A Variational Algorithm for Quantum Neural Networks

A Variational algorithm for Quantum Neural Networks

Antonio Macaluso[1], Luca Clissa[2,3], Stefano Lodi[1], and Claudio Sartori[1]

[1] Dept. of Computer Science and Engineering, University of Bologna, Bologna, Italy
[2] Dept. of Physics and Astronomy, University of Bologna, Bologna, Italy
[3] Istituto Nazionale di Fisica Nucleare (INFN), Bologna, Italy
{antonio.macaluso2, luca.clissa2, stefano.lodi, claudio.sartori}@unibo.it

Abstract. Quantum Computing leverages the laws of quantum mechanics to build computers endowed with tremendous computing power. The field is attracting ever-increasing attention from both academic and private sectors, as testified by the recent demonstration of quantum supremacy in practice. However, the intrinsic restriction to linear operations significantly limits the range of relevant use cases for the application of Quantum Computing. In this work, we introduce a novel variational algorithm for quantum Single Layer Perceptron. Thanks to the universal approximation theorem, and given that the number of hidden neurons scales exponentially with the number of qubits, our framework opens to the possibility of approximating any function on quantum computers. Thus, the proposed approach produces a model with substantial descriptive power, and widens the horizon of potential applications already in the NISQ era, especially the ones related to Quantum Artificial Intelligence. In particular, we design a quantum circuit to perform linear combinations in superposition and discuss adaptations to classification and regression tasks. After this theoretical investigation, we also provide practical implementations using various simulation environments. Finally, we test the proposed algorithm on synthetic data exploiting both simulators and real quantum devices.

Keywords: Quantum AI · Quantum Machine Learning · Quantum Computing · Quantum Variational Algorithms · Machine Learning · Neural Networks

1 Background and Motivation

The field of quantum computing (QC) has recently achieved a historic milestone with quantum supremacy [1], thus attracting increasing interest and fostering future research. One of the topics in which QC may have a higher impact is Quantum Machine Learning (QML), i.e. a sub-discipline of quantum information processing whose intent is developing quantum algorithms that learn from data. However, the ability to deliver a significant boost in performance through quantum algorithms on near-term devices is still to be demonstrated. Given these premises, Neural Networks (NN) are among the most desired targets when

coming to transposing classical models into their quantum counterpart. In fact, NN have demonstrated remarkable performances in many real-world applications and multiple learning tasks, including clustering, classification, regression and pattern recognition.

In this work, we introduce a general model framework that reproduces a quantum state equivalent to the output of a classical Single Layer Perceptron (SLP). This is achieved by implementing an efficient variational algorithm that performs linear combinations in superposition. The results are then passed altogether through an activation function with just one application. Importantly, the framework supports pluggable activation function routines, thus allowing an easy way to adapt the approach to different use cases. In Section 2, we design a quantum circuit that generates a quantum SLP (qSLP) with two hidden neurons. Section 3 is devoted to practical experiments to test our model as a linear classifier. Finally, Section 4 describes how our approach can be extended to the case of more hidden neurons.

1.1 Quantum Variational Algorithms

The construction of full-scale, error-corrected quantum devices still poses many technical challenges. At the same time, significant progress has been made in the development of small-scale quantum computers, thus giving rise to the so-called Noisy Intermediate-Scale Quantum (NISQ) era. For this reason, many researchers are currently focusing on algorithms for NISQ machines that may have an immediate impact on real-world applications, e.g. chemistry [2] and optimisation [3,4].

Such machines, however, are still not sufficiently powerful to be a credible alternative to the classical ones. For this reason, *hybrid computation* was proposed to exploit near-term devices to benefit from the performance boost expected from quantum technologies. Quantum variational algorithms [5,6] represent the most promising attempt in this direction, and they are designed to tackle optimisation problems using both classical and quantum resources. The latter component is referred to as variational circuit, and it presents three ingredients: *i*) a parametrised quantum circuit $U(x;\theta)$, *ii*) a quantum output $f(x;\theta)$ and *iii*) an updating rule for the parameters θ.

The general hybrid approach is illustrated in Figure 1. The data, x, are initially pre-processed on a classical device to determine the input quantum state. The quantum hardware then prepares a quantum state $|x\rangle$ and computes $U(x;\theta)$ with randomly initialised parameters θ. After multiple executions of $U(x;\theta)$, the classical component post-processes the measurements and generates a prediction $f(x;\theta)$. Finally, the parameters are updated, and the whole cycle is run multiple times in a closed loop between the classical and quantum hardware.

Interestingly, the first practical demonstration of quantum advantage over classical supercomputers is related precisely to variational algorithms [7]. Other applications related to Machine Learning (ML) problems were also explored [8,9]. More recently, Schuld et al. [10] proposed a low-depth variational algorithm for

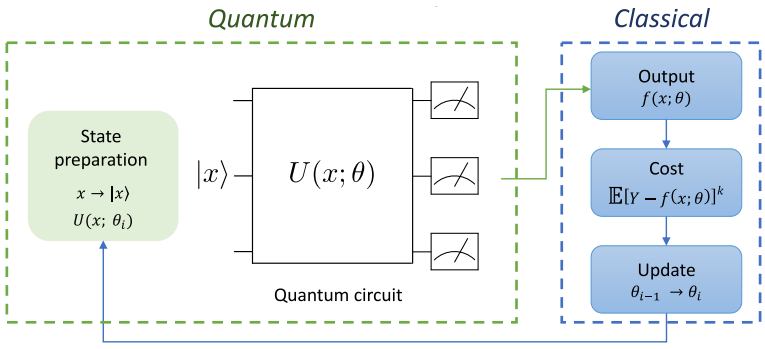


Fig. 1: Scheme of a hybrid quantum-classical algorithm for supervised learning. The quantum variational circuit is depicted in green, while the classical component is represented in blue.

classification. The strengths of this approach are two-fold. On one side, the possibility of learning gate parameters enables the adaptation of the architecture for different use cases. On the other hand, the choice of amplitude encoding allows obtaining model predictions with a single-qubit measurement. Importantly, simulations on standard benchmark datasets showed good performances, with the advantage of requiring fewer parameters than classical alternatives.

1.2 Neural Network as Universal Approximator

A Single Hidden Layer Neural Network (or Single Layer Perceptron - SLP) [11] is a two-stage model suitable for both classification and regression. Given a training point (x_i, y_i), the output of a feedforward NN with a single hidden layer containing H neurons can be expressed as:

$$f(x_i) = \sigma_{\text{out}} \left(\sum_{j=1}^{H} \beta_j \sigma_{\text{hid}} \left(L(x_i; \theta_j) \right) \right). \tag{1}$$

Each hidden neuron j computes a linear combination, $L(\cdot)$, of the input features $x_i \in \mathbb{R}^p$ with coefficients given by the p-dimensional vector θ_j. This operation is performed for all neurons, and the results are individually fed into the inner activation function σ_{hid}. The outputs of the previous operation are then linearly combined with coefficients β_j. Finally, a task-dependent outer activation function, σ_{out}, is applied.

Despite being more straightforward than the deep architectures proposed in recent years, the SLP model can be very expressive. According to the *universal*

approximation theorem [12], in fact, a SLP with a non-constant, bounded and continuous activation function can approximate any continuous function on a closed and bounded subset of $\mathbb{R}^n$, provided that enough hidden neurons are specified. In spite of this crucial theoretical result, SLP are rarely adopted in practice due to the unfeasibility of large amounts of hidden neurons on classical devices. Quantum computers, however, could leverage state superposition to scale the number of hidden neurons exponentially with the number of available qubits. Starting from these considerations, cleverly implementing a quantum SLP endowed with a proper activation function would therefore enable a real chance to benefit from the universal approximation property.

1.3 Related works

Several attempts for building a quantum perceptron unit were discussed in the literature [13,14,15]. A concrete implementation in near-term processors is illustrated in [16], where the authors introduced a model for binary classification using a modified version of the perceptron updating rule. A key characteristic of their architecture is the theoretical exponential advantage in storage resources over classical alternatives. This constitutes the first step towards the efficient implementation of quantum NN on near-term quantum processing hardware.

To the best of our knowledge, however, there are no trainable algorithms that efficiently reproduce a quantum state encoding the output of a classical SLP. Also, the available approaches rely on the introduction of severe constraints on the input data in order to reproduce non-linear activation functions, which makes the algorithms hardly useful in practice.

1.4 Contribution

In this work, we propose a new variational algorithm reproducing a quantum Single Layer Perceptron, whose output is equivalent to the classical counterpart. In particular, building on top of the approach described in [10], we design a general framework that allows efficient computation using just mild constraints on the input. Also, the flexible architecture enables to plug in custom implementations of the activation function routine, thus adapting to different use cases. Thanks to the possibility of learning the parameters for a given task, the proposed framework allows training models that can potentially approximate any function.

However, we do not address the problem of implementing a non-linear activation function. Our goal is to provide a framework that generates multiple linear combinations in superposition entangled with a control register. In this way, instead of executing a given activation function for each hidden neuron, a single application is needed to propagate it to all of the quantum states. This allows scaling the number of hidden neurons exponentially with the number of qubits, thus enabling the qSLP to be a concrete alternative for approximating complex and diverse functions.

2 Variational Algorithm for Single Hidden Layer Neural Network

2.1 Encode Data in Amplitude encoding

The first issue to address when using a quantum computer for data analysis is *state preparation*, i.e. the design of a process that loads the data from a classical memory to a quantum system. The most general encoding adopted in QML is amplitude encoding [17]. This strategy associates quantum amplitudes with real vectors of observations at the cost of introducing just normalisation constraints. Formally, a normalised vector $x \in \mathbb{R}^{2^n}$ can be described by the amplitudes of a quantum state $|x\rangle$ as:

$$|x\rangle = \sum_{k=1}^{2^n} x_k |k\rangle \longleftrightarrow x = \begin{pmatrix} x_1 \\ \vdots \\ x_{2^n} \end{pmatrix}. \tag{2}$$

In this way, it is possible to use the index register to indicate the k-th feature. The main advantage of this encoding is that we only need n qubits for a vector of $p = 2^n$ elements. This means that, if a quantum algorithm is polynomial in n, then it will have a polylogarithmic runtime dependency on the data size. A possible strategy for amplitude encoding has been proposed by Möttönen et al. [18], which is the one used for experiments in this work. The goal of this approach is to map an arbitrary state $|x\rangle$ to the ground $|0 \dots 0\rangle$. Once the circuit is obtained, then all of the operations are inverted and applied in the reversed order.

2.2 Activation function

The implementation of a proper activation function – in the sense of the Universal Approximation Theorem – is one of the major issues for building a complete quantum Neural Network. This is due to the restrictions to linear and unitary operations imposed by the laws of quantum mechanics [19]. The most promising attempt to solve this problem is described in [20], where the authors use the repeat-until-success technique to achieve non-linearity. The most significant limitation is the requirement of inputs in the range $\left[0, \frac{\pi}{2}\right]$, which is a severe constraint for real-world problems.

In this work, we do not discuss how to implement a non-linear activation function. However, we provide an framework that permits to train a quantum SLP for a given activation function Σ. Our architecture is naturally capable of incorporating any implementation of an activation function whose parameters are learned, like the one described in [21]. Indeed, we can think of extending the circuit that trains the qSLP to also learn the activation parameters. For this reason, new implementations of non-linear activation functions are naturally pluggable in the proposed framework as long as they fit in a learning paradigm.

2.3 Gates as Linear Operators

A variational circuit $U(\theta)$ is composed of a series of gates, each one possibly parametrised by a set of parameters $\{\theta_l\}_{l=1,\dots,L}$. Formally, $U(\theta)$ is the product of matrices:

$$U(\theta) = U_L \cdots U_l \cdots U_1, \tag{3}$$

where each U_l is composed of a single-qubit or a two-qubit quantum gate. In order to make the single-qubit gate trainable it is necessary to formulate U_l in terms of parameters that can be learned. This is possible by adopting a single-qubit gate G which is defined as the following 2×2 unitary matrix [22]:

$$G(\alpha, \beta, \gamma) = \begin{pmatrix} e^{i\beta} cos(\alpha/2) & e^{i\gamma} sin(\alpha/2) \\ -e^{-i\gamma} sin(\alpha/2) & e^{-i\beta} cos(\alpha/2) \end{pmatrix}. \tag{4}$$

Thus, we can now express each U_l in terms of single-qubit gates, G_i, acting on the i-th qubit:

$$U_l = \mathbb{1}_1 \otimes \cdots \otimes G_i \otimes \cdots \otimes \mathbb{1}_n, \tag{5}$$

where n is the total number of qubits of the quantum system. This representation of $U(\theta)$ is convenient since it allows computing the gradient analytically, as shown in [10].

Alternatively, we can express equation (4) using complex numbers $z, u \in \mathbb{C}$ instead of trigonometric functions:

$$G(z, v) = \begin{pmatrix} z & v \\ -v^* & z^* \end{pmatrix}, \tag{6}$$

where $|z|^2 + |v|^2 = 1$. This parametrisation avoids non-linear dependencies between the circuit parameters and the model output. Notice that the definition of linear operator given in Equation (6) involves complex coefficients. Therefore, it describes a more general operation with respect to the classical counterpart adopted in an SLP, that only allows for linear combinations with real-valued coefficient. Nonetheless, one can still parametrise the circuit using Pauli-Y rotation in case one wants to restrict the computation to the real domain.

2.4 Quantum Single Hidden Layer Network with two neurons

In this section we introduce the basic idea of a quantum Single Layer Perceptron with two neurons in the hidden layer. The generalisation of the algorithm is then discussed in section 4.

Intuitively, a qSLP can be implemented into a quantum computer in two steps. Firstly, we generate different linear operations in superposition, each one

having different parameters θ_j, entangled with a control register. Secondly, we propagate the activation function to all the linear combinations in superposition. Notice that, thanks this approach, instead of executing a given activation function for each hidden neuron, we need only one application to obtain the output of all the neurons in the hidden layer. To this end, three quantum registers are necessary: *control*, *data* (denoted by $|\psi\rangle$) and *temporary* register ($|\phi\rangle$). The latter is responsible for generating the linear combinations of the input data in superposition. Also, it can be in any arbitrary state, possibly even unknown.

The algorithm is composed of five main steps: *state preparation*, *entangled linear operators in superposition*, *application of the activation function*, *read-out step*, *post-processing*.

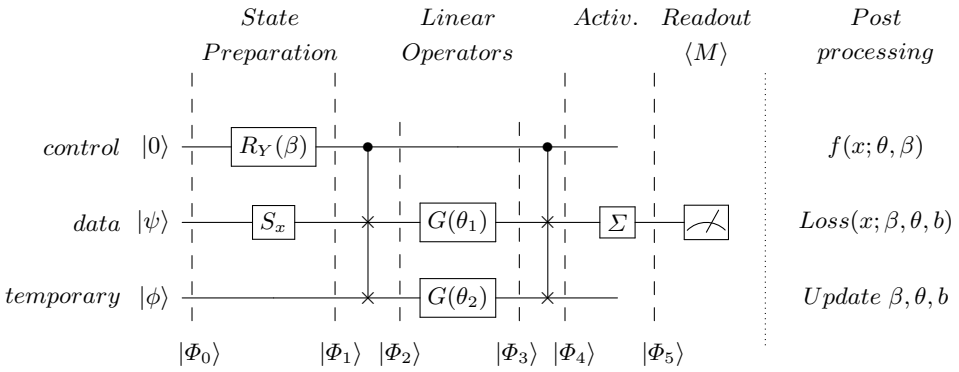


Fig. 2: Quantum circuit for training a qSLP.

(Step 1) The state preparation includes encoding the data, x, in the amplitude of $|\psi\rangle$ and applying a parametrised Y-rotation $R_y(\beta)$ to the control qubit:

$$\begin{aligned}|\Phi_1\rangle &= \big(R_y(\beta)\otimes S_x\otimes \mathbb{1}\big)\,|\Phi_0\rangle = \big(R_y(\beta)\otimes S_x\otimes \mathbb{1}\big)\,|0\rangle\,|0\rangle\,|\phi\rangle \\ &= (\beta_1\,|0\rangle + \beta_2\,|1\rangle)\otimes|x\rangle\otimes|\phi\rangle = \beta_1\,|0\rangle\,|x\rangle\,|\phi\rangle + \beta_2\,|1\rangle\,|x\rangle\,|\phi\rangle),\end{aligned} \tag{7}$$

where S_x indicates the routine that encodes the data, $|\beta_1|^2 + |\beta_2|^2 = 1$ and $\beta_1, \beta_2 \in \mathbb{R}$.

(Step 2) We exploit the idea of quantum forking [23] to generate two different linear operations in superposition, each entangled with the control qubit.

2.1 The first controlled-swap is applied to swap $|x\rangle$ with $|\phi\rangle$ if the control qubit is equal to $|1\rangle$:

$$|\Phi_2\rangle = \frac{1}{\sqrt{E}}\Big(\beta_1\,|0\rangle\,|x\rangle\,|\phi\rangle + \beta_2\,|1\rangle\,|\phi\rangle\,|x\rangle\Big) \tag{8}$$

where E is a normalisation constant.

2.2 Two linear operations parametrised by two different sets (θ_1 and θ_2) act on $|\psi\rangle$ and $|\phi\rangle$ respectively:

$$\begin{aligned}|\Phi_3\rangle &= \Big(\mathbb{1} \otimes G(\theta_1) \otimes G(\theta_2)\Big)\,|\Phi_2\rangle \\ &= \frac{1}{\sqrt{E}}\Big(\beta_1\,|0\rangle\,G(\theta_1)\,|x\rangle\,|\phi\rangle + \beta_2\,|1\rangle\,|\phi\rangle\,G(\theta_2)\,|x\rangle\Big) \\ &= \frac{1}{\sqrt{E}}\Big(\beta_1\,|0\rangle\,|L(x;\theta_1)\rangle\,|\phi\rangle + \beta_2\,|1\rangle\,|\phi\rangle\,|L(x;\theta_2)\rangle\Big). \end{aligned} \tag{9}$$

2.3 Then, the second controlled-swap is executed to swap $|L(x;\theta_2)\rangle$ with $|\phi\rangle$ if the control qubit is equal to $|1\rangle$:

$$|\Phi_4\rangle = \frac{1}{\sqrt{E}}\Big(\beta_1\,|0\rangle\,|L(x;\theta_1)\rangle\,|\phi\rangle + \beta_2\,|1\rangle\,|L(x;\theta_2)\rangle\,|\phi\rangle\Big). \tag{10}$$

Finally, the two linear operations are stored in $|\psi\rangle$ and are then entangled with one state of the control qubit. At this point, a routine is necessary to propagate the activation function in both the trajectories of $|\psi\rangle$.

(Step 3) Activation function:

$$\begin{aligned}|\Phi_5\rangle &= \Big(\mathbb{1} \otimes \Sigma \otimes \mathbb{1}\Big)\,|\Phi_4\rangle \\ &= \frac{1}{\sqrt{E}}\Big(\beta_1\,|0\rangle\,\Sigma\,|L(x;\theta_1)\rangle\,|\phi\rangle + \beta_2\,|1\rangle\,\Sigma\,|L(x;\theta_2)\rangle\,|\phi\rangle\Big) \\ &= \frac{1}{\sqrt{E}}\Big(\beta_1\,|0\rangle\,\big|\sigma_{hid}\big[L(x;\theta_1)\big]\big\rangle\,|\phi\rangle + \beta_2\,|1\rangle\,\big|\sigma_{hid}\big[L(x;\theta_2)\big]\big\rangle\,|\phi\rangle\Big). \end{aligned} \tag{11}$$

At the end of Step 3 the two linear operations, $L(\cdot)$, are put through the same activation function, σ_{hid}, represented by the gate Σ. The results are then encoded in the quantum register $|\psi\rangle$. Each output is finally weighed by the parameters of the control qubit (β), i.e. the coefficients attached to the hidden neurons in the linear combination that produces the output of the NN. This is exactly the quantum version of the two-neurons classical SLP presented in Equation (1).

(Step 4) The measurement of $|\psi\rangle$ can be expressed as the expected value of the Pauli-Z operator acting on the quantum state $|x\rangle$:

$$\langle M\rangle = \langle\Phi_0|\,U^\dagger(\beta,\theta)(\mathbb{1}\otimes\sigma_z\otimes\mathbb{1})U(\beta,\theta)\,|\Phi_0\rangle = \pi(x;\beta,\theta), \tag{12}$$

where $U(\beta,\theta)$ represents the qSLP circuit. In order to get an estimate of $\pi(\cdot)$, we have to run the entire circuit multiple times.

(Step 5) The post-processing is performed classically and is task-dependent. For classification models we need four steps: (i) adding a learnable bias term b to produce a continuous output, (ii) applying a thresholding operation, (iii) computing the loss function and (iv) updating the parameters. Notice that all these steps are customisable and can be adapted to the particular needs of the

application. In the case of the experiments presented in Section 3 we adopt the following thresholding operation:

$$f(x_i;\beta,\theta,b) = \begin{cases} 1 & \text{if } \pi(x_i;\beta,\theta)+b > 0.5 \\ 0 & \text{else} \end{cases}, \tag{13}$$

where b is the bias term and $f(x_i;\beta,\theta,b)$ gives us the predicted class for observation x. As loss function we choose the *Sum of Squared Errors* (SSE) between the predictions and the true values y:

$$SSE = Loss(\Theta; D) = \sum_{i=1}^{N} \left[y_i - f\left(x_i;\Theta\right)\right]^2, \tag{14}$$

where N is the total number of observations in the sample and $\Theta = \{\beta,\theta,b\}$. Finally, we exploit the Nesterov accelerated gradient method for updating the parameters, although many alternative optimisation strategies can be adopted to update the parameters [24] .

To summarise, the variational algorithm described above allows reproducing a classical Neural Network with one hidden layer on a quantum computer. In particular, it includes a variational circuit adopted for encoding the data, performing the linear combinations of input neurons and applying the same activation function to their results with just one execution. A single iteration during the learning process is then completed using classical resources to measure the output of the network, compute the loss function and update the parameters. The whole process is then repeated iteratively until convergence, as for classical Neural Networks.

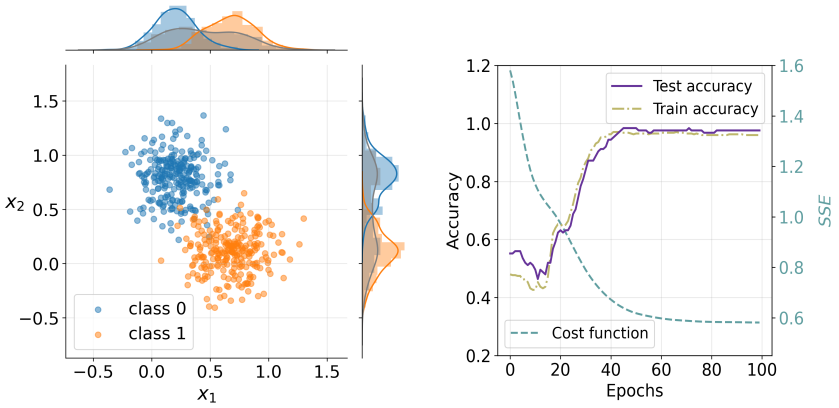
As a final remark, notice that having a post-processing step that is extremely flexible enables the adoption of this model both for regression and classification problems, thus enhancing the impact of such algorithm.

3 Experiments

To test the performances of the qSLP, we implemented the circuit illustrated in Figure 2 using PennyLane [25], a software framework for optimisation and Machine Learning. This library can be used for both quantum and hybrid computations, and allows using quantum objects (e.g. qubits, gates) in conjuction with classical elements (e.g. variables, functions). It can handle many learning tasks such as training a hybrid ML model in a supervised fashion. In addition, we also implemented a version of the qSLP on the Qiskit framework. In this way, we were able to execute the pre-trained algorithm obtained with PennyLane both on QASM simulators and on a real device.

In our case, the goal is to find the parameters of the quantum circuit (β,θ) plus the additional bias term b. In absence of a gate Σ which implements a non-linear activation function, the final quantum state of $|\psi\rangle$ is:

$$|\Phi_5\rangle = \frac{1}{\sqrt{E}}\Big(\beta_1 |0\rangle |L(x;\theta_1)\rangle |\phi\rangle + \beta_2 |1\rangle |L(x;\theta_2)\rangle |\phi\rangle \Big), \tag{15}$$



(a) (b)

Fig. 3: The plot on the left illustrates the distributions of generated data in the two classes $(0, 1)$. The plot on the right shows the trends over training epochs of the cost function and the accuracy.

which is a linear transformation of the input data and defines a linear classifier. Notice that $Pr\left[y_i = 1|x_i\right]$ for a given observation x_i corresponds to the square of the linear transformation of hidden neurons with coefficients β_j plus a bias term, b.

In practice, we generated linearly separable data to test our classifier. In particular, we drew a random sample of 500 observations (250 per class) from two independent bivariate Gaussian distributions, with different mean vectors and the same covariance matrix (Figure 3a). Then, we used the 75% of the data for training and the remaining 25% for testing. The training metrics for the model trained on the PennyLane simulator are illustrated in Figure 3b. The results demonstrate that the quantum SLP is able to classify correctly the observations, as testified by the high classification accuracy in both training and test sets, 0.97 and 0.95 respectively. After the model was trained, the variational algorithm was also implemented using Qiskit, and its performance was tested on 50 newly-generated observations. In this way, it was possible to test the pre-trained model on both the QASM simulator – which emulates the execution

PennyLane	**QASM**	**IBM (Vigo)**
94%	90%	64%

Table 1: Test accuracy of multiple implementations. The performance deteriorates as we introduce intrinsic quantum noise (QASM) and current technology limits (IBM – Vigo).

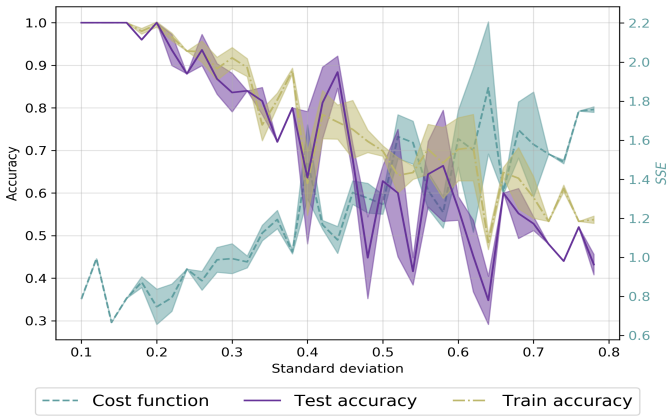


Fig. 4: Assessment metrics trend as a function of distributions overlapping. Larger standard deviations cause the two distributions to overlap, so that observations belonging to the two classes are mixed together and, hence, harder to separate. As a consequence, model performances decrease and non-linearity is required.

of a quantum circuit on a real device, also including highly configurable noise models – and a real device. Results are reported in Table 1. The PennyLane implementation was in line with the training results, and was the most accurate (94% accuracy), as expected since the framework assumes a perfect device. The effects of introducing the intrinsic noise due to quantum computations, instead, can be appreciated in the Qiskit implementations. Both alternatives showed lower performances, although the decrease in accuracy was certainly smaller for QASM. The real device, instead, presented a significant deterioration. This may be due to the depth of the implemented circuit, especially regarding the encoding part, that seems to be prohibitive considering the actual quantum devices.

In addition, we investigated how the performance of the qSLP implemented in PennyLane changes as the generated distributions get closer and less separated. To this end, we drew multiple samples from the two distributions, each time increasing the common standard deviation so to force reciprocal contamination. As expected, the accuracy showed a decreasing trend as the overlap of the distributions increased (Figure 4). In conclusion, the experiments show that the proposed architecture works well for linearly separable data. However, performance decreases as we add to the problem a level of complexity that cannot be solved by linear classifiers.

4 Generalisation to H hidden neurons

In this section we discuss the generalisation of the quantum SLP to the case of $H > 2$ hidden neurons.

In order to extend the quantum state in Equation (11), we can consider a *data* register whose size depends on the number of input features, a *control* register made by d qubits, and another register (*output*) that stores the output of the Neural Network. Intuitively, the algorithm can be summarised into three steps. First, the control register is turned into a non-uniform superposition parameterised by the 2^d-dimensional vector β by means of an oracle B:

$$\begin{aligned} |\Phi_1\rangle &= (\mathbb{1} \otimes B \otimes \mathbb{1}) \, |x\rangle_{\text{data}} \, |0\rangle_{\text{control}} \, |0\rangle_{\text{output}} \\ &\rightarrow \frac{1}{\sqrt{E}} \Big(|x\rangle \otimes \sum_j \beta_j \, |j\rangle \otimes |0\rangle \Big). \end{aligned} \tag{16}$$

The second step generates a superposition of the same linear operation with different parameters entangled with the control register. This is possible by assuming to have a quantum oracle Λ that performs the following operation:

$$|\Phi_2\rangle = \Lambda \, |\Phi_1\rangle \rightarrow \frac{1}{\sqrt{E}} \Big(|x\rangle \sum_j \beta_j \, |j\rangle \, |L(x;\theta_j)\rangle \Big). \tag{17}$$

Finally, the third step applies the Σ gate to the third register, thus propagating the activation function in all of the quantum states of the superposition:

$$|\Phi_3\rangle = (\mathbb{1} \otimes \mathbb{1} \otimes \Sigma) \, |\Phi_2\rangle \rightarrow \frac{1}{\sqrt{E}} \Big(|x\rangle \sum_j \beta_j \, |j\rangle \, |\sigma[L(x;\theta_j)]\rangle \Big). \tag{18}$$

In this way, the result of the algorithm above can be accessed by a single-qubit measurement. Regarding the parameters, β and $\{\theta_j\}_{j=1,\dots,H}$ can be randomly initialised and the same hybrid optimisation process presented in Section 2.4 can be exploited.

As a final remark, it is important to notice that our algorithm entangles linear combinations to the states of the control register. As a consequence, the number of linear combinations that can be performed is equal to the number of possible states of the quantum system. This, in turn, implies that the number of hidden neurons H scales exponentially with the number of states of the control register, 2^d. This is a consequence of each hidden neuron being represented by a single linear combination. Thus, the exponential scaling property enables the construction of quantum Neural Networks with an arbitrary large number of hidden neurons as the amount of available qubits increases. In other terms, we can build qSLP with an incredible descriptive power that may be really capable of being an universal approximator.

5 Conclusions and Outlook

In this work, we proposed an implementation of a quantum version of the Single Layer Perceptron. The key idea is to use a single state preparation routine and apply different linear combinations in superposition, each entangled with a control register. This allows propagating the routine of a generic activation function

to all of the states with only one operation. As a result, a model trained through our algorithm is potentially able to approximate any desired function as long as enough hidden neurons and a non-linear activation function are available.

Furthermore, we provided a practical implementation of our variational algorithm that reproduces a quantum SLP for classification with two hidden neurons and an identity function as activation.

In addition, we tested our algorithm on synthetic data and demonstrated that the model works well in case of linearly separable observations, with a test accuracy of 95%. However, the performance deteriorates when facing the intrinsic noise due to quantum computations and current technology limits. On the other hand, experiments showed how the performance of the model deteriorates as the distributions of the two classes overlap so to contaminate each other, thus testifying the necessity of introducing non-linearity into the model. For this reason, the main challenge to tackle in the near future is the design of a routine that reproduces a non-linear activation function.

Another natural follow-up of this work is the implementation of a generalisation of the quantum SLP to the case of $H > 2$ hidden neurons. This would be beneficial for more hands-on experimentation, including, for instance, the discussion of a regression task.

In conclusion, we are still far from proving that Machine Learning can benefit from Quantum Computing in practice. However, thanks to the flexibility of variational algorithms, we believe that the hybrid quantum-classical approach may be the ideal setting to make universal approximation possible in quantum computers.

References

1. Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.
2. Benjamin P Lanyon, James D Whitfield, Geoff G Gillett, Michael E Goggin, Marcelo P Almeida, Ivan Kassal, Jacob D Biamonte, Masoud Mohseni, Ben J Powell, Marco Barbieri, et al. Towards quantum chemistry on a quantum computer. *Nature chemistry*, 2(2):106, 2010.
3. Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.
4. Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242, 2017.
5. Dave Wecker, Matthew B. Hastings, and Matthias Troyer. Progress towards practical quantum variational algorithms. *Phys. Rev. A*, 92:042303, Oct 2015.
6. Nikolaj Moll, Panagiotis Barkoutsos, Lev S Bishop, Jerry M Chow, Andrew Cross, Daniel J Egger, Stefan Filipp, Andreas Fuhrer, Jay M Gambetta, Marc Ganzhorn, et al. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology*, 3(3):030503, 2018.

7. Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5:4213, 2014.
8. Diego Ristè, Marcus P da Silva, Colm A Ryan, Andrew W Cross, Antonio D Córcoles, John A Smolin, Jay M Gambetta, Jerry M Chow, and Blake R Johnson. Demonstration of quantum advantage in machine learning. *npj Quantum Information*, 3(1):16, 2017.
9. Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.
10. Maria Schuld, Alex Bocharov, Krysta Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *arXiv preprint arXiv:1804.00633*, 2018.
11. Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.
12. Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.
13. Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. The quest for a quantum neural network. *Quantum Information Processing*, 13(11):2567–2586, 2014.
14. Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. Simulating a perceptron on a quantum computer. *Physics Letters A*, 379(7):660–663, 2015.
15. Jean Faber and Gilson A Giraldi. Quantum models of artificial neural networks. *Electronically available: http://arquivosweb. lncc. br/pdfs/QNN-Review. pdf*, 5(7.2):5–7, 2002.
16. Francesco Tacchino, Chiara Macchiavello, Dario Gerace, and Daniele Bajoni. An artificial neuron implemented on an actual quantum processor, zak1998quantum. *npj Quantum Information*, 5(1):26, 2019.
17. Maria Schuld and Francesco Petruccione. *Supervised Learning with Quantum Computers*, volume 17. Springer, 2018.
18. Mikko Mottonen, Juha J Vartiainen, Ville Bergholm, and Martti M Salomaa. Transformation of quantum states using uniformly controlled rotations. *arXiv preprint quant-ph/0407010*, 2004.
19. Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.
20. Yudong Cao, Gian Giacomo Guerreschi, and Alán Aspuru-Guzik. Quantum neuron: an elementary building block for machine learning on quantum computers. *arXiv preprint arXiv:1711.11240*, 2017.
21. Wei Hu. Towards a real quantum neuron. *Natural Science*, 10(3):99–109, 2018.
22. Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52:3457–3467, Nov 1995.
23. Daniel K Park, Ilya Sinayskiy, Mark Fingerhuth, Francesco Petruccione, and June-Koo Kevin Rhee. Quantum forking for fast weighted power summation. *arXiv preprint arXiv:1902.07959*, 2019.
24. Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.
25. Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Carsten Blank, Keri McKiernan, and Nathan Killoran. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.