

Quantum perceptron: the basis of quantum neural networks

Noah Hellen

(Dated: March 15, 2024)

Quantum neural networks (QNNs) have been proposed as a novel computing paradigm, rooted in the domain of quantum computing. QNNs draw inspiration from artificial neural networks (ANNs), their classical equivalent. Theoretical advances in the study of variational quantum algorithms (VQAs) underpin the modern understanding and design of QNNs. In this paper, we will cover the quantum perceptron, a promising candidate under the umbrella of VQAs, which could be the building block for more sophisticated QNNs. Specifically, the paper will address the quantum circuit and variational algorithm used in designing the quantum perceptron. A quantum perceptron implementing four qubits will be discussed, serving as a practical demonstration of the proposal. Moreover, the learning capabilities of the quantum perceptron will be covered. In particular, a quantum perceptron using two qubits will be shown to effectively recognise patterns in data. The paper will conclude by covering how an efficient learning rule and continuously-valued data can be used to improve the model.

I. INTRODUCTION

Artificial neural networks gained prominence in the 20th Century for their ability to recognise structures in data sets. The structure of ANNs model the brain in abstract, with individual components designed as highly simplified neurons.

Feed-forward neural networks (FFNNs) served as the first architecture capable of performing the underlying capabilities of ANNs. Several other architectures, notably recurrent neural networks (RNNs) and Boltzmann machines, further expanded their capabilities.

The success of ANNs, paired with the advent of quantum computing in the 1990s, spurred activity and engagement in an exciting confluence of the two fields, aptly known as quantum neural networks. Quantum computing utilises uniquely quantum phenomena to solve certain problems faster than possible through classical techniques, and forms the framework from which QNNs are built. The publication of [1] by S. Kak marked the beginning of the ongoing research into this field.

The advent of Noisy Intermediate-Scale Quantum (NISQ) devices has made experimental realisation of QNNs possible [2]. Demand for new computational paradigms due to the slowdown of the phenomenon described in Moore's law [3, 4], and ever-increasing computational requirements [5] provides additional motivation for research in this field.

Following the publication of [1], the proposals for designing QNNs largely focused on quantum circuit modelling and the quantum perceptron [6, 7], which closely mirrors the current research. These proposals were not complete, and encountered challenges largely in the transition from the classical to quantum regime. In particular, non-linearity failed to be addressed properly due to the linear dynamics of quantum computing [6].

While there is not yet a unified proposal for QNNs, the main strand of research that represents the current state of the field is that of variational quantum algorithms. VQAs are a class of hybrid quantum-classical algorithms, consisting of a combination of parameterised quantum circuits and classical optimisation methods [8]. They are seen as a leading strategy for quantum advantage in several applications, of which QNNs are a subclass [9]. Several QNN architectures

derived from their classical equivalents have been proposed, relying heavily on this model [5].

Current research has amended the flaws of initial proposals but has been faced with several new challenges. The barren plateau (vanishing gradient) phenomenon presents scalability problems, where the gradient of the loss function vanishes exponentially in input size [10, 11]. Hardware noise on NISQ devices also leads to this phenomenon [9, 12]. Non-convexity of the cost function landscape of VQAs also poses a problem for scalability, whereby sub-optimal local minima are found via backpropagation [13]. Attempts to reconcile these scalability problems have been proposed, such as dampening the barren plateau effect with layer-wise learning [14].

Classical neural networks are derived from a single artificial neuron. A promising approach for research is to borrow this intuition for constructing QNNs using the theoretical groundwork of VQAs. In contrast to the classical case, simple implementation of non-linear activation functions is not possible in the linear dynamics of the quantum regime. The perceptron is a classical artificial neuron, distinct in its use of sign or binary step function as its activation function (see Fig. 2). The quantum equivalent of this design has therefore been studied, and has become a mature field of study [15], informing current research in developing QNNs [16].

This work will provide an in depth proposal for the design of the quantum perceptron, drawing from the work spearheaded by Tacchino and Mangini, as well as the necessary requisite material. Firstly, sections II, III, IV, and V will describe the classical background in ANNs that QNNs draw from. Next, the necessary understanding of quantum computing will be covered in section VI. The focus of this paper will cover the precise design of the quantum perceptron in section VII, as well as providing an example of its learning capabilities.

II. ARTIFICIAL NEURON

McCulloch and Pitts were the first to propose the idea of a mathematical abstraction of biological neural networks [17]. The design of the McCulloch-Pitts neuron emerged from their work, and it has served as the foundation for

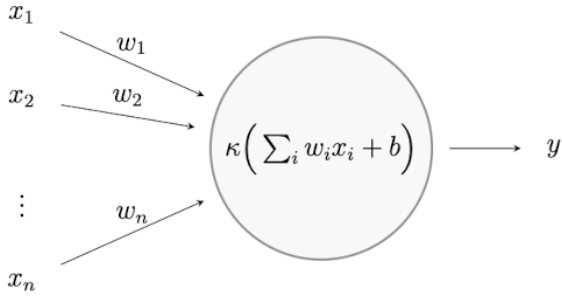


FIG. 1. Abstract model of an artificial neuron, producing an activation y from an input vector x [2].

contemporary neural networks. After several revisions to this model, it was understood that connecting these artificial neurons in a network could drastically increase computational power and ability [18].

An artificial neuron is designed to accept an input vector $x \in \mathbb{R}^n$ to produce a single output $y \in [0, 1]$, which is termed as the activation of the neuron. We also assign a weight vector $w \in \mathbb{R}^n$ to our input vector. An additional bias $b \in \mathbb{R}$ is assigned to the neuron. The weight vector and bias, as will be explained in section IV, are important for learning. In order to produce the desired output y , each neuron is equipped with an activation function κ , defined as

$$y = \kappa(z) = \kappa\left(\sum_i w_i x_i + b\right). \quad (1)$$

A pictorial representation of an artificial neuron is shown in Fig. 1.

The choice of activation function κ is crucial for effective ANNs. Activation functions determine the degree to which an artificial neuron is activated, and hence the degree to which an input is considered relevant. Modern neural networks typically use the sigmoid function (Fig. 2), because it is more suitable in learning tasks when compared with others [19]. The activation functions described in (Fig. 2) possess unique properties, which are exploited in different ANNs. The Rectified Linear Unit (ReLU), for example, is an activation function most commonly used in convolutional neural networks (CNNs) and restricted Boltzmann Machines, and has also shown to solve the problem of vanishing gradients in deep neural networks [20].

III. ARTIFICIAL NEURAL NETWORKS

Understanding the architectures and problem categories of modern ANNs is necessary to properly inform the design of QNNs. Feed-forward neural networks (FFNNs) were the first multi-layer networks constructed after the advent of the artificial neuron, and exhibit the simplest design [2]. Such networks consist of $L + 2$ layers of interconnected neurons, occupying so-called input, hidden and output layers, representing the direction of propagation. Every neuron in the

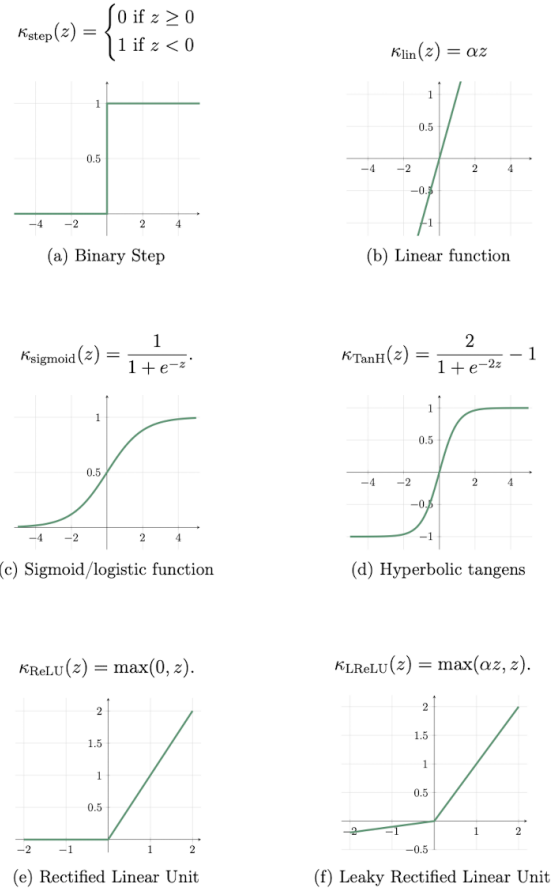


FIG. 2. A selection of activation functions most typically used in modern ANNs [2].

network has its own weight vector and bias in accordance with Eq. 1. Neurons occupying hidden and output layers receive the outputs of the previous layer. The degree of connectivity between neurons is problem-dependent. A representation of FFNNs is shown in Fig. 3.

Numerous other architectures have been designed since the advent of FFNNs, since the components and design of ANNs are inherently variable. This has led to ANNs becoming versatile in a number of problem categories. Preliminary models were restricted to linear classification tasks, though contemporary models are now capable of cluster analysis, forecasting, and image generation to name a few [18].

IV. LEARNING

ANNs owe their utility in their ability to recognise patterns and structures in data - they are able to learn from data. Modern ANNs are capable of supervised, unsupervised and reinforcement learning [19]. In supervised learning, each input x has an associated target value $t \in \mathbb{R}$, and the model is concerned with learning the input-output map. Unsupervised learning aims to find patterns in data without such intervention. Reinforcement learning is a framework in which a system learns by interacting with an environment or data.

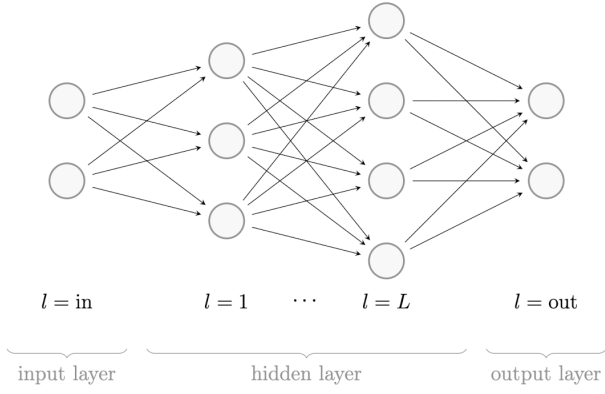


FIG. 3. A schematic representation of a feed-forward neural network, with circles representing neurons and arrows the connections between them. The number of hidden layers in the network is denoted with L [2].

The models considered in this paper are trained solely by supervised learning, and so the discussion will be restricted to problems of this nature.

Consider a data set $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(s)}\} \cup \{\mathbf{x}^{(s+1)}, \dots, \mathbf{x}^{(m)}\}$ that we wish to assign a pattern to. In supervised learning, a subset of these data points $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(s)}\}$ must be designated as *training data*, with an associated target value $t^{(j)}$ for $j \in [1, s]$. We describe the remaining data points $\{\mathbf{x}^{(s+1)}, \dots, \mathbf{x}^{(m)}\}$ as *unseen data*, used only for validation [2]. It is then precisely the task to find the optimal solution of the input-output mapping. In the case of ANNs, this is achieved through the use of variable parameters and a learning rule.

The variable parameters have previously been described as weight vectors in Eq. 1, and are varied in the training process. The learning rule is described as minimising a certain cost function (see Eq. 3) associated with the ANN, with the aim of finding the ideal set of weight vectors and biases. This learning rule is robust, and is suitable for arbitrary network depth and complexity.

Upon initialisation, an ANN is coupled with a random set of weight vectors and biases for which it is tasked with finding the optimal values. Given an input vector $\mathbf{x}^{(j)}$ in the training data, each iteration of the network will produce an output $y^{(j)}$, with which we compare to its target value $t^{(j)}$. It is assumed for simplicity that the output layer (see Fig. 3) contains one neuron, and $L = 0$. The loss function $\mathcal{L}(y^{(j)}, t^{(j)})$ is defined as

$$\mathcal{L}(y^{(j)}, t^{(j)}) = \frac{1}{2}(y^{(j)} - t^{(j)})^2, \quad (2)$$

and quantifies the difference between these values [21]. We use the mean squared error (MSE) for our loss function, although viable alternatives exist [2].

By averaging across all training examples $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(s)}\}$, we define the cost function as

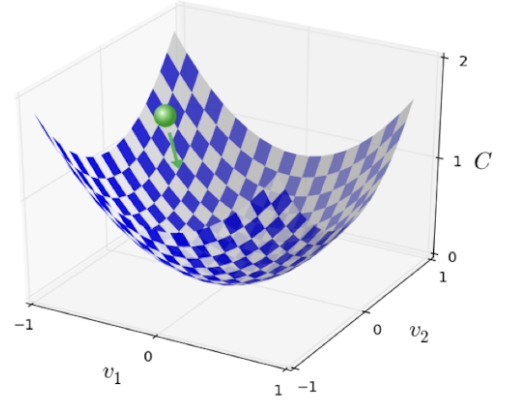


FIG. 4. Cost function landscape with $\mathbf{w} \in \mathbb{R}^2$. The green arrow represents the iterative process of gradient descent to locate the global minimum, and green sphere the current iteration [19].

$$\mathcal{E}(w_1, \dots, w_n, b) = \frac{1}{s} \sum_{j=1}^s \mathcal{L}(y^{(j)}, t^{(j)}). \quad (3)$$

By minimising this cost function, we extract the optimal values for weights and biases. This minimisation is achieved through gradient descent, in which the global minimum of the cost function is found. In practice, cost functions may be non-convex, and exhibit multiple local minima, resulting in a sub-optimal set of parameters.

Gradient descent informs us in which direction we must move in the cost function landscape to find our optimal parameters. A schematic diagram representing the process of gradient descent is demonstrated in Fig. 4. By identifying the gradient of the cost function, the parameters \mathbf{w} and b can be adjusted in the direction of this gradient [2, 22]. The learning rate $\eta \in [0, 1]$ controls the speed of iteration, and hence allows the gradient descent learning rule to be fully defined:

$$\begin{aligned} w_i &\leftarrow w_i - \eta \nabla_{w_i} \mathcal{E} \\ b &\leftarrow b - \eta \nabla_b \mathcal{E}. \end{aligned} \quad (4)$$

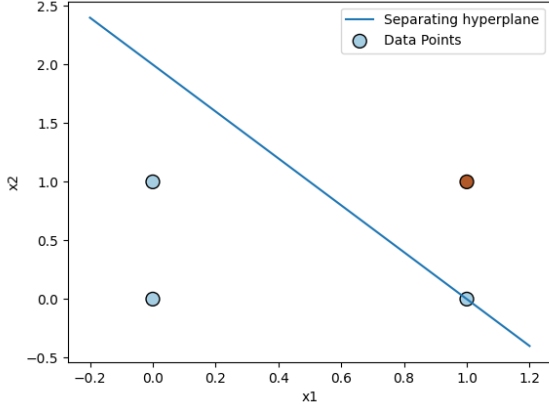
This learning rule is iterated until convergence such that the optimal parameters are found, and the learning task has been achieved. Computing these gradients practically is achieved by an algorithm known as backpropagation, of which a rigorous derivation is provided in [19].

V. PERCEPTRON

The quantum perceptron is the focus of this paper, necessitating a comprehensive understanding of its classical equivalent. As such, we will restrict our discussion of multi-layered ANNs to the special case of the perceptron. The perceptron was formulated in 1958 by Frank Rosenblatt. As discussed in the Introduction I, the perceptron is an ANN

TABLE I. Input-output mapping of boolean AND operator.

x_1	x_2	t
0	0	0
0	1	0
1	0	0
1	1	1

FIG. 5. Graphical representation of a perceptron classifying the boolean AND operator. See Table I for data points plotted. Blue circles represent $t = 0$ and red $t = 1$.

with $L = 0$, acting with a binary step function (see Fig. 2). The learning rule associated with the perceptron can be derived from the gradient descent method in Eq. 4. We first restrict the training to update parameters with a single example at a time: a process known as stochastic gradient descent [22]. By considering each weight co-ordinate separately w_i for $i \in [1, n]$, and one input $\mathbf{x}^{(j)}$ at a time, we are able to define the perceptron learning rule:

$$w_i \leftarrow w_i - \eta(y^{(j)} - t^{(j)}). \quad (5)$$

As a simple example demonstrating the learning ability of a perceptron, consider the boolean AND operator [21]. The input-output mapping of this operator is shown in Table I. The perceptron will be tasked with correctly classifying these data points, creating two separate classes depending on their target values. A graphical representation is provided in Fig. 5 by plotting the input values in \mathbb{R}^2 . The optimal weights and bias found by iterated use of Eq. 5 are represented by the separating hyperplane shown in Fig. 5.

VI. QUANTUM COMPUTING

Quantum computing promises to utilise uniquely quantum phenomena to perform computations. Both quantum and classical computational regimes borrow from the same binary set $\{0, 1\}$ to represent information. The linearity of solutions allowed in the quantum regime distinguishes information as being represented as qubits. The qubit is a two-dimensional quantum system, such that an arbitrary pure

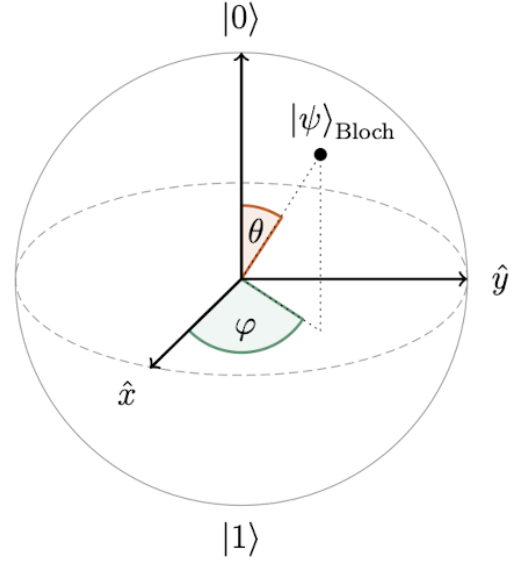


FIG. 6. Arbitrary qubit states can be represented as points on the surface and inside the Bloch sphere [2].

state can be represented as

$$|\psi\rangle = a|0\rangle + b|1\rangle, \quad (6)$$

with $a, b \in \mathbb{C}$ and normalised such that $|a|^2 + |b|^2 = 1$. By noting the normalisation condition, the qubit can be parameterised in terms of $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$, and is instead written in the form

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle. \quad (7)$$

Therefore, a pure qubit state is described as points on the sphere S^2 , which is called the Bloch sphere [23]. Mixed qubit states are described as points inside the Bloch sphere. A diagram of the Bloch sphere is shown in Fig. 6. Qubits will also be described as members of the Hilbert space $\mathcal{H} = \mathbb{C}^2$.

Meaningful computational power requires longer representations of information. This is achieved classically by bit concatenation, forming strings of bits. In quantum systems, we achieve this result by use of the tensor product. A system of N coupled qubits inhabits a 2^N -dimensional Hilbert space $\mathcal{H} = \mathbb{C}^{2^N}$. This system is described by

$$|q_{N-1}\rangle \otimes \cdots \otimes |q_1\rangle \otimes |q_0\rangle = |q_{N-1} \cdots q_1 q_0\rangle. \quad (8)$$

We describe the span of the Hilbert space $\mathcal{H} = \mathbb{C}^{2^N}$ as the computational basis.

Classically, logic gates are required to manipulate information. In the quantum regime, we require quantum gates. Quantum gates are unitary operators U , and in the single-qubit case, can be viewed as rotations in the Bloch sphere. It is common to describe quantum gates as $N \times N$ matrices acting on N qubits. Universal quantum computation states that any quantum algorithm can be decomposed into a finite set of quantum gates [23]. This is a useful result in the context of QNNs and the quantum perceptron, where complex











Operator	Gate(s)	Matrix
Pauli-X (X)	 \oplus	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

FIG. 7. A selection of single-qubit and multi-qubit quantum gates, with their corresponding matrices [24].

unitaries U are prevalent. The quantum gates useful in the discussion of the quantum perceptron are shown in Fig. 7.

The quantum circuit model of quantum computing is similar to the classical case, providing a visual representation of quantum computation and the order of operation. The quantum circuit model includes auxiliary qubits, denoted as *ancillary* qubits. These qubits are initialised in a well-defined state and independent from the input. Ancillary qubits are used to store results from the circuit, and will be of importance in the circuit model of the quantum perceptron. We will also briefly recap notational subtleties in quantum circuit diagrams to avoid confusion in the next section.

A string of length n of classical bits is represented as $v = v_{n-1}v_{n-2} \cdots v_0$. This notation is the same for N qubits, that are represented as $|q_{N-1}\rangle|q_{N-2}\rangle \cdots |q_0\rangle$. On a quantum circuit diagram, qubits will be ordered as in Fig. 8.

We will also briefly cover another notational subtlety, which is used for efficiently describing qubit states. Qubit states will be described using their corresponding bit string. For example, the qubit state $|1011\rangle$ will be written as $|11\rangle$, since 1011 in binary is equivalent to 11.

VII. QUANTUM PERCEPTRON

This section will provide an in depth proposal for the quantum perceptron, as well as its proficiency in learning tasks such as pattern recognition. The article [15] by Tacchino and other authors is the primary source for the design of the quantum perceptron covered in this paper. In this model,

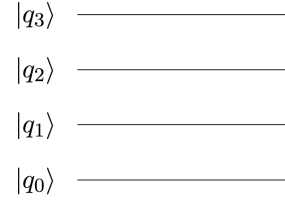


FIG. 8. Quantum circuit diagram with $N = 4$ qubits, demonstrating ordering of qubits in a quantum circuit. Image obtained and edited from [23].

the components of input and weight vectors are restricted to binary values $\{-1, 1\}$ as in the case of the McCulloch-Pitts neuron. Moreover, the bias has been excluded for simplicity.

The overall goal of a quantum perceptron is no different than in the classical case. Given an input set $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(s)}\} \cup \{\mathbf{x}^{(s+1)}, \dots, \mathbf{x}^{(m)}\}$ of training and unseen data with $\mathbf{x}^{(j)} \in \{-1, 1\}^n$, we wish to find the weight vector $\mathbf{w} \in \{-1, 1\}^n$ that satisfies the training criteria. In this discussion, we will only consider training data.

As noted in section I, the activation function of a classical perceptron is the binary step function, with 0 and 1 representing de-activated and activated perceptrons respectively. The equivalent activation function of the quantum perceptron will be represented as the qubit states $|0\rangle$ and $|1\rangle$. The key difference, however, is that the quantum perceptron finds a quadratic probability of activation. As such, non-linearity, a stringent requirement for developing mature QNNs, will be achieved.

To replicate the structure of a perceptron as in Fig. 1, the quantum perceptron will aim to compute the inner product between $\mathbf{x}^{(j)}$ and \mathbf{w} . This will be used to define the activation of the perceptron. The design of the quantum algorithm and circuit for this is the focus of VII A. Varying this circuit to be able to include all weight and input vectors will be the focus of VII B. Completion of these tasks will fully define the quantum perceptron.

The task of a well-defined learning rule as explained in IV for the classical case will be discussed in the final section VIII, highlighting the motivation and scope for future research.

We first define how to represent classical data in the quantum regime. This is achieved by encoding the input and weight vectors as coefficients on the quantum states $|\psi_{\mathbf{x}^{(j)}}\rangle$ and $|\psi_{\mathbf{w}}\rangle$, defined as:

$$|\psi_{\mathbf{x}^{(j)}}\rangle = \frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} x_i^{(j)} |i\rangle; \quad |\psi_{\mathbf{w}}\rangle = \frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} w_i |i\rangle. \quad (9)$$

The variable $x_i^{(j)}$ in this definition signifies that we are iterating through the i components of an arbitrary n -dimensional input vector $\mathbf{x}^{(j)}$ of the input set. The states $|i\rangle$ form the computational basis of the n -dimensional Hilbert

space. Therefore, we must use N qubits in the computation, such that $2^N = n$. It is now the task of computing the scalar product between these states, and finding the respective activation, by designing a specialised quantum circuit of N qubits.

A. Quantum circuit design

We prepare the system with N qubits each in the state $|0\rangle$, and a single ancillary qubit $|a\rangle$ also initialised in the state $|0\rangle$. The first step in the algorithm is to design the U_x operator, which encodes $|\psi_{x(j)}\rangle$ into the circuit. This is achieved by first entangling the N qubits with $H^{\otimes N}$ (see Fig. 10 for an example with $N = 4$). This creates a superposition of the computational basis states in the n -dimensional Hilbert space, allowing for maximal encoding of input and weight vectors. This is described by

$$H^{\otimes N}|0\rangle^{\otimes N} = \frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} |i\rangle \equiv |\psi_0\rangle. \quad (10)$$

Next, a set of Z and $C^p Z$ gates form the rest of U_x , where p is the number of qubits the gate acts on. The design and variability of these gates is precisely the aim of next section VII B. This step creates the state $|\psi_{x(j)}\rangle$ defined in Eq. 9 above. The combination of Z , $C^p Z$ and $H^{\otimes N}$ gates define U_x , and its operation can be neatly defined as

$$U_x|0\rangle^{\otimes N} = |\psi_{x(j)}\rangle. \quad (11)$$

The next stage of the algorithm imprints $|\psi_w\rangle$ onto $|\psi_{x(j)}\rangle$ with the unitary U_w . The required variability of U_w will also be fully defined in the next section VII B. For the moment, we can effectively define the action of U_w as the unitary rotating $|\psi_w\rangle$ as

$$U_w|\psi_w\rangle = |1\rangle^{\otimes N} = |n-1\rangle. \quad (12)$$

This definition is to provide a description of the form of U_w , as opposed to its action in the circuit. The action of U_w in the quantum circuit is to operate on $|\psi_{x(j)}\rangle$. This first imprints $|\psi_w\rangle$ on to $|\psi_{x(j)}\rangle$, before applying $H^{\otimes N}$ and $NOT^{\otimes N}$ gates. The resulting state is

$$U_w|\psi_{x(j)}\rangle = \sum_{i=0}^{n-1} c_i |i\rangle \equiv |\phi_{x(j),w}\rangle, \quad (13)$$

such that the scalar product is contained in the c_{n-1} coefficient of the state $|\phi_{x(j),w}\rangle$. We now aim to extract this information, by first considering the ancillary qubit $|a\rangle = |0\rangle$ with $|\phi_{x(j),w}\rangle$ (i.e. $|\phi_{x(j),w}\rangle \otimes |0\rangle$). We apply a $C^N NOT$ gate, which affects only the state $|1 \cdots 1\rangle \otimes |0\rangle$, creating the state $|1 \cdots 1\rangle \otimes |1\rangle$. The full effect of this gate is described as

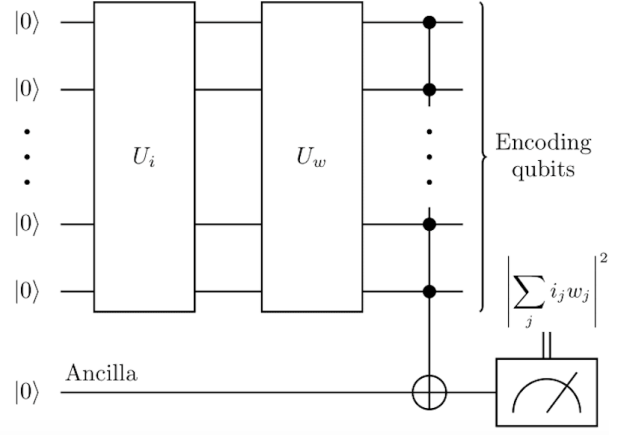


FIG. 9. Quantum circuit diagram of the quantum perceptron. The system is first initialised with N qubits. Next, the unitary operators U_x and U_w encode the input and weight vectors onto the quantum state. Finally, a $C^N NOT$ gate translates the desired result onto an ancillary qubit, which is measured to determine the activation of the quantum perceptron. The author uses different notation: U_i instead of U_x and i_j instead of $x_i^{(j)}$ [15].

$$C^N NOT|\phi_{x(j),w}\rangle \otimes |0\rangle = \sum_{i=0}^{n-2} c_i |i\rangle \otimes |0\rangle + c_{n-1} |n-1\rangle \otimes |1\rangle. \quad (14)$$

Measurement of the ancillary qubit reveals the probability $|c_{n-1}|^2$ that it is in the state $|1\rangle$. This corresponds to the probability of the quantum perceptron being activated. As a result of the activation function being quadratic, we achieve the requirement of non-linearity.

We briefly compare the standard scalar product between input and weight vectors with the effect of U_w . The unitary U_w can be summarised as the scalar product between $|\psi_{x(j)}\rangle$ and $|\psi_w\rangle$, as defined in

$$\langle \psi_w | \psi_{x(j)} \rangle = \langle \psi_w | U_w^\dagger U_w | \psi_{x(j)} \rangle = \langle n-1 | \phi_{x(j),w} \rangle = c_{n-1}. \quad (15)$$

Comparing Eq. 15 with the standard scalar product between $\mathbf{x}^{(j)}$ and \mathbf{w} , such that $\mathbf{x}^{(j)} \cdot \mathbf{w} = nc_{n-1}$, we find equivalence up to an overall normalisation factor. The point of divergence between classical and quantum perceptron models is defining an activation function with this scalar product. This can be seen when comparing the activation function of Fig. 1 with measurement of ancillary qubit in Fig. 9.

A pictorial representation of the quantum circuit design is shown in Fig. 9.

B. Variational model

As it currently stands, we have defined an architecture for the quantum perceptron. However, the current quantum

circuit is static in its ability to accept different inputs and weights. It is therefore necessary to define an algorithm for this purpose.

The algorithm designed to vary U_x and U_w is termed the hypergraph states generation subroutine (HSGS), due to the structure of the quantum gates implemented in the algorithm corresponding to an underlying hypergraph [25].

In practical terms, we first choose a pair of input and weight vectors $\mathbf{x}^{(j)}$ and \mathbf{w} to encode into the circuit. The HSGS procedure modifies the number and arrangement of Z and C^pZ gates in the quantum circuit, corresponding to the pair of vectors chosen.

This procedure follows an iterative approach, targeting each component of $\mathbf{x}^{(j)}$ and \mathbf{w} with a -1 factor. Components with a $+1$ are not targeted. For a circuit of N qubits, we reiterate that p is the number of qubits involved in the gate. The first states to check are the ones with only one qubit in the state $|1\rangle$, for example the state $|10\cdots 0\rangle$. If the coefficient of corresponding state is -1 , a Z gate is applied to that qubit. For $p = 2, \dots, N$, this procedure is once again iterated using C^pZ gates for the components of \mathbf{w} and \mathbf{x}_j requiring a -1 factor. Previous iterations may affect other states. For example, if both $|10\cdots 0\rangle$ and $|11\cdots 0\rangle$ require a -1 factor, the Z gate applied to $|10\cdots 0\rangle$ will also affect $|11\cdots 0\rangle$. This can be reconciled by applying the C^2Z (or C^pZ for the general case) gate twice. This procedure is carried out until all states have the required coefficients $\{-1, 1\}$.

The HSGS procedure is therefore able to vary the quantum circuit to include all possible input and weight vectors. A case study where $N = 4$ is shown in Fig. 10. An explicit proof demonstrating that the design of the circuit in Fig. 10 is correct is found in Appendix A.

It is noted that the state $|0\cdots 0\rangle$ is unaffected by the Z and C^pZ gates. This is a problem if the corresponding vector component required, $x_0^{(j)}$ or w_0 , is -1 . The resolution of this issue, involving the $X^{\otimes N}$ gate, is demonstrated in the example provided in Appendix A.

C. Pattern recognition

We have now defined the quantum perceptron. In light of the learning capabilities of the classical perceptron in section IV, the learning capabilities of the quantum perceptron will be demonstrated.

As noted at the beginning of this section, an efficient learning algorithm for the quantum perceptron is not the purpose of this paper. Therefore, we will consider low-dimensional vectors, such that the optimal weight vector(s) can be found by inspection. The learning task this section focuses on will be pattern recognition.

We first consider an efficient encoding scheme. The quantum perceptron model we have described accepts binary valued input and weight vectors. Therefore, we will use a string of bits of length n to define a set of weight and in-

put vectors. Consider the binary string $v_{input} = v_{weight} = v_{n-1}v_{n-2}\cdots v_0$. We can encode inputs and weight vectors such that $x_i^{(j)} = (-1)^{v_i}$ and $w_i = (-1)^{v_i}$ for $i \in [0, n-1]$.

Using the binary string, we can also represent input and weight vectors as monochromatic images, also known as binary images. This will provide a visual representation of input and weight vectors. The quantum perceptron will therefore act to find the similarities in images. As such, we assign $v_i = 0(1)$ to a white (black) pixel.

We will focus on the case of $N = 2$ qubits, or equivalently, a bit string of length $n = 4$. The binary images for a selection of bit strings is shown in Fig. 11. We will choose $v_{input} = 1001 = 9$ and $v_{weight} = 1100 = 12$, and determine if the v_{weight} can capture the pattern of v_{input} , based on the activation of the quantum perceptron. Using the encoding scheme, we translate these binary strings into vectors

$$\mathbf{w} = \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix} ; \quad \mathbf{x} = \begin{pmatrix} -1 \\ 1 \\ 1 \\ -1 \end{pmatrix}, \quad (16)$$

where we drop the notation \mathbf{x}_j since we only consider one input vector in this example. We can immediately see that the scalar product between these two vectors is 0, and so the perceptron activation is 0. The corresponding quantum perceptron circuit can be derived using the HSGS procedure. Therefore, the quantum perceptron effectively recognises that these two patterns (see Fig. 11) are not compatible. It can be found (see Fig. 13) that the optimal weight vectors for recognising the pattern of $v_{input} = 1001 = 9$ are $v_{weight} = 1001 = 9$ and $v_{weight} = 0110 = 6$, due to the high activation of the quantum perceptron.

An example of the quantum circuit for the quantum perceptron utilising the HSGS procedure is shown in Fig. 12 for $v_{input} = 11$ and $v_{weight} = 7$.

Scaling this system to even $N = 4$ is computational demanding, as there are 2^{32} possible combinations of input and weight vectors. However, an example of pattern recognition for $N = 4$ qubits is provided in Fig. 14. This therefore necessitates a learning rule, as will be discussed in the next section, to avoid iterating through all input and weight vectors.

VIII. CONCLUSION AND OUTLOOK

This paper has provided a comprehensive description of the design of the quantum perceptron, while first setting out the purpose and requisite material necessary for its study. In particular, the proposed design takes direct inspiration from the classical equivalent, and reproduces comparable results with higher computational efficiency. Achieving this required firstly a general quantum circuit capable of both computing the inner product between weight and input vectors, as well as providing a suitable means of reproducing the activation function as in the classical case. The scalar product was simply realised by the unitaries U_x and U_w , and

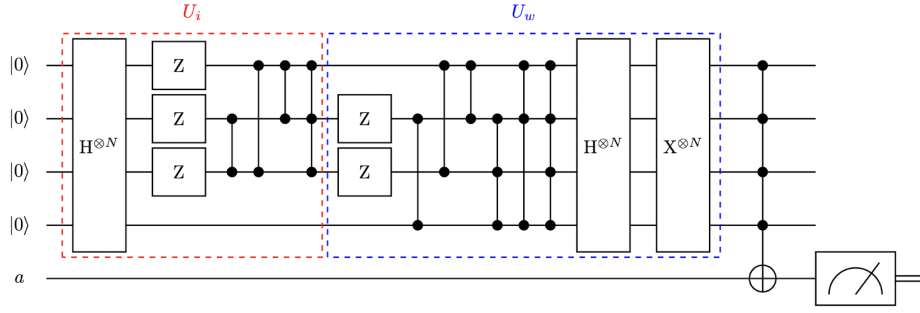


FIG. 10. Quantum circuit of an $N = 4$ quantum perceptron. The weight vector has components $w_2 = w_3 = w_4 = -1$ and $+1$ for all other components. The input vector has components $x_0^{(j)} = x_1^{(j)} = -1$ and $+1$ for all other components. The implementation of Z and C^pZ gates are a direct result of the HSGS procedure. Black vertical lines with black circles represent C^pZ gates, with the number of black circles corresponding to the p qubits involved. The author uses different notation: U_i instead of U_x [15].

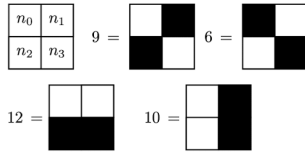


FIG. 11. Binary images for a bit string of length $n = 4$. White squares correspond to the bit 0, and black to the bit 1 [15]. The patterns for 9 and 6 are incorrect. Swapping these patterns provides the correction.

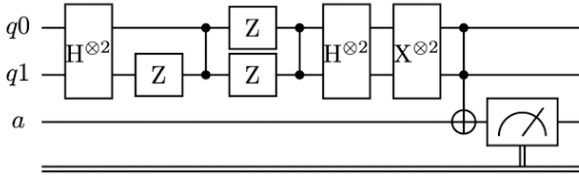


FIG. 12. Quantum perceptron circuit for implementing the bit strings $v_{input} = 11$ and $v_{weight} = 7$. The author orders $|q_0\rangle$ above $|q_1\rangle$, while this paper order them contrariwise. Note that the circuit is still correct, although to match the format of this paper, $|q_0\rangle$ should be flipped with $|q_1\rangle$ and the first Z gate moved to the top wire [15].

activation through measurement of an ancillary qubit. Implementation of the HSGS procedure provided the model with a means of varying U_x and U_w , effectively allowing for all input and weight vectors to be encoded. This model of the quantum perceptron is capable of pattern recognition tasks. This was demonstrated by the use of input binary images, with higher activation for images closer to the chosen weight binary image.

Since the release of [15] in 2019, the authors and other researchers have sought to improve this model. The proposed model is restricted to binary input and weight vectors. While this is useful for simplifying the model, it is restrictive in the number of problems it is able to solve. Moreover, there is a lack of a learning rule to find the optimal weight vector to satisfy the learning criteria. They have therefore worked to replace the binary input and weight vectors with continuously-valued vectors. This will allow for a larger class of problems to be solved, as well as allowing for the use of gradient descent based learning rules [8, 26]. More-

over, a learning algorithm using the framework of VQAs has been proposed as a method of identifying the optimal weight, and hence unitary U_w [8, 27].

Scalability of this quantum perceptron model is faced with the same challenges facing VQAs, as described in the Introduction I. Nevertheless, a proof of concept quantum feed-forward neural network constructed with quantum perceptrons has been proposed [16].

Regardless of any drawbacks or refinement required, the quantum perceptron proposed in this paper has informed current work, and is likely to be a crucial foundation in constructing mature quantum perceptron-based FFNNs in the future.

Appendix A: Proof of quantum perceptron circuit design for $N = 4$ qubits as shown in Fig. 10

We will provide an in depth discussion of the quantum circuit described in Fig. 10. This is firstly to provide a proof of the design of the circuit. Secondly, as mentioned in VII B, encoding $|00 \dots 0\rangle$ with a -1 factor is difficult due to the state unaffected by Z and C^pZ gates. Hence, this example provides the clear strategy for handling this problem. We will drop the notation $x^{(j)}$ for x since we only consider one input vector.

Fig. 10 provides the realisation of the quantum circuit design for the quantum perceptron with $N = 4$ qubits, such that the weight and input vectors are as follows:

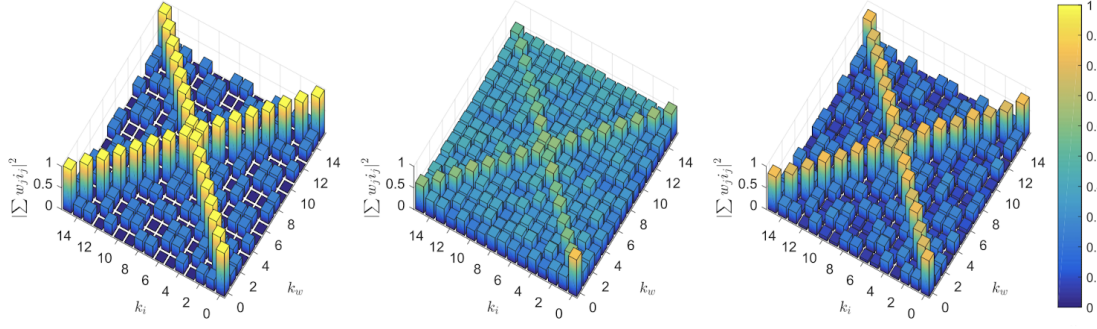


FIG. 13. Quantum perceptron activation for $N = 2$ qubits, demonstrated in different settings. The axis k_i and k_w represent the binary strings of length $n = 4$. The leftmost represents the ideal outcome of computing the scalar product between input and weight vectors, simulated on a classical computer. The rightmost represents the HSGS procedure. The middle represents a less optimal procedure, and has thus not been covered. The author uses different notation: i_j instead of $x_i^{(j)}$, k_i instead of v_{input} , and k_w instead of v_{weight} [15].

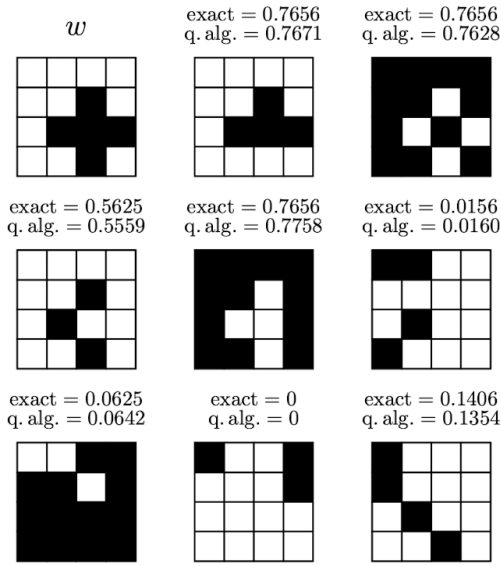


FIG. 14. Quantum perceptron circuit with $N = 4$ qubits. The decimal numbers represent the quantum perceptron activation of the input binary images when paired with the weight w , here represented as a binary image. Inverse images receive high activation due to the quantum perceptron activation being quadratic [15].

$$w = \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}; \quad x = \begin{pmatrix} -1 \\ -1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}. \quad (A1)$$

Standard computation of the scalar product, noting the normalisation factor n , returns $\frac{1}{n}(x \cdot w) = c_{n-1} = \frac{6}{16}$, such that $|c_{n-1}|^2 \approx 0.234$. We demonstrate this result is achieved by the quantum circuit design in Fig. 10.

As described, we will be encoding on to $N = 4$ qubits with one ancillary qubit. Firstly, as part of U_x , we implement a $H^{\otimes N}$ to the initial state of $|0000\rangle$, producing the superposition of computational basis states as in Eq. 10.

Next, in order to encode x into the quantum circuit, we employ use of the HSGS procedure, realised in the circuit by the use of Z and $C^p Z$ gates. The Z and $C^p Z$ gates operating on the qubits $|q_1\rangle$, $|q_2\rangle$, and $|q_3\rangle$ produce the result

$$|\psi_0\rangle \longrightarrow \frac{1}{\sqrt{16}} \left(\sum_{i=0}^1 |i\rangle - \sum_{i=2}^{n-1} |i\rangle \right) = -|\psi_x\rangle. \quad (A2)$$

Having successfully implemented the U_x unitary, it is now the task of implementing the U_w unitary. It can be seen that the series of Z and $C^p Z$ gates in U_w in Fig. 10, chosen as a result of the HSGS algorithm, transform the state $|\psi_w\rangle$ to $|\psi_0\rangle$ in accordance with Eq. 12. Operation with $H^{\otimes N}$ and $X^{\otimes N}$ will realise the full effect on $|\psi_w\rangle$ as required in Eq. 12. Therefore, implementing these gates on the state $-|\psi_x\rangle$ has the effect

$$-|\psi_x\rangle \longrightarrow \frac{1}{\sqrt{16}} \left(\sum_{i=0}^4 |i\rangle - \sum_{i=5}^{n-1} |i\rangle \right). \quad (A3)$$

We are now left with implementation of the $H^{\otimes N}$ and $X^{\otimes N}$ gates to complete U_w . The direct proof is computational exhaustive, and unnecessary for the purpose of this work. Instead, we focus on the qubit state $|1111\rangle$. After applying $H^{\otimes N}$, we find the coefficient of the $|0000\rangle$ state to be $-\frac{6}{16}$. Hence, by applying $X^{\otimes N}$, we are left with the state $-\frac{6}{16}|1111\rangle$. Using Eq. 14, and by considering the ancillary qubit, we are left with the state

$$|\phi_{x,w}\rangle \otimes |a\rangle = \sum_{i=0}^{n-2} c_i |i\rangle \otimes |0\rangle - \frac{6}{16} |n-1\rangle \otimes |1\rangle. \quad (A4)$$

By measuring the ancillary qubit, we find the activation of our perceptron to therefore be $|-\frac{6}{16}|^2 \approx 0.234$, in direct

agreement with the standard computation of the inner product.

This therefore completes the proof that the design of the circuit in Fig. 10 is correct, as it successfully computes the scalar product and finds the correct activation of the quantum perceptron.

In this example, we were tasked with adding a factor of -1

to the qubit state $|0000\rangle$. The quantum circuit designed in section VII and implemented in this example is incapable of directly adding this factor due to the nature of the Z and C^pZ gates. Hence, in this example, the workaround was finding $-\lvert\psi_x\rangle$ in Eq. A2, as opposed to $\lvert\psi_x\rangle$ as described in Eq. 11, which in turn was used to find the coefficient $c_{n-1} = -\frac{6}{16}$, of opposite parity to the standard computation. However, since only the square of this coefficient is of importance, the correct activation was found.

-
- [1] Kak S., Information Sciences, 1995, **83**, No. 3, 143–160, doi:[https://doi.org/10.1016/0020-0255\(94\)00095-S](https://doi.org/10.1016/0020-0255(94)00095-S), URL <https://www.sciencedirect.com/science/article/pii/002002559400095S>.
- [2] Beer K., Quantum neural networks, Ph.D. thesis, Leibniz U., Hannover, 2022, doi:10.15488/11896, 2205.08154.
- [3] Prati E., Rotta D., Sebastiano F., Charbon E., In: 2017 IEEE International Conference on Rebooting Computing (ICRC), 1–4, doi:10.1109/ICRC.2017.8123662.
- [4] Shalf J., Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 2020, **378**, 20190061, doi:10.1098/rsta.2019.0061.
- [5] Peral-García D., Cruz-Benito J., García-Peñalvo F. J., Computer Science Review, 2024, **51**, 100619, doi: <https://doi.org/10.1016/j.cosrev.2024.100619>, URL <https://www.sciencedirect.com/science/article/pii/S1574013724000030>.
- [6] Schuld M., Sinayskiy I., Petruccione F., Quantum Information Processing, 2014, **13**, No. 11, 2567–2586, doi:10.1007/s11128-014-0809-8, URL <http://dx.doi.org/10.1007/s11128-014-0809-8>.
- [7] Chakraborty S., Das T., Sutradhar S., Das M., Deb S., In: 2020 5th International Conference on Communication and Electronics Systems (ICES), 1395–1400, doi: 10.1109/ICES48766.2020.9137960.
- [8] Mangini S., Variational quantum algorithms for machine learning: theory and applications, Ph.D. thesis, Pavia U., 2023, 2306.09984.
- [9] Cerezo M., Arrasmith A., Babbush R., Benjamin S. C., Endo S., Fujii K., McClean J. R., Mitarai K., Yuan X., Cincio L., Coles P. J., Nature Reviews Physics, 2021, **3**, No. 9, 625–644, doi:10.1038/s42254-021-00348-9, URL <http://dx.doi.org/10.1038/s42254-021-00348-9>.
- [10] Zhao R., Wang S., A review of quantum neural networks: Methods, models, dilemma, 2021, 2109.01840.
- [11] McClean J. R., Boixo S., Smelyanskiy V. N., Babbush R., Neven H., Nature Communications, 2018, **9**, No. 1, 4812, doi:10.1038/s41467-018-07090-4, URL <https://doi.org/10.1038/s41467-018-07090-4>.
- [12] Wang S., Fontana E., Cerezo M., Sharma K., Sone A., Cincio L., Coles P. J., Nature Communications, 2021, **12**, No. 1, doi:10.1038/s41467-021-27045-6, URL <http://dx.doi.org/10.1038/s41467-021-27045-6>.
- [13] Rivera-Dean J., Huembeli P., Acín A., Bowles J., Avoiding local minima in variational quantum algorithms with neural networks, 2021, 2104.02955.
- [14] Skolik A., McClean J. R., Mohseni M., van der Smagt P., Leib M., Quantum Machine Intelligence, 2021, **3**, No. 1, doi: 10.1007/s42484-020-00036-4, URL <http://dx.doi.org/10.1007/s42484-020-00036-4>.
- [15] Tacchino F., Macchiavello C., Gerace D., Bajoni D., npj Quantum Information, 2019, **5**, No. 1, doi:10.1038/s41534-019-0140-4, URL <http://dx.doi.org/10.1038/s41534-019-0140-4>.
- [16] Tacchino F., Barkoutsos P., Macchiavello C., Tavernelli I., Gerace D., Bajoni D., Quantum Science and Technology, 2020, **5**, No. 4, 044010, doi:10.1088/2058-9565/abb8e4, URL <http://dx.doi.org/10.1088/2058-9565/abb8e4>.
- [17] McCulloch W. S., Pitts W., The Bulletin of Mathematical Biophysics, 1943, **5**, No. 4, 115–133, doi: 10.1007/bf02478259.
- [18] Basheer I., Hajmeer M., Journal of microbiological methods, 2001, **43**, 3–31, doi:10.1016/S0167-7012(00)00201-3.
- [19] Nielsen M. A., Neural networks and deep learning, 2018, URL <http://neuralnetworksanddeeplearning.com/>.
- [20] Ide H., Kurita T., In: 2017 International Joint Conference on Neural Networks (IJCNN), 2684–2691, doi: 10.1109/IJCNN.2017.7966185.
- [21] Grosse R., Intro to neural networks and machine learning, 2018, last accessed 12 March 2024, URL https://www.cs.toronto.edu/~rgrosse/courses/csc321_2018/.
- [22] Ahmed B., The perceptron algorithm, 2015, last accessed 12 March 2024, URL https://course.ccs.neu.edu/cs6140sp15/2_GD_REG_pton_NN/lecture_notes/lectureNotes_Perceptron.pdf.
- [23] Etzebarria I. G., Quantum computing epiphany term, 2024, last accessed 13 March 2024.
- [24] Wikipedia contributors, Quantum logic gate, 2024, [Online; accessed 13-March-2024], URL https://en.wikipedia.org/w/index.php?title=Quantum_logic_gate&oldid=1210238147.
- [25] Rossi M., Huber M., Bruß D., Macchiavello C., New Journal of Physics, 2013, **15**, No. 11, 113022, doi:10.1088/1367-2630/15/11/113022, URL <http://dx.doi.org/10.1088/1367-2630/15/11/113022>.
- [26] Mangini S., Tacchino F., Gerace D., Macchiavello C., Bajoni D., Machine Learning: Science and Technology, 2020, **1**, No. 4, 045008, doi:10.1088/2632-2153/abaf98, URL <http://dx.doi.org/10.1088/2632-2153/abaf98>.
- [27] Tacchino F., Mangini S., Barkoutsos P. K., Macchiavello C., Gerace D., Tavernelli I., Bajoni D., IEEE Transactions on Quantum Engineering, 2021, **2**, 1–10, doi: 10.1109/tqe.2021.3062494, URL <http://dx.doi.org/10.1109/tqe.2021.3062494>.

STATEMENT OF GENERATIVE AI

ChatGPT was used to provide a summary of the current state of research in the field of quantum computing, which helped to identify where the most promising strands of research lie. It was also used as a way of solving errors in the software, LaTeX, encountered in the writing process. Validity was evaluated by comparing the information provided by ChatGPT with the equivalent research found after its use.

LAY SUMMARY

The 20th century saw the shift from mechanical to electronic computers. Significant progress has been made to refine these computer systems, in an effort to improve computational power and efficiency. There are a large class of problems that we want computers to solve, some of which may be done sub-optimally. As a result, there has been extensive research into developing new computing paradigms to solve certain classes of problems faster than in classical computing systems. This led to the advent of artificial neural networks, a novel computing paradigm designed to mimic the learning process of the brain. Artificial neural networks are used extensively in modern computing, most notably in generative artificial intelligence programs which can be found in all the corners of the internet. Quantum computing is another computing paradigm that promises to solve certain problems more efficiently than in classical systems. This paper discusses the research in combining these two computing paradigms, known as quantum neural networks, by considering the fundamental unit with which more complex and powerful models will be developed from. Continual development of quantum computing and its ancillary disciplines could rival classical computer systems, possibly paving way for a new model of computation.