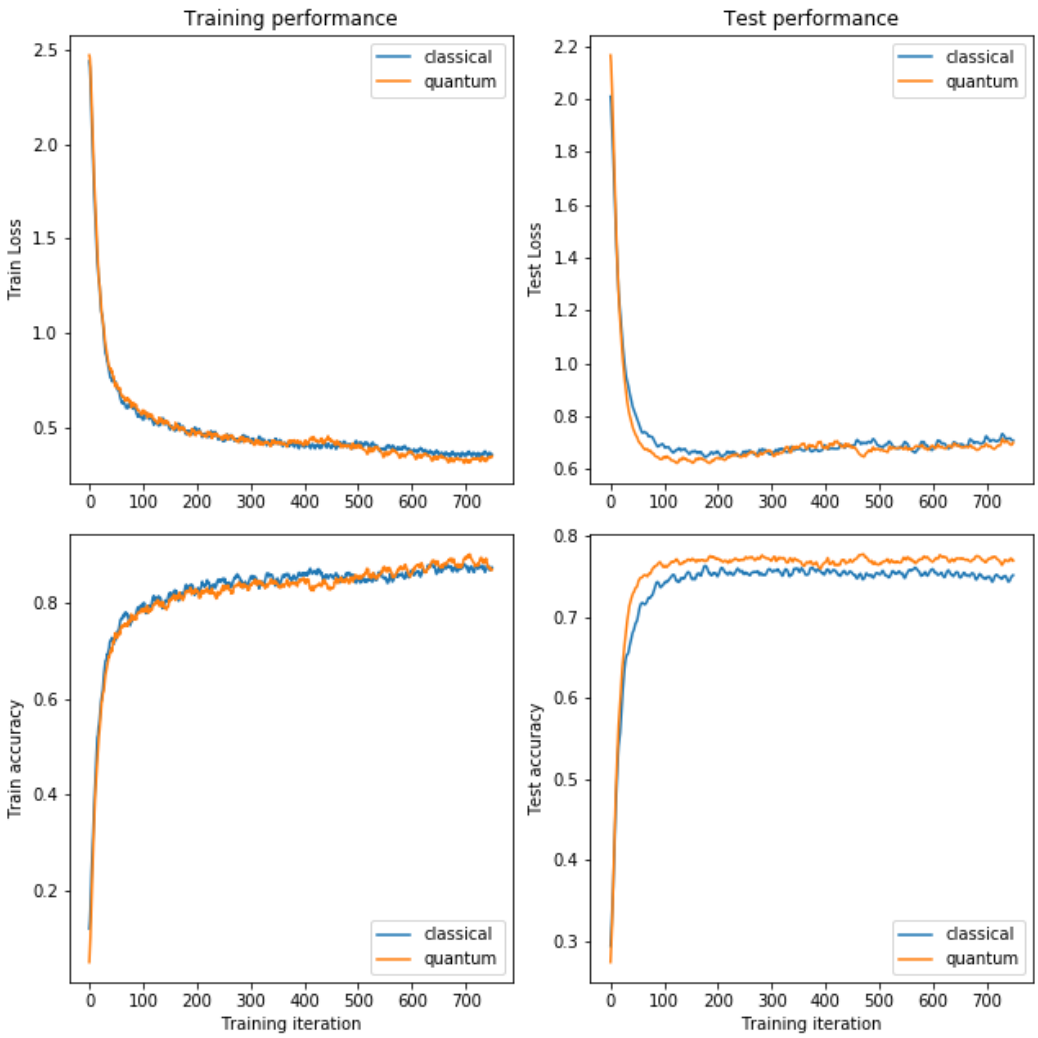


Figure 14: The results indicate that the quanvolutional layer improves the generalization of the model.

it provides could easily be replicated by a series of random convolutional kernels, and I am of the opinion that it is unlikely to provide the same advantage on non-grayscale images. Instead, I suspect on RGB images test performance will be worse since the data augmentation will not at all match the data distribution.

## 5.3 Outlook and critical analysis of this implementation

With the technical implementation pursued above, and as in the reference paper, there is no clear hypothesis for where a quantum advantage could be derived. Using a binary basis encoding, as previously discussed, requires an identical number of classical and quantum bits to store the image subsection. Instead, it would be easy to transition to a continuous basis encoding, where there would be a linear reduction in the number of required bits. The decision to use a binary basis encoding was out of necessity, since it meant that the

Figure 15: The quanvolutional layer acts to pick out regions of similarity, and produces plots similar to those that a normal convolutional layer would.

total possible inputs to each quantum circuit was small, and I could store the results of each of these possible inputs to look-up if they were seen later, instead of re-running the quantum computation. This is highlighted in comparing my technical implementation with the implementation from quantum algorithm company Xanadu [29], where a continuous basis encoding meant that their training set was forced to have as little as 50 images, in comparison to the 60000 used in my case. Fig. 16 reveals the layer architecture that Xanadu pursued.

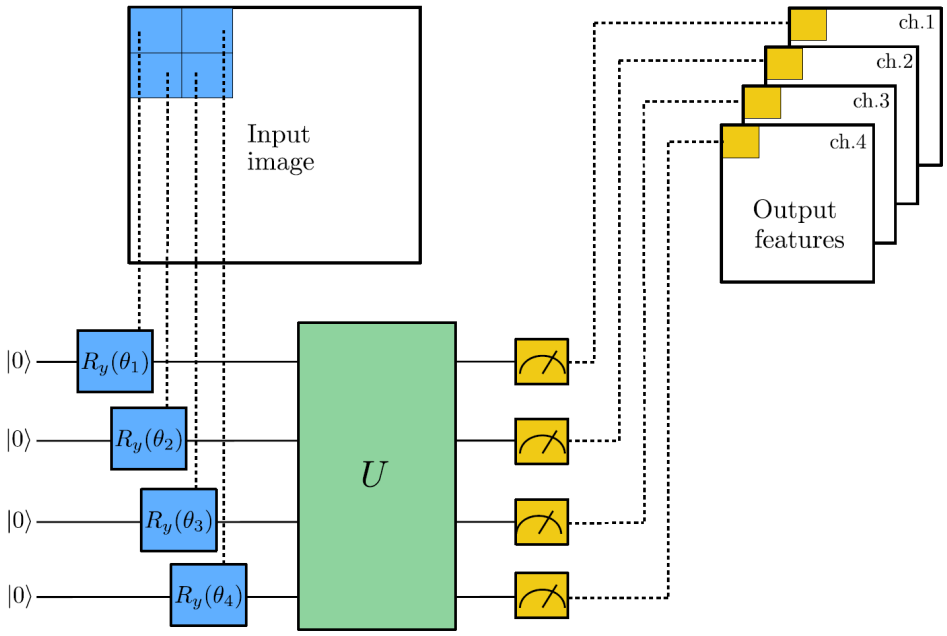


Figure 16: Xanadu's implementation of the same paper decided to make different choices to the original paper, revealing how important trade-offs surrounding computational complexity and layer architecture mean pursuing different strategies. Image source [29].

Furthermore, the computational complexity of a single convolutional operation is small, on the order of O(filter_size$^2$), and it is the pixel-wise application, on the order of O(width$\times$height) of many images which introduces the main computational complexity. Therefore, even in the case that the quantum circuit could be executed significantly faster than a single convolution, it likely wouldn't provide a sufficient speed-up to justify the continual requirement for reading the classical information from classical data storage and performing a state-preparation

routine to encode the date onto a quantum processing unit. This means that the increase in model performance would need to be significantly greater before it was worthwhile, even in the case of ideal quantum computers, to perform this operation.

Furthermore, in using a random quantum circuit with a max state sum decoding methodology, there is no clear reasoning as to how quantum parallelism, or other known useful quantum effects, such as Grover's or the Fourier transform is being leveraged.

Overall, this method of performing a binary basis encoding, using a random circuit, and using max-state sum decoding, shows little promise of developing into a useful algorithm. However, this paper introduced a general methodology for using quantum circuits as a convolutional kernel. I think there are two ways to turn this algorithm into something that could leverage quantum advantage and allow it to use the potential of quantum. Firstly, the quantum circuit could be parameterised into a variational quantum circuit, and the gradient with respect to the output could be used to update these parameters. Furthermore, if the quantum circuit was differentiable with respect to previous convolutional layers, then the position of the quantum layer could be changed, to occuring later in the neural network, where the feature maps are significantly smaller and as such significantly fewer iterations of the quantum circuit would need to be applied. Secondly, there is no need to define a decoding method. Instead, this decoding method could easily be incorporated as a fully connected layer in a neural network, and then the optimal decoding could be learned for this specific problem. This would allow the network to learn from the total probability distribution over the output states, instead of focussing solely on the state with the highest probability.

# 6 Conclusion

Quantum Machine Learning is an exciting possible application of Quantum Computers in the near-term. While much work has gone into designing sub-routines to achieve a quantum speed-up on classical, CC and CQ, algorithms, these algorithms assume an ideal quantum computer, with a very large number of qubits and the existence of easy quantum state preparation. Instead, Variational Quantum-Classical Hybrid algorithms have empirically demonstrated their ability to leverage classical optimization routines to find quantum advantage on specific tasks that scale well with the high-dimensional spaces that quantum computers exploit. In exploring a specific implementation of a recent quantum machine learning algorithm, I demonstrated the types of choices that researchers are forced to make when hand-crafting an algorithm, and identified multiple avenues for improving this particular approach.

The implications of this paper are that the most likely method for producing useful quantum algorithms in the near term are through the use of classical machine learning techniques applied to parameterised quantum circuits. A possible methodology for producing novel quantum machine learning research arises naturally. First, create a quantum formalism for an existing classical task. Secondly, produce a trainable parameterised circuit that can be trained to perform this task. Lastly, empirically determine whether this provides quantum advantage over its classical counterpart.

The next piece of work is to parameterise the quanvolutional layer in such a way that it is differentiable both with respect to the cost function, and with respect to the network

architecture, in order that it can be moved to other parts of the networks, and can learn convolutional kernels that exploit quantum effects.

# References


[1]M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information: 10th anniversary edition*, 10th (Cambridge University Press, USA, 2011).

[2]F. Arute, J. M. Arya, et al., "Quantum supremacy using a programmable superconducting processor", Nature **574**, 505–510 (2019).

[3]C. M. Bishop, *Pattern recognition and machine learning (information science and statistics)* (Springer-Verlag, Berlin, Heidelberg, 2006).

[4]T. J. Sejnowski, "The unreasonable effectiveness of deep learning in artificial intelligence", Proceedings of the National Academy of Sciences (2020).

[5]B. C. Csáji et al., "Approximation with artificial neural networks", Faculty of Sciences, Etvs Lornd University, Hungary **24**, 7 (2001).

[6]J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning", Nature **549**, 195–202 (2017).

[7]N. Mishra, M. Kapil, H. Rakesh, A. Anand, N. Mishra, Warke, et al., *Quantum machine learning: a review and current status*, 2019.

[8]D. Aharonov, "A simple proof that toffoli and hadamard are quantum universal", (2003).

[9]V. Dunjko, J. M. Taylor, and H. J. Briegel, "Advances in quantum reinforcement learning", 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (2017).

[10]D. Amodei, *Ai and compute*, May 2018.

[11]K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning", Journal of Big Data **3**, 9 (2016).

[12]C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, and P. J. Coles, "Variational quantum linear solver", (2019).

[13]A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations", Physical review letters **103**, 150502 (2009).

[14]N.-H. Chia, H.-H. Lin, and C. Wang, "Quantum-inspired sublinear classical algorithms for solving low-rank linear systems", (2018).

[15]Y. Lee, J. Joo, and S. Lee, "Hybrid quantum linear equation algorithm and its experimental test on ibm quantum experience", Scientific Reports **9** (2019).

[16]T. Gabor, L. Sünkel, F. Ritz, T. Phan, L. Belzner, C. Roch, S. Feld, and C. Linnhoff-Popien, *The holy grail of quantum artificial intelligence: major challenges in accelerating the machine learning pipeline*, 2020.

[17]V. Dunjko and H. J. Briegel, "Machine learning & artificial intelligence in the quantum domain: a review of recent progress", Reports on Progress in Physics **81**, 074001 (2018).

[18]L. Zhao, Z. Zhao, P. Rebentrost, and J. Fitzsimons, *Compiling basic linear algebra subroutines for quantum computers*, 2019.

[19]S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum principal component analysis", Nature Physics **10**, 631–633 (2014).

[20]P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big data classification", Physical Review Letters **113** (2014).

[21]H. Yong, J. Huang, X. Hua, and L. Zhang, *Gradient centralization: a new optimization technique for deep neural networks*, 2020.

[22]K. Fujii, H. Kobayashi, T. Morimae, H. Nishimura, S. Tamate, and S. Tani, "Impossibility of classically simulating one-clean-qubit model with multiplicative error", Phys. Rev. Lett. **120**, 200502 (2018).

[23]T. Morimae, "Hardness of classically sampling the one-clean-qubit model with constant total variation distance error", Phys. Rev. A **96**, 040302 (2017).

[24]E. Anschuetz, J. Olson, A. Aspuru-Guzik, and Y. Cao, "Variational quantum factoring", in Quantum technology and optimization problems, edited by S. Feld and C. Linnhoff-Popien (2019), pp. 74–85.

[25]Y. Liu, D. Wang, S. Xue, A. Huang, X. Fu, X. Qiang, P. Xu, H.-L. Huang, M. Deng, C. Guo, X. Yang, and J. Wu, "Variational quantum circuits for quantum state tomography", Phys. Rev. A **101**, 052316 (2020).

[26]Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. D. Sawaya, S. Sim, L. Veis, and A. Aspuru-Guzik, "Quantum chemistry in the age of quantum computing", Chemical Reviews **119**, 10856–10915 (2019).

[27]M. Henderson, S. Shakya, S. Pradhan, and T. Cook, *Quanvolutional neural networks: powering image recognition with quantum circuits*, 2019.

[28]H. Xiao, K. Rasul, and R. Vollgraf, *Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms*, (Aug. 28, 2017)

[29]A. Mari, *Quanvolutional neural networks — pennylane*, 2020.