A Variational Algorithm for Quantum Single Layer Perceptron

Antonio Macaluso[1⋆], Filippo Orazi[2⋆], Matthias Klusch[1], Stefano Lodi[2], and Claudio Sartori[2]

[1] German Research Center for Artificial Intelligence (DFKI), Saarbruecken, Germany
{antonio.macaluso, matthias.klusch}@dfki.de
[2] University of Bologna, Bologna, Italy
filippo.orazi@studio.unibo.it, {stefano.lodi, claudio.sartori}@unibo.it

Abstract. Hybrid quantum-classical computation represents one of the most promising approaches to deliver novel machine learning models capable of overcoming the limitations imposed by the classical computing paradigm. In this work, we propose a novel variational algorithm for quantum Single Layer Perceptron (qSLP) which allows producing a quantum state equivalent to the output of a classical single-layer neural network. In particular, the proposed qSLP generates an exponentially large number of parametrized linear combinations in superposition that can be learnt using quantum-classical optimization. As a consequence, the number of hidden neurons scales exponentially with the number of qubits and, thanks to the universal approximation theorem, our algorithm 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 using near-term quantum computation, especially those related to quantum machine learning. Finally, we test the qSLP as a classification model against two different quantum models on two different real-world datasets usually adopted for benchmarking classical algorithms.

Keywords: Quantum Machine Learning · Quantum Computing · Machine Learning · Neural Networks.

1 Background and Motivation

Machine learning (ML) can be considered as one of the most disruptive technology of the last decades. However, the ever-increasing size of datasets and Moore's law coming to an end emphasizes that the current computational tools will no longer be sufficient in the near future. An opportunity for ML to overcome these limitations is to leverage quantum computation, where quantum effects allow solving selected problems that are intractable using classical machines. The intersection between ML and quantum computing is known as Quantum Machine Learning (QML) where a quantum algorithm is employed to solve typical ML

⋆ Both authors equally contributed to this research.

tasks. In particular, QML algorithms are designed to tackle optimization problems using both classical and quantum resources and they are composed by three main ingredients: *i*) a parametrized quantum circuit $U(x;\Theta)$, *ii*) a quantum output $f(x;\Theta)$ and *iii*) an updating rule for Θ.

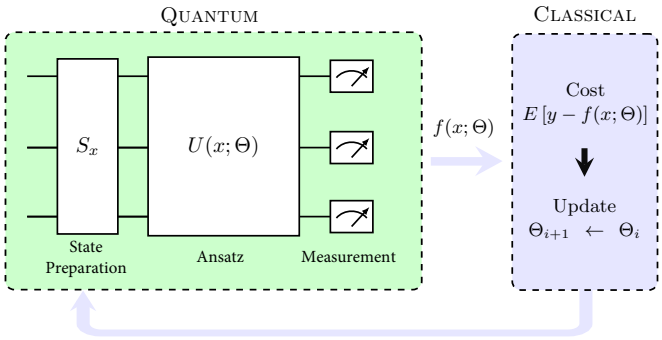

Fig. 1: Scheme of a hybrid quantum-classical algorithm for supervised learning. The quantum variational circuit is depicted in green, while the classical component is represented in blue.

The hybrid quantum-classical approach (Figure 1) proceeds as follows. The data, x, are initially encoded as a quantum state through the use of a quantum routine S_x. Then the quantum hardware computes $U(x;\Theta)$ with randomly initialized 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.

1.1 Neural Network as Universal Approximator

A Single Layer Perceptron (SLP) [1] is a single-layer neural network suitable for classification and regression problems. Given a training point $(\mathbf{x}, y)$, the output of a SLP containing H neurons can be expressed as:

$$f(\mathbf{x};\beta,\theta) = \sigma_{\text{out}}\left(\sum_{j=1}^{H}\beta_j\sigma_{\text{hid}}\left[L(\mathbf{x};\boldsymbol{\theta}_j)\right]\right) = \sum_{j=1}^{H}\beta_j g(\mathbf{x};\theta_j). \tag{1}$$

where σ_{out} is the identify function in case of a continuous target variable y. Each hidden neuron j computes a linear combination, $L(\cdot)$, of the input features $\mathbf{x} \in \mathbb{R}^p$ with coefficients given by the p-dimensional vector $\boldsymbol{\theta}_j$. 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 less utilized than the deep architectures, the SLP model can be very expressive. According to the *universal approximation theorem* [2], a SLP with an arbitrarily large number of hidden neurons and a non-constant, bounded and continuous activation function can approximate any continuous function on a closed and bounded subset of $\mathbb{R}^p$. In spite of this crucial theoretical result, the SLP are rarely adopted in practice due to the unfeasibility of training large amounts of hidden neurons on classical devices. Quantum computers, however, could leverage the superposition of states to scale the number of hidden neurons exponentially with the number of available qubits. Starting from these considerations, cleverly implementing a quantum SLP would therefore enable a real chance to benefit from the universal approximation property.

1.2 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. Recently, the idea of quantum splines (QSplines) [3] has been proposed to approximate non-linear activation functions by means of a fault-tolerant quantum algorithm. Although a QSpline provides a method to store the value of a non-linear function in the amplitudes of a quantum state, it uses the HHL [4] as subroutine, a quantum algorithm for matrix inversion which imposes several practical limitations [5].

In this work, we do not discuss how to implement in practice a non-linear activation function. Nevertheless, we provide a framework that permits to train a quantum SLP with an exponentially large number of hidden neurons for a given activation function Σ. Furthermore, our architecture is naturally capable of incorporating new implementations of non-linear activation functions as long as they fit in the learning paradigm.

2 Related Works

Variational circuits can be considered as the composition of multiple layers of connected computational units controlled by trainable parameters. This definition is similar to the characterization of neural networks, but the comparison between the two poses several challenges. In fact, classical neural networks require a non-linear activation function which is a limit for quantum computation since quantum operations are unitary and therefore linear. Furthermore, it is impossible to access the quantum state at intermediate points during computation. Thus, backpropagation [6], which is the standard to work on large-scale models, cannot be used to train quantum circuits.

In practice, several attempts for building a quantum neural network are discussed in the literature. Although a potential quantum advantage of quantum neural networks over classical ones has been recently investigated [7], to the best of our knowledge, there are no trainable quantum algorithms that can effectively

implement a complete neural network in a quantum setting. A concrete implementation of a quantum circuit to solve a typical ML task using a near-term processor is illustrated in [8], where the authors introduced a model for binary classification using a modified version of the perceptron updating rule. Another approach is represented by Tensor Networks inspired by quantum many-body physics. Tensor networks [9] are methods to represent quantum states whose entanglement is constrained by local interactions. This approach enables the numerical treatment of systems through layers of abstraction, as for deep neural networks. Some of the most studied tensor networks have been adopted for classification and generative modeling [10,11]. Recently, the idea of building a parametrized quantum circuit to generate a quantum state equivalent to the output of a two-layer neural network has been proposed [12]. However, the generalization to more than two neurons is not provided.

In general, the standard approach for quantum neural networks (QNN) consists of building a quantum circuit on top of existing approaches for variational circuits which solve a supervised learning task [13]. Methods within this approach take care of the supervised task end-to-end, where the input quantum state is fed through a parametrized unitary operator and the final measurement provides the estimate of the target variable of interest.

Alternatively, quantum support vector machines (QSVM) [14] use a quantum circuit to map classical data into a quantum feature space. The resulting features are then fed into a classical linear model which calculates the final output.

2.1 Contribution

In this work, we propose the quantum Single Layer Perceptron (qSLP), a novel quantum algorithm that reproduces a quantum state equivalent to the output of a classical SLP with an arbitrarily large number of hidden neurons. In particular, building on top of the approach described in [12], we design an efficient variational algorithm that performs unitary parametrized transformations of the input in superposition. The results are then passed through an activation function with just one application. The flexible architecture of the qSLP enables to plug in custom implementations of the activation function routine. Thus, thanks to the possibility of learning the parameters for a given task, the proposed algorithm allows training models that can potentially approximate any function if provided of a proper activation 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. As a consequence, 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, while increasing linearly the depth of the correspondent quantum circuit and candidating the qSLP to be a concrete alternative for approximating complex and diverse functions.

Finally, we test the qSLP as classification model against two different quantum approaches on two different real-world datasets usually adopted for benchmarking classical ML algorithms.

3 Generalized Quantum Single Layer Perceptron

3.1 Preliminaries

Gates as Linear Operators. As discussed in Sec. 1.1, an SLP applies multiple linear combinations on the same input based on different sets of parameters. In case of qSLP, we can consider the following parametrized quantum gate to map a generic input quantum state any possible other quantum states:

$$U_3(\boldsymbol{\theta}) = U_3(\alpha, \beta, \gamma) = \begin{pmatrix} e^{i\beta}cos(\alpha/2) & e^{i\gamma}sin(\alpha/2) \\ -e^{-i\gamma}sin(\alpha/2) & e^{-i\beta}cos(\alpha/2) \end{pmatrix}. \tag{2}$$

The unitary $U_3(\boldsymbol{\theta})$ represents a bounded linear operation where the set of parameters vector $\boldsymbol{\theta} = \{\alpha, \beta, \gamma\}$ can be learnt using classical optimization procedures. Importantly, the adoption of $U_3(\boldsymbol{\theta})$ as linear operator implies transforming the input data using complex coefficients. Therefore, it describes a more general operation with respect to the classical SLP, that only allows for linear combinations with real-valued coefficient. Nonetheless, one can still parametrize the circuit using Pauli-Y rotation to restrict the computation to the real domain.

Ansatz for Linear Operators in Superposition. The original proposal of the qSLP [12] creates entanglement between the *control* and the *data* register using two CSWAP operations and an additional *temp* register to generate a superposition of two different linear transformations. Although this approach allows obtaining the desired result, it implies the adoption of twice the number of qubits to encode the input data (*temp* register). Furthermore, using the CSWAP gates introduces a linear overhead with respect to the size of the two registers *data* and *temp* [15], in terms of gate complexity. To reduce such complexity, we propose a different approach (illustrated in Figure 2) which does not require the additional *temp* register and the use of the CSWAP gates.

3.2 Quantum Single Layer Perceptron

Intuitively, a generalized qSLP can be implemented into five steps: *state preparation*, *entangled linear operators in superposition*, *application of the activation function*, *measurement step*, *post processing optimization*. To this end, two quantum registers are necessary: *control* (d qubits), *data* (n qubits).

Step 1: State Preparation. The *control* register is turned into a non-uniform superposition parameterized by the d-dimensional vector $\hat{\boldsymbol{\beta}}$ by means of d Pauli-Y rotation gates. Also, a single p-dimensional training point is encoded into the

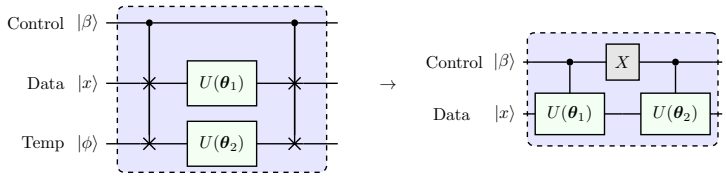

Fig. 2: Alternative ansatz for linear operators in superposition. On the left the quantum gates adopted in the first version of the two-layer qSLP [12]. On the right the proposed schema to generate two different linear operation in superposition each entangled with one of the basis states of the *control* quibit.

n-qubit *data* register through the quantum gate S_x:

$$\begin{aligned} |\Phi\rangle_{\text{SP}} &= \left(\bigotimes_{i=1}^{d} R_y(\hat{\beta}_i) \otimes S_x \right) |0\rangle_{\text{control}}^{\otimes d} \otimes |0\rangle_{\text{data}}^{\otimes n} \\ &= \bigotimes_{i=1}^{d} \left(a_i |0\rangle + b_i |1\rangle \right) \otimes |x\rangle = \bigotimes_{i=1}^{d} |c_i\rangle \otimes |x\rangle \,, \end{aligned} \tag{3}$$

where $\{a_i, b_i\}$ are real numbers such that $|a_i|^2 + |b_i|^2 = 1$. Importantly, the algorithm is completely independent by the encoding strategy chosen for S_x.

Step 2: Entangled Linear Operators in Superposition. The second step generates a superposition of 2^d linear operations with different parameters, each entangled with a basis state of the *control* register. In particular, once the two registers are initialized, each qubit in the *control* register is entangled with two different random transformations of the *data* register. As a result, 2^d different transformations in superposition of the input are generated. Here we describe the procedure assuming 3 *control* qubits where the ansatz described in Sec. 3.1 is applied only $d = 3$ times in order to obtain $2^d = 8$ different parametrized unitary transformations of the input in superposition.

Step 2.1 ($d = 1$) The first step applies the unitary $U(\boldsymbol{\theta}_{1,1}, \boldsymbol{\theta}_{1,2})$ to the initialized quantum system, which is defined as follows:

$$U(\boldsymbol{\theta}_{1,1}, \boldsymbol{\theta}_{1,2}) = \left[\mathbb{1}^{\otimes d-1} \otimes C\text{-}U(\boldsymbol{\theta}_{1,1}) \right] \left[\mathbb{1}^{\otimes d-1} \otimes X \otimes \mathbb{1} \right] \left[\mathbb{1}^{\otimes d-1} \otimes C\text{-}U(\boldsymbol{\theta}_{1,2}) \right]$$

where $\mathbb{1}$ is the identity matrix and X is the NOT gate (i.e., Pauli-X gate). Thus, the first step ($d = 1$) leads to the following quantum state:

$$\begin{aligned} |\Phi\rangle_{(d=1)} &= U(\boldsymbol{\theta}_{1,1}, \boldsymbol{\theta}_{1,2}) \, |\Phi\rangle_{\text{SP}} \\ &= \bigotimes_{i=1}^{2} |c_i\rangle \otimes (a_1 |1\rangle U(\boldsymbol{\theta}_{1,2}) |x\rangle + b_1 |0\rangle U(\boldsymbol{\theta}_{1,1}) |x\rangle). \end{aligned} \tag{4}$$

The basis states of the first *control* qubit is then entangled with two parametrized unitary transformations of the *data* register.

Step 2.2 ($d = 2$) The second step employs another *control* qubit and other two sets of parameters $\boldsymbol{\theta}_{2,1}$ and $\boldsymbol{\theta}_{2,2}$. Then the same procedure is applied through the unitary operations $U(\boldsymbol{\theta}_{2,1}, \boldsymbol{\theta}_{2,2})$:

$$\begin{aligned}|\Phi\rangle_{(d=2)} &= U(\boldsymbol{\theta}_{2,1}, \boldsymbol{\theta}_{2,2}) |\Phi\rangle_{(d=1)} \\ &= \frac{1}{\sqrt{4}}\Big[b_2 a_1 |00\rangle U(\boldsymbol{\theta}_{2,1})U(\boldsymbol{\theta}_{1,1}) |x\rangle + b_2 b_1 |01\rangle U(\boldsymbol{\theta}_{2,1})U(\boldsymbol{\theta}_{1,2}) |x\rangle \\ &\quad + a_2 a_1 |10\rangle U(\boldsymbol{\theta}_{2,2})U(\boldsymbol{\theta}_{1,1}) |x\rangle + a_2 b_1 |11\rangle U(\boldsymbol{\theta}_{2,2})U(\boldsymbol{\theta}_{1,2}) |x\rangle\Big] \qquad (5)\end{aligned}$$

with $U(\boldsymbol{\theta}_{2,1}, \boldsymbol{\theta}_{2,2}) = [\mathbb{1} \otimes C \otimes \mathbb{1} \otimes U(\boldsymbol{\theta}_{2,1})] \left[\mathbb{1} \otimes X \otimes \mathbb{1}^{\otimes n+1}\right] [\mathbb{1} \otimes C \otimes \mathbb{1} \otimes U(\boldsymbol{\theta}_{2,1})]$. The position of the C gate indicates the *control* qubit used to perform the controlled operations C-$U(\boldsymbol{\theta}_{2,1})$ and C-$U(\boldsymbol{\theta}_{2,2})$. As a consequence of this second step, four parametrized transformations of the input $|x\rangle$ are generated, each results from the product of two $U(\boldsymbol{\theta}_{i,k})$ gates, for $i, k \in \{1, 2\}$. Furthermore, these transformations are entangled with the basis states of the two *control* qubits.

Step 2.3 ($d = 3$) We perform the same procedure for the third *control* qubit, using the unitary $U(\boldsymbol{\theta}_{3,1}, \boldsymbol{\theta}_{3,2})$:

$$\begin{aligned}&|\Phi\rangle_{(d=3)} = U(\boldsymbol{\theta}_{3,1}, \boldsymbol{\theta}_{3,2}) |\Phi\rangle_{(d=2)} \\ &= \frac{1}{\sqrt{8}}\Big[\beta_1 |000\rangle U(\boldsymbol{\theta}_{3,1})U(\boldsymbol{\theta}_{2,1})U(\boldsymbol{\theta}_{1,1}) |x\rangle + \beta_2 |001\rangle U(\boldsymbol{\theta}_{3,1})U(\boldsymbol{\theta}_{2,1})U(\boldsymbol{\theta}_{1,2}) |x\rangle \\ &\quad + \beta_3 |010\rangle U(\boldsymbol{\theta}_{3,1})U(\boldsymbol{\theta}_{2,2})U(\boldsymbol{\theta}_{1,1}) |x\rangle + \beta_4 |011\rangle U(\boldsymbol{\theta}_{3,1})U(\boldsymbol{\theta}_{2,2})U(\boldsymbol{\theta}_{1,2}) |x\rangle \\ &\quad + \beta_5 |100\rangle U(\boldsymbol{\theta}_{3,2})U(\boldsymbol{\theta}_{2,1})U(\boldsymbol{\theta}_{1,1}) |x\rangle + \beta_6 |101\rangle U(\boldsymbol{\theta}_{3,2})U(\boldsymbol{\theta}_{2,1})U(\boldsymbol{\theta}_{1,2}) |x\rangle \\ &\quad + \beta_7 |110\rangle U(\boldsymbol{\theta}_{3,2})U(\boldsymbol{\theta}_{2,2})U(\boldsymbol{\theta}_{1,1}) |x\rangle + \beta_8 |111\rangle U(\boldsymbol{\theta}_{3,2})U(\boldsymbol{\theta}_{2,2})U(\boldsymbol{\theta}_{1,2}) |x\rangle\Big] \\ &= \frac{1}{\sqrt{8}} \sum_{j=1}^{8} \beta_j |j\rangle U(\boldsymbol{\Theta}_j) |x\rangle , \qquad (6)\end{aligned}$$

where $U(\boldsymbol{\theta}_{3,1}, \boldsymbol{\theta}_{3,2}) = \left[C \otimes \mathbb{1}^{\otimes 2} \otimes U(\boldsymbol{\theta}_{2,1})\right] [X \otimes \mathbb{1} \otimes \mathbb{1}] \left[C \otimes \mathbb{1}^{\otimes 2} \otimes U(\boldsymbol{\theta}_{2,1})\right]$.

The final quantum state is a superposition of 8 different parametrized unitary transformations of the input, each entangled with a basis state of the *control* register whose amplitudes depend on a set of parameters $\{\beta_i\}_{i=1,\ldots,d}$. We can generalize the quantum state $|\Phi\rangle_{(d=3)}$ for a generic d-qubit *control* register:

$$\begin{aligned}|\Phi\rangle_d &= \prod_{i=1}^{d} U(\boldsymbol{\theta}_{i,1}, \boldsymbol{\theta}_{i,2}) \left(\bigotimes_{i=1}^{d} |c_i\rangle \otimes |x\rangle\right) \\ &= \frac{1}{\sqrt{2^d}} \sum_{j=1}^{2^d} \beta_j |j\rangle U(\boldsymbol{\Theta}_j) |x\rangle = \frac{1}{\sqrt{2^d}} \sum_{j=1}^{2^d} \beta_j |j\rangle |L(x; \boldsymbol{\Theta}_j)\rangle , \qquad (7)\end{aligned}$$

where $U(\boldsymbol{\Theta}_j)$ is the product of d unitaries $U(\boldsymbol{\theta}_{i,k})$ for $i = 1, \ldots, d$ and $k = 1, 2$.

To summarize, the underlying idea of this procedure is to initialize the *control* register according to a set of weights and assign each weight β_j to a parametrized unitary function $U(\boldsymbol{\Theta}_j)$. This approach is extremely flexible and allows learning all the parameters β_j and $\boldsymbol{\Theta}_j$ for specific use cases. Furthermore, it allows to generate a superposition of 2^d diverse unitary transformation while increasing linearly the depth correspondent quantum circuit.

Step 3: Activation. A further step consists of applying the Σ gate, representing the quantum version of the classical activation function to the *data* register. Notice that, having all the parametrized unitary operations in superposition allows propagating the application of Σ with a single execution, as follows:

$$\begin{aligned} |\Phi\rangle_\Sigma = (\mathbb{1}^{\otimes d} \otimes \Sigma)\,|\Phi_d\rangle &\rightarrow \frac{1}{\sqrt{2^d}} \sum_{j=1}^{2^d} \beta_j\,|j\rangle\,|\sigma[L(x;\theta_j)]\rangle) \\ &= \frac{1}{\sqrt{H}} \sum_{j=1}^{H} \beta_j\,|j\rangle\,|g(x;\boldsymbol{\Theta}_j)\rangle\,, \end{aligned} \tag{8}$$

where $H = 2^d$ and $\mathbb{1}^{\otimes d}$ is the identity matrix. In this way, the result of the algorithm corresponds to the output of the SLP (Eq. (1)) with 2^d hidden neurons that can be accessed by measuring the *data* register only.

Step 4: Measurement. Finally, the expectation measurement on the *data* register is performed:

$$\begin{aligned} \langle M\rangle &= \left\langle \Phi_\Sigma | \mathbb{1}^{\otimes d} \otimes M \middle| \Phi_\Sigma \right\rangle \\ &= \sum_{j=1}^{2^d} \beta_j^{'}\,\langle g(x;\boldsymbol{\Theta}_j)|M|g(x;\boldsymbol{\Theta}_j)\rangle \\ &= \sum_{j=1}^{2^d} \beta_j^{'}\,\langle M_j\rangle = \sum_{j=1}^{2^d} \beta_j^{'} g_j = f(\mathbf{x};\boldsymbol{\beta},\boldsymbol{\Theta}), \end{aligned} \tag{9}$$

where M is a generic measurement operator (e.g., the Pauli σ_z), the function $g(x;\boldsymbol{\Theta}_j) = \sigma[L(x;\theta_j)]$ and $\beta_j^{'} = |\beta_j|^2$ with $\sum_j |\beta_j|^2 = 1$. Although we do not measure the *control* register, the j-th transformation of the input is associated to a specific amplitude β_j of the *control* register. The parameters of the quantum circuit $\{\hat{\beta}_i\}_{i=1,\ldots,d}$ and $\{\boldsymbol{\theta}_{i,1}, \boldsymbol{\theta}_{i,2}\}_{i=1,\ldots,d}$, that indirectly determine the parameters in Eq. (9), can be randomly initialized and hybrid quantum-classical optimization process can be exploited (Section 1).

Thus, we extended the proposed approach of the qSLP [12] to an exponentially large number of neurons in the hidden layer. In fact, the entanglement of linear combinations to the basis states of the *control* register implies that the number of different linear combinations is equal to the number of basis states of the *control* register. This translates in a number of hidden neurons H which

scales exponentially with the number of qubits of the *control* register, d. This is a consequence of each hidden neuron being represented by a single independent trajectory (Eq. (8)). The exponential scaling alongside the ability to freely learn the parameters, enables the construction of quantum neural network 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. The quantum circuit to implement the qSLP ($d = 3$) is depicted in Figure 3.

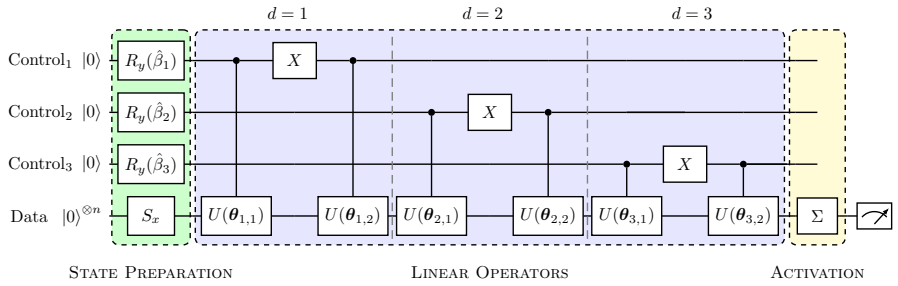

Fig. 3: Quantum circuit for a qSLP with 8 ($d = 3$) hidden neurons.

Step 5: Optimization. The post-processing is task-dependent and performed classically. In particular, given the measurement of the quantum circuit $f(\mathbf{x}; \boldsymbol{\beta}, \boldsymbol{\Theta})$, a cost function is computed and then classical approaches to update the parameters of the quantum circuit $\{\boldsymbol{\beta}, \boldsymbol{\Theta}\}$ are employed. As loss function, in case of supervised problem, the *Sum of Squared Errors* (SSE) between the predictions and the true values y is usually adopted:

$$SSE = Loss(\beta, \boldsymbol{\Theta}; D) = \sum_{i=1}^{N} \left[y_i - f\left(\mathbf{x}_i; \boldsymbol{\beta}, \boldsymbol{\Theta}\right) \right]^2, \tag{10}$$

where N is the total number of observations in the training set.

To summarize, 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 to measure the output of the network, compute the loss function and update the parameters. The whole process is repeated iteratively until convergence, as for classical neural networks.

3.3 Discussion

The algorithm provided in the previous section allows building a generalized qSLP with an exponentially large number of neurons, increasing the depth of

the quantum circuit linearly. Furthermore, the model supports amplitude encoding strategy, which translates into a exponential advantage in terms of space complexity (i.e., number of qubits) when encoding data into the *data* quantum register. This implies a potential polylogarithmic advantage in terms of the number of parameters with respect to its classical counterpart [16].

There are two main differences between the classical and quantum SLP, deriving from the normalization constraint introduced when dealing with the amplitudes of quantum systems. In the qSLP, the input data $\mathbf{x}$ and the weight vector $\boldsymbol{\beta} = \{\beta_j\}_{j=1,\dots,2^d}$ are normalized to 1. This may seem a limitation since classical neural networks may take raw input and freely learn the weight parameters. However, rescaling the inputs and limiting the magnitudes of the weights are two common strategies adopted in the classical case to avoid overfitting. In particular, these procedures are known as *batch normalization* and *weight decay* [17]. Thus, quantum mechanics' normalization constraint allows automatically implementing of ad-hoc procedures developed in the context of classical neural networks without any additional computational effort.

From a computational point of view, given H hidden neurons and L training epochs, the training of a classical SLP scales (at least) linearly in H and L, since the output of each hidden neuron needs to be calculated explicitly to obtain the final output. Also, if H is too large (a necessary condition for an SLP to be a universal approximator [2]), the problem becomes NP-hard [18]. The proposed qSLP, instead, allows scaling linearly with respect to $log_2(H) = d$ thanks to the entanglement between the two quantum registers that generates an exponentially large number of quantum trajectories in superposition. However, the cost of state preparation (gate S_x) needs to be taken into consideration, as well as the cost implementing the quantum activation function Σ. Nonetheless, once the optimal set of parameters of the qSLP is obtained for a specific task, the whole quantum algorithm can be employed as subroutine in other quantum algorithms to reproduce the function for which it is trained.

4 Experiments

To test the performances of the qSLP, we implemented the circuit illustrated in Figure 3 using IBM Qiskit environment on two different real-world datasets. After training the qSLP on the simulator, we executed the pre-trained algorithm on two real quantum devices. In addition, QNNs [7] and QSVMs [19] are adopted as benchmark to compare the proposed qSLP with state-of-the-art QML models[3].

4.1 Dataset Description

The simulation of a quantum system on a classical device is a challenging task even for systems of moderate size. For this reason, the experiments consider only datasets with a relatively small number of observations (100–200) that will be

[3] All code to generate the data, figures and analyses is available at github.com/filorazi/qSLP-quantum-Single-Layer-Perceptron

split in training (80%) and test (20%) set. Furthermore, the PCA is performed to reduce the number of features to 2. We consider five different binary classification problems from two real-world datasets (MNIST and Iris) usually adopted for benchmarking classical machine learning algorithms. The Iris dataset consists of 50 examples for three species of Iris flower (*Setosa, Virginica, Versicolor*) described by four different features. The MNIST dataset contains $60k$ images of ten possible handwritten digits, each represented by 28×28 pixels. Since these two datasets exhibit a multiclass target variable, but the current implementations of qSLP, QNN, and QSVM solve a binary classification problem, we consider five different datasets extracted from the original multiclasses problems (Table 1).

	MNIST09	**MNIST38**	**SeVe**	**SeVi**	**VeVi**
Training set	160	160	80	80	80
Test set	40	40	20	20	20
Raw features	784	784	4	4	4
PCA features	2	2	2	2	2
% Variance	31%	20.3%	98%	98.4%	92.2%
Target classes	0 vs 9	3 vs 8	Setosa vs Versicolor	Setosa vs Virginica	Versicolor vs Virginica

Table 1: **Datasets description**. The columns represent five different binary classification datasets. Each row describes a characteristic of the data: the number of training points (Training set), the number of test points (Test set), the number of features in the original data (Raw features), the number of PCA features, the explained variance of the PCA (% Variance) and the target classes chosen as the binary target variable (Target classes).

4.2 State Preparation for the qSLP

The most efficient encoding strategy adopted in QML is amplitude encoding that associates quantum amplitudes with real vectors at the cost of introducing a normalization constraint to the raw input. Formally, a normalized vector $\mathbf{x} \in \mathbb{R}^p$ can be described by the amplitudes of a quantum state $|x\rangle$ as:

$$|x\rangle = \sum_{k=1}^{p} x_k |k\rangle \longleftrightarrow \mathbf{x} = [x_1, \ldots, x_p]^T . \tag{11}$$

Two different approaches are used for amplitude encoding in the qSLP: *Padded state preparation* and *Single-qubit state preparation*.

Single-qubit state preparation. This approach encodes a two-dimensional real vector $\mathbf{x}$ into the amplitudes of a qubit by performing two parametric rotations $R_y(x_1), R_z(x_2)$, where the angles $\{x_1, x_2\}$ are the two components of the (normalized) feature vector $\mathbf{x} = (x_1, x_2)$ [20]. In this case, the parametrized quantum gate $U(\boldsymbol{\theta}_{i,k})$ of the qSLP is directly implemented using the gate U_3 (Eq. (2)).

Padded state preparation. A second strategy for amplitude encoding consists of mapping an input state $|x\rangle$ to the all-zero state $|0\dots0\rangle$. Once the circuit is obtained, all of the operations are inverted and applied in the reversed order. In this case, a two-dimensional input data $\mathbf{x}$ is first mapped into a four-dimensional vector adding constant values and then normalized. Thus, following the procedure described in [21], a set of angles are defined in such a way to apply a sequence of R_y and CNOT gates to generate a quantum state whose amplitudes are equivalent to the input (four-dimensional) feature vector. The quantum circuit for *padded state preparation* is depicted in Figure 4. When using this strategy, since the *data* register is made up of 2 qubits, the unitary operator $U(\boldsymbol{\theta}_{i,k})$ of the qSLP is implemented by two U_3 gates (one per qubit) and a CNOT gate.

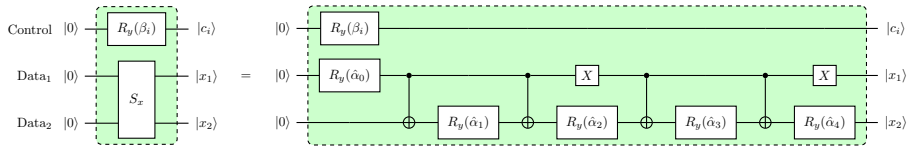

Fig. 4: Quantum circuit for *padded state preparation* in the qSLP. The $\{\hat{\alpha}_i\}_{i=0,\dots,4}$ parameters are determined following the procedure described in [21].

4.3 Results

We consider nine different algorithms based on the three QML models. A QSVM [19] is trained using a two-layer quantum feature map (*ZZFeature-map*). Two types of QNNs are tested: the first one (QNNC v1) uses single-layer quantum feature map (*ZFeature-map*) as state preparation routine and *Real amplitudes* approach as classification ansatz; the second type (QNNC v2) differs from the first for the use of *ZZFeature-map* as state preparation. Regarding the qSLP, six different configurations are tested based on two possible implementations for amplitude encoding (Section 4.2) and three values for the parameter d. To the best of our knowledge, there is no quantum routine suitable for near-term quantum computation capable of approximating the behaviour of an activation function. For this reason, as Σ gate (Eq. (8)), we adopt the identity gate[4].

The training is performed on a QASM simulator, a backend that simulates the execution of a quantum algorithm in a fault-tolerant quantum device. The number of measurements for each run of the quantum circuits is fixed to 1024. The results for each model are reported in Table 2.

Assuming a perfect quantum device, the QSVM and the QNN outperform the qSLP considering the VeVi dataset. This means that for specific cases the use of a quantum feature map provides a practical advantage with respect to standard amplitude encoding. However, in case of MNIST09, MNIST38 and SeVe the qSLP outperforms the QNNs, thanks to its ability to aggregate multiple and diverse unitary functions. Thus, the flexible architecture of the ansatz of the qSLP

[4] Importantly, the code already allows the embedding of a gate Σ different from the identity gate, as quantum activation function.

		MNIST09		MNIST38		SeVe		SeVi		VeVi	
Model	**d**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
QSVM		.97	**.91**	.83	**.84**	1.0	**1.0**	1.0	**1.0**	.98	**.95**
QNN (v1)		.87	.86	.55	.61	.95	.90	.98	.95	.92	.90
QNN (v2)		.84	.84	.74	.70	1.0	.95	1.0	**1.0**	.85	.80
(padded) qSLP	1	.92	.89	.89	.81	1.0	**1.0**	1.0	**1.0**	.82	.70
	2	.93	**.91**	.87	.82	1.0	**1.0**	1.0	**1.0**	.82	.70
	3	.92	**.91**	.86	**.84**	1.0	**1.0**	1.0	**1.0**	.82	.70
(Single-qubit) qSLP	1	.83	.85	.86	**.84**	1.0	**1.0**	1.0	**1.0**	.80	.70
	2	.83	.85	.87	**.84**	1.0	**1.0**	1.0	**1.0**	.78	.65
	3	.83	.85	.87	.82	1.0	**1.0**	1.0	**1.0**	.80	.65

Table 2: Training and test results of three different models (qSLP, QNN, QSVM) on five different binary classification problems.

allows achieving a better performance when comparing full quantum models[5]. Nonetheless, the qSLP and QSVM equally perform on four cases (MNIST09, MNIST38, SeVe, SeVi), although the implementation of the qSLP is not complete due to the lack of a proper activation function and a limited number of hidden neurons ($H = 2^3 = 8$).

Model	**d**	**MNIST09**	**MNIST38**	**SeVe**	**SeVi**	**ViVe**
QSVM	-	**.91**	**.84**	**1.0**	**1.0**	**.95**
QNN (v1)	-	.89	.64	.95	**1.0**	**.95**
QNN (v2)	-	.89	.68	**1.0**	**1.0**	.75
(padded) qSLP	1	.89	.81	**1.0**	.90	.70
	2	.81	.65	**1.0**	.60	.50
	3	.81	.65	**1.0**	.60	.50
(Single-qubit) qSLP	1	.82	.78	**1.0**	**1.0**	.50
	2	.82	.70	**1.0**	**1.0**	.55
	3	.84	.78	**1.0**	.95	.45

Table 3: Test performances using real devices (*ibmq_lima* and *ibmq_quito*).

Furthermore, the trained algorithms have been executed on real IBM quantum devices. Results are reported in Table 3. The deterioration in the performance of the *(Single-qubit) qSLP* is negligible. While the results of the *(Padded) qSLP*, which uses a higher number of qubits and deeper quantum circuits, deteriorate as d increases. Instead, the QSVM seems not to be affected by the use of a real device since the quantum component only generates the new features, while the classification is performed classically. However, being each feature represented by a single qubit (as for QNN), the use of the quantum feature map on real-world datasets seems to be prohibitive even when considering quantum devices with hundreds of (noisy) qubits. Instead, the amplitude encoding strategy adopted in the qSLP represents an optimal strategy to efficiently encode a large dimensional input data in a small number of qubits. Thus, the *(Single-*

[5] The QSVM uses a quantum circuit to translate the classical data into quantum states while the classification is performed classically.

qubit) qSLP is the best compromise when using a real device, since it seems to preserve its performance while adopting the amplitude encoding strategy.

Importantly, these results are an indication of how the tested QML models perform and do not represent an exhaustive evaluation of their performance.

5 Conclusion and Outlook

In this work, we proposed a quantum algorithm to generate the quantum version of the Single Layer Perceptron (qSLP). 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 a generic activation function to all the basis states with only one execution. 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 different possible hidden neurons and an identity function as activation.

In addition, we performed a comparative analysis between our algorithm and two quantum baselines on real-world data and demonstrated that the qSLP outperforms other full quantum approaches (QNN) and matches with the QSVM.

The main challenges to tackle in the near future is the design of a routine that reproduces a non-linear activation function, as well as the adoption of the qSLP for regression tasks. Yet, recently it has been shown that quantum feature maps alongside functions aggregation are able to achieve universal approximation [22]. Thus, another promising future work is the study of the qSLP on top of the quantum feature map, to obtain a universal approximator, without implementing a non-linear quantum activation function.

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

Acknowledgments

This work has been partially funded by the German Ministry for Education and Research (BMB+F) in the project QAI2-QAICO under grant 13N15586.

References

1. Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.
2. Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

3. Antonio Macaluso, Luca Clissa, Stefano Lodi, and Claudio Sartori. Quantum splines for non-linear approximations. In *Proceedings of the 17th ACM International Conference on Computing Frontiers*, pages 249–252, 2020.
4. Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.
5. Scott Aaronson. Read the fine print. *Nature Physics*, 11(4):291, 2015.
6. David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.
7. Amira Abbas, David Sutter, Christa Zoufal, Aurélien Lucchi, Alessio Figalli, and Stefan Woerner. The power of quantum neural networks. *Nature Computational Science*, 1(6):403–409, 2021.
8. 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.
9. Edward Grant, Marcello Benedetti, Shuxiang Cao, Andrew Hallam, Joshua Lockhart, Vid Stojevic, Andrew G Green, and Simone Severini. Hierarchical quantum classifiers. *npj Quantum Information*, 4(1):1–8, 2018.
10. William Huggins, Piyush Patil, Bradley Mitchell, K Birgitta Whaley, and E Miles Stoudenmire. Towards quantum machine learning with tensor networks. *Quantum Science and technology*, 4(2):024001, 2019.
11. Ding Liu, Shi-Ju Ran, Peter Wittek, Cheng Peng, Raul Blázquez García, Gang Su, and Maciej Lewenstein. Machine learning by unitary tensor network of hierarchical tree structure. *New Journal of Physics*, 21(7):073059, 2019.
12. Antonio Macaluso, Luca Clissa, Stefano Lodi, and Claudio Sartori. A variational algorithm for quantum neural networks. In *International Conference on Computational Science*, pages 591–604. Springer, 2020.
13. Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, 2019.
14. Vojtech Havlicek, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. Supervised learning with quantum enhanced feature spaces. *Nature*, 2018.
15. John A Smolin and David P DiVincenzo. Five two-bit quantum gates are sufficient to implement the quantum fredkin gate. *Physical Review A*, 53(4):2855, 1996.
16. Maria Schuld, Alex Bocharov, Krysta Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *arXiv preprint arXiv:1804.00633*, 2018.
17. Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.
18. J Stephen Judd. *Neural network design and the complexity of learning*. MIT press, 1990.
19. Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.
20. Vivek V. Shende, Stephen S. Bullock, and Igor L. Markov. Synthesis of quantum logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2006.
21. 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.
22. Takahiro Goto, Quoc Hoan Tran, and Kohei Nakajima. Universal approximation property of quantum feature map. *arXiv preprint arXiv:2009.00298*, 2020.