The Viability of Quantum Measurement as the Activation Function in Quantum Neural Network Models

Marko Balogh and Devan Cakebread

June 15, 2017

Abstract

In this work we discuss the motivations for attempting to implement some features of artificial neural networks in a quantum system, and the difficulties in doing so—the main one being the reconciliation of the nonlinear, irreversible dynamics of neural networks and the linear, reversible dynamics of a quantum system. We examine the details of a proposal by Zak and Williams [ZW98] and discuss the compatibility of this proposal with the two computationally demanding algorithms relating to neural networks—the forward pass and backward pass. We conclude that the Zak-Williams Quantum Neural Network proposal may be useful in the layer-by-layer computations of the forward pass algorithm, but fundamental properties of quantum systems prevent it from being adapted for use in computing the backward pass.

1 Introduction

Connectionism is a paradigm in the study of artificial intelligence which hypothesizes that intelligence is an emergent property of systems composed of simple units connected together in complex architectures. Examples of algorithms falling under the connectionist paradigm include cellular automata and artificial neural networks, the latter of which is our focus in this work. The artificial neural network (ANN) is a ubiquitious algorithm in the study of artificial intelligence and machine learning, with applications like regression and classification, and elsewhere, including computational neuroscience. The algorithm evolved from attempts to simulate the information processing of biological neurons.

A biological neuron propagates electrical pulses down its axon. Whether the neuron fires is determined by the amount of stimulus the neuron receives through its dendrites, appendages that connect to the axons of other neurons. The neuron fires, or is 'activated', if and only if the total input via its dendrites surpasses a predetermined threshold.

An artificial neural network is a functional composition of many individual artificial neurons. Each artificial neuron is a model of a biological neuron simplified as much as possible but complex enough to capture the critical properties that give it information processing capabilities. All artificial neurons are nonlinear functions mapping vectors to scalars. The function implemented by the neuron is a composition of a linear 'integration' function and a nonlinear 'transfer' or 'activation' function. The integration function aggregates the inputs of the neuron, hence modeling the aggregation of signals from dendrites in the body of a neuron. This is typically a simple linear combination. The activation function then maps this linear combination to a firing state. In a neural network, by necessity, the domain of the artificial neuron is a cartesian power of its codomain. The simplest artificial neuron model is the McCulloch-Pitts neuron, represented by a function $f : \{0,1\}^n \to \{0,1\}$ such that

$$f(x_0, x_1, \ldots, x_{n-1}) = H(\sum_{i=0}^{n-1} w_i x_i), \tag{1}$$

where H is the Heaviside step function. However, a more general neuron model called the 'perceptron', often used for machine learning applications, is $f : [-1,1]^n \to [-1,1]$ such that

$$f(x_0, x_1, \ldots, x_{n-1}) = act(\sum_{i=0}^{n-1} w_i x_i), \tag{2}$$

where *act* is a differentiable activation function with codomain $[-1,1]$ like hyperbolic tangent or the error function (we use square brackets to denote an inclusive interval in $\mathbb{R}$). Through optimization of the weights w_i, a single neuron can perform linear regression on data normalized to the range $[-1,1]$—clearly, the integration function alone can implement any linear function of the inputs. When connected in a feedforward ANN (a notion defined in the preliminaries section), the combination of the artificial neuron's linear integration function and nonlinear activation function (eq.2) is known to allow the ANN to approximate any continuous function of its inputs to arbitrary precision, given that the number of neurons in the network is sufficiently large. This powerful result is known as the universal approximation theorem [Hor91].

Although an ANN is theoretically powerful, training ANNs is tremendously difficult in practice. Typically, the weights of the network are optimized using an algorithm called backpropagation [Hec88], a specialized form of stochastic gradient descent down the network's error surface. The calculation of the gradient is sometimes referred to as a "backward pass", to contrast the so-called "forward pass" in which the network's activation/firing state is computed. The computation of the network's state is called a forward pass because information is propagated forward through the network, passed from one layer to the next, until all the neuron activations have been computed. Precise descriptions of the forward and backward pass algorithms are given in the preliminaries section. Because the error surface is in general non-convex, with many local minima and saddle points, there is generally a high probability that any training attempt does not find the global minimum. In such cases, more advanced techniques like inertial gradient descent [Qia99] and particle swarm optimization [RB07] may be necessary, although local minima can often provide a sufficient amount of training.

Even with well-behaved data, training an ANN using stochastic gradient descent can involve millions of iterations of backpropagation. In each iteration, a forward pass is computed and the network's output is compared the correct value of the function which the network is learning. Then a backward pass computes the gradient of the network's error with respect to its synaptic weights. The weights of the network are then updated (element-wise) by an amount proportional to the gradient, where the proportionality constant, called the *learning rate*, is a hyperparameter whose optimal value depends on the size of the training set, the degree of non-convexity of the error surface, and the resources available for training. For complex learning tasks, an extremely large number of iterations may be required—this makes training ANNs one of the principle drivers of growing demand for high-performance computing. For this reason, whether it is possible to speed up either the forward pass or backward pass subroutines using quantum computation is a compelling research question.

This is the question that motivates this work. A recent review paper by Schuld, Sinayskiy, and Petruccione provides an introduction to the body of prior work involving some intersection of artificial neural networks and quantum computing [SSP14]. The paper begins by introducing the properties of ANNs and quantum computing which make the problem of how to combine the two both interesting and non-trivial. The authors explain that one of the critical properties of ANNs is their nonlinear dynamics. This immediately makes the problem of implementing ANNs as quantum algorithms challenging, because quantum systems evolve according to unitary linear dynamics. Schuld et al review a handful of attempts to reconcile these facts—of these, here we examine in detail a particular proposal Schuld et al describe as "largely unnoticed but nevertheless interesting". We examine this proposal because Schuld et al claim that the authors fail to explain crucial details, opening space in which we may be able to provide valuable insight, and because the other proposals require a more technical understanding of the physical implementation of quantum computers which is not yet accessible to us.

The proposal we study here, by Zak and Williams [ZW98], attempts to model the evolution of an artificial neural network as the action of a unitary transformation on the network's present firing state, such that the network's firing state after the forward pass is given by a measurement of the transformed state. This unitary transformation of course would have to include information about the probabilities of transition between each basis state. According to Schuld et al, Zak and Williams did not provide a detailed description of how to obtain such a unitary transformation implementing the desired ANN dynamics. The main contributions of Zak's and Williams' work to the study of QNNs are the idea of using quantum measurement to provide the nonlinearity required for ANNs, and the idea of allowing the QNN to evolve probabilistically, but so that it converges to the state which a classical ANN evolves to deterministically. Later we will evaluate the compatibility of this proposal with the interesting features of classical ANNs.

Before diving into the details of this proposal, we formalize a few concepts valuable for our analysis.

2 Preliminaries

2.1 Quantum Computing (reference: [KLM07])

The state of a quantum system is described by a vector of complex *probability amplitudes* denoted $|\Psi\rangle$. By convention, this vector is written in the eigenbasis of a measurement operator. Typically in quantum computing we consider systems for which there is a measurement operator which has two eigenstates, like a spin-1/2 particle. We denote these eigenstates $|0\rangle$ and $|1\rangle$, and refer to this as the *computational basis*. The word *qubit* is used to refer to a quantum system with a measurement operator with two eigenstates, as a quantum analog of a classical bit.

The state of a composite system composed of n qubits is a 2^n-dimensional vector in the space that is the tensor product of the individual qubit bases. We write the state $|\Psi\rangle \otimes |\phi\rangle$ as $|\Psi\rangle\,|\phi\rangle$ or $|\Psi\phi\rangle$, where $\otimes$ denotes the tensor product. For example, the four computational basis states for a 2-qubit system are written $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$.

The specifics of the quantum system and measurement operator used to physically store a quantum state is usually not of much interest in the study of quantum computing within theoretical computer science. Similarly, the physical identities of the system's eigenstates are usually irrelevant. For this reason, although a quantum state (physically) is a complex superposition of the system's eigenstates, we identify a quantum state with the complex coefficients defining that superposition. Formally, a quantum state (of n qubits) is a vector $|\Psi\rangle = (a_0, a_1, \ldots, a_{2^n-1})$ where $a_i \in \mathbb{C}$ and $\sum_i |a_i|^2 = 1$. We sometimes write the basis vectors explicitly, so that $|\Psi\rangle = \sum_i a_i\,|i\rangle$.

Measurement refers to the application of a measurement operator to the quantum system. For our purposes, measurement of a quantum state $|\Psi\rangle = \sum_i a_i\,|i\rangle$ collapses the system into the eigenstate $|i\rangle$ with probability $|a_i|^2$. This is why a_i are referred to as probability amplitudes, and why we require that $\sum_i |a_i|^2 = 1$. We use $\mathcal{M}$ to denote a random function performing measurement on a given quantum state. Hence the value of $\mathcal{M}(|\Psi\rangle)$ is a random variable equal to the ith computational basis state with probability $|a_i|^2$.

An admissible operation on a quantum state is called a *quantum gate*. In the standard quantum circuit model, admissible operations must be possible to represent by a unitary matrix, whose action on a quantum state is computed by left-multiplication with the quantum state as a column vector. In this category we do not include so called 'general quantum operations', as the density matrix representation of quantum systems/ensembles is not necessary for our purposes here. A quantum gate U may be implemented by applying a hamiltonian H (an operator which encodes the forces on a physical system) to the quantum system for a time τ such that $U = e^{iH\tau/\hbar}$.

2.2 Artificial Neural Networks

Definition 1 (General Artificial Neuron)**.** Let X be an n-dimensional vector space, and Y be a set. We call a function $f : X \to Y$ an n-dimensional artificial neuron if there exists some nonlinear function g and some family of linear functions h_i such that $f(\mathbf{x}) = g(\sum_i h_i(x_i))$ for all $\mathbf{x} = (x_0, x_1, \ldots, x_{n-1}) \in X$.

Definition 2 (McCulloch-Pitts Neuron)**.** An n-dimensional McCulloch-Pitts neuron is a function $f : \{0,1\}^n \to \{0,1\}$ whose output on a given binary string $\mathbf{x} \in \{0,1\}^n$ is defined by an n-dimensional vector of weights $\mathbf{w} = (w_0, w_1, \ldots, w_{n-1})$, where $w_i \in [-1,1]$ such that

$$f(x_0, x_1, \ldots, x_{n-1}) = H(\sum_{i=0}^{n-1} w_i x_i), \tag{3}$$

where H is the Heaviside step function.

Definition 3 (Perceptron)**.** An n-dimensional perceptron is a function $f : [-1,1]^n \to [-1,1]$ whose output on a given vector $\mathbf{x} \in [-1,1]^n$ is defined by an n-dimensional vector of weights $\mathbf{w} = (w_0, w_1, \ldots, w_{n-1})$, where $w_i \in [-1,1]$ such that

$$f(x_0, x_1, \ldots, x_{n-1}) = act(\sum_{i=0}^{n-1} w_i x_i), \tag{4}$$

where *act* is a differentiable nonlinear activation function with codomain $[-1,1]$ like hyperbolic tangent or the error function.

Definition 4 (Feedforward Artificial Neural Network)**.** Let L be a tuple of tuples of n-dimensional artificial neurons, each tuple called a 'layer' and labeled L_k with some index $k \in \{0, 1, \ldots, |L| - 1\}$. Let W be a matrix of weights $w_{ij} \in [-1,1]$ where each row is the weights vector of a neuron in L. The tuple L is an n-neuron feedforward artificial neural network if the cardinalities of the tuples in L sum to n and $w_{ij} \neq 0$ if and only if the neuron corresponding to the i-th row in W belongs to L_k and the neuron corresponding to the j-th row in W belongs to L_{k-1} for some $k \in \{1, 2, \ldots, |L| - 1\}$ or the neuron corresponding to the i-th row in W belongs to L_0. The "state" of the network (sometimes called a "firing state" or "activation state") is a function of an input vector $\mathbf{x} \in X^n$ where X is the domain of the artificial neuron model used. The state is computed by Forward Pass$(L, \mathbf{x})$, described next. Note that we use $f_{a,b}$ with no argument to denote the current firing state of the b-th neuron in the a-th layer. If $f_{a,b}$ has not yet been defined at the time of use it can be assumed to be zero.

Algorithm 1 Forward Pass [Ng11]

Input: An n-neuron feedforward artificial neural network L and an input vector $\mathbf{x} \in X^n$ where X is the domain of the artificial neuron used.

Output: The state $\mathbf{y} \in Y^n$ of network L, where Y is the co-domain of the artificial neuron used.

```
for f_{0,i} in L_0 do
    compute f_{0,i}(x)
end for
for k = 1 to |L| − 1 do
    for f_{k,i} in L_k do
        compute f_{k,i}(f) {where f is the vector (f_{0,0}, f_{0,1}, ..., f_{1,0}, f_{1,1}, ..., f_{|L|−1,0}, f_{|L|−1,1}, ...)}
    end for
end for
return y = f
```

In the forward pass, the activation states of the neurons of the first layer (called the "input layer") are computed as a function of the input vector. Then, the activation states of the next layer are computed as

a function of the current network state (of which only the first layer is nonzero). Then, the states of the next layer are computed as a function of the current network state (of which now the first two layers may be nonzero). Continuing like this, the activation state of the entire network is computed layer by layer. Each layer inherits information from the previous layer (with the exception of the input layer, which is provided information by the input vector). Hence the forward pass algorithm propagates information forward through the network. The output of the forward pass is the entire network firing state, although for many applications, the only information which is interesting is the firing state of the last layer of neurons, called the output layer. This is because this is the only information in the firing state which has been processed by the entire network. Training an ANN usually involves optimizing the activation of the output layer as a function of the synaptic weights in the network via stochastic gradient descent. For each training example, the gradient of the output error with respect to the synpatic weights must be calculated. This is what the backward pass algorithm does, which we describe next. We describe the algorithm under the assumption that there is a single neuron in the output layer. This does not sacrifice any generality since performing a backward pass for a network with multiple output neurons is identical to performing the following backward pass separately for each output neuron, as the output neurons must be independent to obey definition 4. Backpropagation requires that the activation function is differentiable, so we will assume that the underlying neuron model is the perceptron.

Algorithm 2 Backward Pass [Ng11]

Input: An n-neuron feedforward artificial neural network L, an activation state $\mathbf{y}$ of L, and a desired output activation $y^* \in [-1, 1]$.
Output: A gradient vector $\Delta \in [-1, 1]^n$.

```
let l = |L| − 1
let δ^l = −(y* − y^l) · act′(z^l)
for k = l − 1 to 0 do
    for f_{k,i} in L_k do
        let δ_i^k = (Σ_j W_{ji} δ_j^{k+1}) · act′(z_i^k)
        let Δ_{ij} = y_j^k · δ_i^{k+1}
    end for
end for
return Δ
```

We use z_j^k to denote the argument to the activation function in the j-th perceptron in layer L_k. Similarly, we use y_j^k to denote the activation state of the j-th perceptron in layer L_k. We use act' to denote the derivative of the activation function used by the network's perceptrons. Note that although we sometimes refer to vector components with two indices, this is merely for convenience to identify the component as corresponding to a specific neuron in a specific layer. One can view the actual index of the vector component in question as a function of the two indices used. With this in mind, we make a slight abuse of notation in the above description. The W matrix above is not the exactly the weights matrix used earlier, although it contains the same information. Some permutation has been applied to its elements, depending on the layer the backward pass algorithm is currently computing—the weight W_{ij} is the weight of neuron $f_{k,j}$ in the weight vector of neuron $f_{k+1,i}$. The gradient component Δ_{ij} is the component of the gradient corresponding to the weight W_{ij} as we use it here.

After the backward pass is computed, the gradient vector, modulated by the learning rate, is subtracted from the weights vector, thus performing gradient descent. A derivation and cultivation of intuition regarding the backward pass algorithm is available in many free resources including [Ng11]. For our purposes the critical insights are simply what sort of operations are performed by the algorithm, and what information it requires.

We now examine the details of the proposal by Zak and Williams before discussing the compatibility

of the proposal with the concepts defined above.

3 Measurement as the Activation Function

Zak and Williams begin by noticing that there is a strong parallel between the activation function of a neural network and measurement of a quantum system. The activation function converts the stimulus at each neuron into an activation state, while measurement converts a quantum system's probability amplitudes into one of the system's eigenstates. We take for granted that these eigenstates are the computational basis states. Although quantum systems in general can be measured via an operator with a continuous spectrum, in the context of quantum computing it is typically assumed that the quantum state is a spin state and therefore has a discrete spectrum. As a consequence, it is clear that Zak and Williams are limited to constructing neural networks out of McCulloch-Pitts neurons, since the use of measurement as the activation function implies that the neural network can only assume discrete activation states. Unfortunately, by sacrificing the use of continuous activation states, we also lose the universal approximation theorem—one of its requirements is that the activation function be continuous. However, there are interesting applications of neural networks other than function approximation/regression, which we will discuss in a moment. Let us now introduce the formalities of the work by Zak and Williams.

Zak and Williams conceive of an n-neuron network as a dynamical system obeying the differential equation

$$\tau_i \frac{\partial x_i}{\partial t} = -x_i + act\left(\sum_{j=1}^{n} T_{ij} x_j\right), \tag{5}$$

where τ_i is some positive time constant, x_i is the activation of the ith neuron, $T_{ij} \in [-1, 1]$ is the synaptic weight feeding the activation of the jth neuron as stimulus into the ith neuron, and act is the activation function. Notice that the above equation is a continuous-time generalization of the ANN in definition 4. In definition 4 there is no restriction that prevents the input vector from being the current firing state of the network. In this self-referential paradigm, then, the network is initialized to some firing state and is allowed to evolve to a new firing state. The notion of "feedforwardness" does not appear in equation 5, as there is no privileged direction in the network and no requirement of independence among neurons belonging to non-adjacent 'layers'. It is straightforward to see that allowing such a self-referential network to update continuously yields a dynamical system like the one in equation 5. Zak and Williams explain that generally, the applications of neural networks involve finding the appropriate values of the synaptic weights so that this dynamical system converges to an attractor. In optimization, for example, the network state at the basin of the attractor will somehow relate to the solution of the optimization problem (backpropagation is a special case of this), and in classification, there are different attractors corresponding with the different classes in the data. In developing a quantum analog to this neural network, it is therefore essential that the evolution of the system have this same characteristic of converging to attractors. Let's try to port equation (5) into the quantum setting, and see whether we can preserve this characteristic.

In order to use measurement of a quantum system as the activation function, time must be converted to a discrete parameter (indicating the number of measurements that have been performed). A quantum system cannot evolve at all if measured continuously, as a consequence of the fact that any operator $U = e^{iHt/\hbar}$ approaches the identity as t approaches zero. Furthermore, the argument of the activation function must be a quantum state, since measurement maps a quantum state to one of the computational basis states. Notice that the weighted sum in equation (5) can be thought of as the action of a linear operator T on the network state x. Naturally, T can be replaced with a unitary operator U acting on the quantum state $|\Psi\rangle$. Adopting these modifications, Zak and Williams reach the following quantum

analog of the neural network in (5):

$$a_i(t+1) = \mathcal{M}(\sum_j U_{ij} a_j(t)), \tag{6}$$

or, equivalently,

$$|\Psi\rangle_{t+1} = \mathcal{M}(U|\Psi\rangle_t). \tag{7}$$

Zak and Williams envision initializing the QNN into a computational basis state, so that each artificial neuron's activation state is fully determined, and then applying the unitary U to the system, so that the network is in a superposition of activation states as dictated by the probability amplitudes of $U|\Psi\rangle$, and then performing measurement, so that the network is placed back into a basis state. Then, as this process is repeated, the system will converge to a desired state, provided that the unitary operator U is chosen carefully.

Since the set of network states is discrete, and the evolution between states at each step is probabilistic, equation (7) can be thought of as describing a Markov chain. At any point in the chain, the probability that a network in the *jth* basis state transitions to the *i*th basis state is given by the modulus squared of the probability amplitude in the unitary operator which connects the *i*th and *j*th basis states:

$$\Pr[|\Psi\rangle_{t+1} = |i\rangle \,|\, |\Psi\rangle_t = |j\rangle] = |U_{ij}|^2. \tag{8}$$

Immediately, we can see one possible application for this QNN framework: Markov Chain Monte Carlo simulations. By choosing the elements of U appropriately, any desired Markov Chain can be simulated without the need for a random number generator, due to the inherent randomness of the quantum measurement process. Depending on details of the physical implementation, running Monte Carlo simulations of this sort on Zak's and Williams' QNN may be far more efficient than running an equivalent simulation on a classical computer, where computational resources would be consumed solely for producing psuedorandom numbers. Indeed, this application is one of Zak's and Williams' core motivations for their work. However, the usefulness of this idea in practice, of course, depends on ability of the physical device the QNN is embodied by to implement a robust set of unitary operators. Such details about physical implementations of quantum computers are beyond the scope of Zak's and Williams' work, and this report as well.

Without regard for the physical implementation of a unitary transformation, how does one find the ideal unitary transformation for a given application? For Monte Carlo modeling, the process is clear: a complete description of a Markov Chain is given by its state transition probabilities, meaning that the unitary transformation will be composed of this description directly, as in equation (8). Another application is associative (or "content-addressable") memory, in which the network's evolution over many interations of equation (7) converges to the basis state which is closest in terms of *Hamming distance* (the number of bit flips required to change one state into another) to the initial network state [SSP14]. Indeed, by inspecting equation (5) one can see why it is reasonable to expect this sort of behavior from a self-referential ANN. After all, the ANN must envolve into a valid firing state, and what better state than the one most similar to the initial state? This is one example of how artificial neurons adequately emulate biological neurons to inherit the emergent properties of biological neural networks, since the human brain clearly exhibits associative memory (we do not need to know the memory address to retrieve its contents; we need stimulus which can be associated with its contents).

To explain how their QNN architecture can be programmed to remember certain input patterns associatively, Zak and Williams introduce the notation π_i^m to represent the probability that the QNN is in basis state i after m iterations of (7). Using the law of total probability, we can decompose the probability that the QNN is in basis state i into a sum over the possible initial states as follows:

$$\pi_i^t = \Pr[|\Psi\rangle_{t+1} = |i\rangle] = \sum_{j=1}^{2^n} \Pr[|\Psi\rangle_{t+1} = |i\rangle \,|\, |\Psi\rangle_t = |j\rangle] \cdot \Pr[|\Psi\rangle_t = |j\rangle]. \tag{9}$$

Using equation (8) we may write

$$\pi_i^t = \sum_{j=0}^{2^n-1} |U_{ij}|^2 \cdot \pi_j^{t-1}, \tag{10}$$

which we can recast into a matrix equation as follows

$$\begin{bmatrix} \pi_0^t \\ \vdots \\ \pi_{2^n-1}^t \end{bmatrix} = \begin{bmatrix} |U_{00}|^2 & \cdots & |U_{0(2^n-1)}|^2 \\ \vdots & \cdots & \vdots \\ |U_{(2^n-1)0}|^2 & \cdots & |U_{(2^n-1)(2^n-1)}|^2 \end{bmatrix} \begin{bmatrix} \pi_0^{t-1} \\ \vdots \\ \pi_{2^n-1}^{t-1} \end{bmatrix}. \tag{11}$$

Applying this result iteratively m times using the associativity of matrix multiplication, we arrive at

$$\begin{bmatrix} \pi_0^m \\ \vdots \\ \pi_{2^n-1}^m \end{bmatrix} = \begin{bmatrix} |U_{00}|^2 & \cdots & |U_{0(2^n-1)}|^2 \\ \vdots & \cdots & \vdots \\ |U_{(2^n-1)0}|^2 & \cdots & |U_{(2^n-1)(2^n-1)}|^2 \end{bmatrix}^m \begin{bmatrix} \pi_0^0 \\ \vdots \\ \pi_{2^n-1}^0 \end{bmatrix}, \tag{12}$$

where π^0 is some initial distribution of network states. Allowing for the possibility of multiple initial state distributions, we can extend the above result into the following

$$\begin{bmatrix} \pi_{00}^m & \cdots & \pi_{0(2^n-1)}^m \\ \vdots & \cdots & \vdots \\ \pi_{(2^n-1)0}^m & \cdots & \pi_{(2^n-1)(2^n-1)}^m \end{bmatrix} = \begin{bmatrix} |U_{00}|^2 & \cdots & |U_{0(2^n-1)}|^2 \\ \vdots & \cdots & \vdots \\ |U_{(2^n-1)0}|^2 & \cdots & |U_{(2^n-1)(2^n-1)}|^2 \end{bmatrix}^m \begin{bmatrix} \pi_{00}^0 & \cdots & \pi_{0(2^n-1)}^0 \\ \vdots & \cdots & \vdots \\ \pi_{(2^n-1)0}^0 & \cdots & \pi_{(2^n-1)(2^n-1)}^0 \end{bmatrix}. \tag{13}$$

Now we can see that in order to program Zak's and Williams' QNN to a desired behavior, one must find a unitary matrix satisfying

$$\begin{bmatrix} |U_{00}|^2 & \cdots & |U_{0(2^n-1)}|^2 \\ \vdots & \cdots & \vdots \\ |U_{(2^n-1)0}|^2 & \cdots & |U_{(2^n-1)(2^n-1)}|^2 \end{bmatrix} = \lim_{m\to\infty} \left(\left(\begin{bmatrix} \pi_{00}^m & \cdots & \pi_{0(2^n-1)}^m \\ \vdots & \cdots & \vdots \\ \pi_{(2^n-1)0}^m & \cdots & \pi_{(2^n-1)(2^n-1)}^m \end{bmatrix} \begin{bmatrix} \pi_{00}^0 & \cdots & \pi_{0(2^n-1)}^0 \\ \vdots & \cdots & \vdots \\ \pi_{(2^n-1)0}^0 & \cdots & \pi_{(2^n-1)(2^n-1)}^0 \end{bmatrix}^{-1} \right)^{1/m} \right), \tag{14}$$

such that each distribution of initial states, forming a column vector in the rightmost matrix, converges in the long run to a desired distribution of final states, forming a column vector in the matrix second from the right.[1] Typically, these distributions of networks states will be deterministic basis states, i.e. there will be a single state with probability 1 and all others with probability 0. But more general probabilistic behavior is compatible with this framework, at least in principle. In associative memory, the desired state is the memorized pattern closest in Hamming distance to the initial state. In classification, the desired state is representative of the class to which the initial state belongs.

Of course, there are two big limitations to equation (14). Obviously, the matrix U must be unitary, which places a constraint on the possible values of $|U_{ij}|^2$. Additionally, the limit must actually exist, otherwise the desired convergence behavior is not possible. Even if these constraints are satisfied, finding the solution analytically is unlikely to be feasible. This is why in the field of classical ANNs, there are instead iterative approximation procedures that converge to acceptable solutions reasonably quickly, like backpropagation. There is no clear analogous procedure for the QNN architecture proposed by Zak and Williams, but they do offer a clever strategy which can reduce the dimensionality of the optimization problem that has to be solved to find the appropriate unitary matrix.

Suppose that H is some hamiltonian that is easy to implement on a timescale of about τ. Then according to the elementary solution of Schrodinger's equation, we can implement a unitary transformation $U = e^{iH\tau/\hbar}$ by exposing the quantum system to this hamiltonian for a period of time τ. Here, Zak and Williams apply some tool called the "Sylvester decomposition", for which they do not have a

[1] This result differs from equation (70) in Zak's and Williams' work. However, it seems correct to us.

citation, and which was too esoteric for us to reproduce. Doing so, they come to the following matrix equation

$$U^\tau = e^{iH\tau/\hbar} = \sum_{k=1}^{n} e^{i\lambda_k \tau/\hbar} \frac{\prod_{j \neq k}(H - \lambda_j I)}{\prod_{j \neq k}(\lambda_k - \lambda_j)}, \tag{15}$$

where λ_j are the n eigenvalues of the Hamiltonian. Zak and Williams point out that this decomposition implies that "any fixed Hamiltonian is a sum of n periodic functions with periods $2\pi/\lambda_k\tau$. If the eigenvalues are not commensurate, the behavior of the unitary matrix U^τ as a function of τ will be ergodic, i.e., this matrix eventually will take all possible values which result from all the possible combinations of its n eigenvalues $e^{i\lambda_j\tau/\hbar}$." We take this to mean that, if all of the eigenvalues are distinct, then the sum above acts as a sort of Fourier series, which has a range equal to the union of the ranges of each of the functions in the sum. It follows that just by varying the time parameter τ representing the duration which a fixed Hamiltonian is applied between measurements, we can select from "a set of unitary matrices which are equivalent to those obtained from variations of n independent parameters" [ZW98].

Zak and Williams suggest optimizing the τ parameter to achieve a desired unitary transformation U^* by minizing the following function, which they refer to as a "Liapunov function", although we recognize it as equivalent to simply minimizing the sum of the squared errors:

$$L = \frac{1}{2} \sum_{i,j=1}^{n} (|U^*_{ij}|^2 - |U_{ij}|^2)^2. \tag{16}$$

Zak and Williams acknowledge that this objective function can have more than one minimum (indeed, it seems guaranteed to have one minima per distinct eigenvalue), but they reason that "for a one-dimensional case, finding a global minimum is not a hard problem". We are not so convinced that this is an easy problem, although it seems correct that the simplication of the parameter space would make it easier to some extent.

4 Discussion

What we have described above is just one of several different quantum neural network frameworks proposed in the work of Zak and Williams. Their other proposals are generalizations of the one above, using either multiple correlated systems, each similar to equation (7), or one system exposed to an arbitrary interference vector which can be used to dynamically control the behavior of the QNN. These, too, are interesting, but we don't have space to consider them in this context. For now we consider the applications of Zak's and Williams' basic proposal to use quantum measurement as the activation function. We have seen that there exists some procedure by which their QNN can ostensibly be programmed to converge to a desired state over time. But the forward and backward pass algorithms, significantly, entail not knowing ahead of time what the desired final state is. To use the programming procedure proposed by Zak and Williams, one must know the entire set of input-state and output-state pairs ahead of time. This means that any use of Zak's and Williams' QNN proposal relating to the forward or backward pass algorithms must construct the unitary operator from information contained in the neural network (i.e. the weights matrix). Additionally, the ZW QNN proposal computes the activation state of the entire system simultaneously, which we know is incompatible with either of these algorithms, since they iterate through the network layer by layer.

It may be possible to adapt the ZW QNN proposal to these algorithms by using one measurement per layer in the ANN. In this case the unitary operator applied would vary from layer to layer, so that the hamiltonian applied to the quantum system would change after each measurement, which may make efficient physical implementation even more challenging. After applying all of the unitary operators corresponding to the network's layers and the performing the final measurement, the state of the quantum system will be a computational basis state representing the activation of the output layer.

While such a network would not be a universal function approximator because it necessarily simulates McCulloch-Pitts neurons, it may be useful for classification, as each computational basis state could be considered to correspond with a class. However, this network would not be as general as a classical feedforward network of McCulloch-Pitts neurons, simply because of the requirement that the matrix of weights connecting each layer to the one before it must be unitary. The practical implications of this constraint are not entirely clear and could be the subject of future research. The fact that the unitary matrix may contain complex numbers in general may be an advantage for this sort of quantum neural network. Given these facts, it is likely that this sort of network will retain at least some interesting behaviors, even if it is not able to be used for function approximation/regression. Note that in a more general model of quantum computation, the eigenbasis of the measurement operator may be continuous, meaning that this QNN framework could be generalized to have a continuous set of activation states, which opens up the possibility that it could be used for function approximation and regression. However, the unitarity requirement remains, and is likely to pose a problem and prevent the QNN from being a *universal* function approximator. Questions relating to these issues could be the subject of interesting further research.

How the backward pass could be implemented in this framework is less clear. Importantly, the backward pass requires the argument to each neuron's activation function during the prior forward pass (we used z to denote this information in our description of the algorithm). However, the framework we've just discussed destroys that information during measurement, and even if it did not, the information is not physically accessible since accessing that information would require obtaining the full description of the pre-measurement quantum state, a violation of the no-cloning theorem. Furthermore, it is not clear that the backward pass even makes sense in the context of quantum measurement as the activation function, because the derivative of the activation function is undefined. Therefore it seems that this proposal has some limited application in the layer-by-layer computation of the forward pass, but not much else. It seems that much of the difficulty in adapting quantum systems so that they perform computations relating to ANNs is the preservation and accessibility of necessary information. Each of the quantities referenced by the forward and backward pass algorithms, including the neuron activations, the derivative of the activation function, and the argument to the activation function of each neuron, among others, must be preserved by the forward pass and physically accessible. This means that they must be encoded into the unitary operator applied to the system or into an eigenstate of a measurement operator—both of which place a significant constraint on the values these variables can assume. Any advantages coming from this type of quantum implementation of a neural network seem likely to reduce to the ability of a quantum operation to perform matrix multiplication, other than novel interesting behaviors perhaps arising from interference and the possibility of complex-valued synaptic weights.

Although this particular proposal has only a small degree of compatibility with the forward and backward pass algorithms used on classical ANNs, there are some interesting questions relating to the advantages of quantum computing in this context if we assume that the forward and backward pass algorithms are given as quantum oracles. If we query a forward pass oracle on a quantum superposition of input vectors, is it possible to retrieve any useful information from the superposition of activation states which is returned? We know that a straightforward measurement of the returned state will simply collapse the state and return just a single activation state, but perhaps by applying some operations before measurement it is possible to compute some descriptive statistics of the set of activation states, which could be useful in a machine learning/data analysis setting. Similarly, if we query a backward pass oracle on a quantum superposition of activation states, is it possible to extract useful information from the superposition which is returned—for example, the sum of multiple gradient vectors is just as useful as single gradient vector, since it is equivalent to performing the backward pass on multiple activation states sequentially. Undoubtedly, a strong assortment of interesting questions remain in the intersection of artificial neural networks and quantum computing.

References

[Hec88] Robert Hecht-Nielsen. "Theory of the Backpropagation Neural Network". In: *Neural Networks* 1 (1988), pp. 445–448.

[Hor91] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural Networks* 4.2 (1991), pp. 251–257.

[ZW98] Michail Zak and Colin P. Williams. "Quantum Neural Nets". In: *International Journal of Theoretical Physics* 37.2 (1998), pp. 651–684.

[Qia99] Ning Qian. "On the momentum term in gradient descent learning algorithms". In: *Neural Networks* 12.1 (Jan. 1999), pp. 145–151.

[KLM07] Phillip Kaye, Raymond LaFlamme, and Michele Mosca. *An Introduction to Quantum Computing*. Oxford University Press, 2007.

[RB07] James Kennedy Riccardo Poli and Tim Blackwell. "Particle swarm optimization: An Overview". In: *Swarm Intelligence* 1 (2007), pp. 33–57.

[Ng11] Andrew Ng. *Sparse Autoencoder*. `http://web.stanford.edu/class/cs294a/sparseAutoencoder2011new.pdf`. 2011.

[SSP14] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. "The quest for a Quantum Neural Network". In: *Quantum Information Processing* 13.11 (Nov. 2014), pp. 2567–2586.