Image Classification via Quantum Machine Learning

Héctor Iván García-Hernández, Raymundo Torres-Ruiz, Guo-Hua Sun

Instituto Politécnico Nacional,
Centro de Investigación en Computación,
Mexico

gsun@cic.ipn.mx

Abstract. Quantum Computing and especially Quantum Machine Learning, in a short period of time, has gained a lot of interest through research groups around the world. This can be seen in the increasing number of proposed models for pattern classification applying quantum principles to a certain degree. Despise the increasing volume of models, there is a void in testing these models on real datasets and not only on synthetic ones. The objective of this work is to classify patterns with binary attributes using a quantum classifier. Specially, we show results of a complete quantum classifier applied to image datasets. The experiments show favorable output while dealing with balanced classification problems as well as with imbalanced classes where the minority class is the most relevant. This is promising in medical areas, where usually the important class is also the minority class.

Keywords: Quantum machine learning, image classification, quantum computing, computational intelligence, imbalanced classification.

1 Introduction

Image classification is of utmost importance in several areas of science and technology such as medical diagnosis and prognosis, face detection, or multiple object detection for autonomous cars. By classical models, this task can be solved using Convolutional Neural Networks [1] but it is notorious the enormous number of parameters needed to train, as seen, for example, in [2]. Nevertheless, exploiting the prowess of Quantum Mechanics such as interference, superposition, and entanglement, which promises great power of computation and in compass with the recent implementation of several quantum computers, it is worth to propose and evaluate quantum models for machine learning. Although these models are, in essence, simple and with performances lower than the state of the art, they serve as stepping stones for the construction of increasingly complex models with much better performance.

The **advantage** of the quantum models is the inherent parallelism in their execution, the speed at which they are executed, and even more important is the exponential reduction of the number of qubits necessary to encode the information compared to the classical models, for example, only 6 qubits are needed to encode a 64-dimensional pattern and with just 30 qubits we could encode a 32768 x 32768 binary image, which is more than a billion-dimensional flatten vector. This reduction is possible by exploiting superposition states and quantum entanglement.

Let us present some high-level descriptions of models proposed by various research groups, either purely quantum or hybrid combining classical and quantum processing. Yamamoto et al. [3] proposed a quantum perceptron model that allows classifying non-linearly separable data. Maria Schuld et al. [4] proposed another quantum perceptron model using unitary operators acting on two qubits and the inverse quantum Fourier transform. Maxwell Henderson et al. [5] put forward a quantum convolutional layer model for the extraction of features in images. Sebastien Piat et al. [6] proposed a preprocessing with auto-encoders, a restricted Boltzmann machine (RBM) is trained in a quantum computer. This RBM is used to initialize a classical neural network which is subsequently trained in a classical way.

Iris Cong et al. [7] utilized another model for a quantum convolutional network. Zhao et al. [8] proposed a swap-based red neural quantum test. Dang et al. [9] proposed a KNN-based quantum classifier, with a classical model for feature extraction. Francesco et al. [10] proposed a new model for a quantum neuron implemented in a real quantum processor. Using Qiskit [11] and Pytorch, arbitrarily large hybrid models can be generated [12]. Despite having various proposals and with various applications [13], each of the models omits a feasible implementation in a real quantum processor, they lack a proof in a real dataset or in their most extreme case they do not correctly use quantum mechanics [14].

With this work, we aim to show the potential application of Quantum Machine Learning to real-world problems in image classification validating and testing a quantum classification model beyond the theoretical realm and the standard proof of concept.

In this work we first explore two real image datasets, some performance measures are discussed, and the implementation of a quantum classifier is described both at a theoretical and at a high level in Section 2. The performance of the classifier in the two datasets is presented in Section 3, evaluating its performance when facing a problem of both, balanced classes and highly unbalanced classes. Finally, the conclusions and future work are presented in Section 4.

2 Datasets and Algorithms

In this section, we summarize two digits images datasets. The performance measures used to evaluate the classifier are discussed and the classifier itself is analyzed. Before starting to introduce them, it is necessary to propose the theoretical description to be used in this work.

2.1 Theoretical Model

Prior to describing the practical steps in the algorithm, a theoretical approach must be taken, so with a little more in-depth description, and following [10] almost verbatim, we start with the binary pattern $\vec{x}^T = (x_0 \dots x_{m-1})$ and the weight vector $\vec{\Omega}^T = (\Omega_0 \dots \Omega_{m-1})$ with $x_j, \Omega_j \in \{0, 1\}$ and then we map them to:

$$\vec{i} = \begin{pmatrix} i_0 \\ i_1 \\ \vdots \\ i_{m-1} \end{pmatrix}, \vec{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{m-1} \end{pmatrix}, \tag{1}$$

with $i_j, w_j \in \{-1, 1\}$ and with them we can define two quantum states:

$$|\psi_i\rangle = \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} i_j |j\rangle \text{ and } |\psi_w\rangle = \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} w_j |j\rangle \ . \tag{2}$$

The states $|j\rangle \in \{|00\dots00\rangle, |00\dots01\rangle, \dots, |11\dots11\rangle\}$ form the computational basis of a quantum processor. If N qubits are used in the register, there are $m = 2^N$ basis states labeled $|j\rangle$ and we can use factors ± 1 to encode the m -dimensional classical patterns and weights into a uniformly weighted superposition of the computational basis.

The first step is to prepare the state $|\psi_i\rangle$ by encoding the input values of $\vec{i}$. With the qubits initialized in the zero state $|00\dots00\rangle \equiv |0\rangle^{\otimes N}$, we perform a unitary transformation U_i:

$$U_i |0\rangle^{\otimes N} = |\psi_i\rangle \ . \tag{3}$$

The second step computes the inner product between $\vec{w}$ and $\vec{i}$ using the quantum register. This can be done defining a unitary transformation, U_w, such that:

$$U_w |\psi_w\rangle = |1\rangle^{\otimes N} = |m-1\rangle \ . \tag{4}$$

If we apply U_w after U_i, the quantum state becomes:

$$U_w |\psi_i\rangle = \sum_{j=0}^{m-1} c_j |j\rangle \equiv |\phi_{i,w}\rangle \ . \tag{5}$$

Using Eq. (4), the scalar product between the two quantum states is:

$$\begin{aligned} \langle \psi_w \mid \psi_i \rangle &= \left\langle \psi_w \left| U_w^\dagger U_w \right| \psi_i \right\rangle \\ &= \langle m-1 \mid \phi_{i,w} \rangle = c_{m-1} \ , \end{aligned} \tag{6}$$

and from the definitions in Eq. (2) we see that the scalar product of input and weight vectors is $\vec{w} \cdot \vec{i} = m \langle \psi_w \mid \psi_i \rangle$. Hence, the desired result is contained, up to a normalization factor, in the coefficient c_{m-1} of the final state $|\phi_{i,w}\rangle$.

In order to extract such an information, an ancilla qubit (a) initially set in the state $|0\rangle$ is toggled by a multi-controlled NOT gate between the N encoding qubits, this leads to:

$$|\phi_{i,w}\rangle\, |0\rangle_a \rightarrow \sum_{j=0}^{m-2} c_j |j\rangle |0\rangle_a + c_{m-1} |m-1\rangle |1\rangle_a \ . \tag{7}$$

The nonlinearity required by the threshold function at the output of the perceptron is immediately obtained by performing a quantum measurement. By measuring the state of the ancilla qubit produces the output $|1\rangle_a$ with probability $|c_{m-1}|^2$.

Even though general advantages or disadvantages cannot be outlined, we can mention some of the differences between this quantum model and a classical one: an execution of a classical model gives an activation directly comparable to a threshold whereas the quantum model gives a binary output which must be repeated several times in order to acquire a measure approximately to theory and comparable to the threshold. Another difference is that since the quantum measure is binary, it can be readily interpreted as the assigned class given the nonlinearity function is already satisfied. And lastly, in general, a classical model does not require input patterns with numerical attributes to be codified into other representations, although it can be useful, however, most quantum models, including this one, are limited to patterns with binary attributes or to be coded as such.

2.2 Datasets

Two databases are used for the experiments. They are both images of digits. The first is the “Optical Recognition of Handwritten Digits Data Set (Digits Dataset)”, which contains 64 attributes in a range from 0 to 16 with 5620 total instances [15]. The test set with 1797 instances, is available directly on the scikit-learn Python package. The second database is the ”Semeion Handwritten Digit Data Set (Semeion Dataset)” which contains 1593 instances each with 256 binary attributes [17]. Both datasets are balanced, containing approximately the same number of instances per class.

Table 1. Summary of both used digits images datasets.

Dataset	Classes	Imbalance Ratio	Total Instances
Digits	10	1.032	5620
Semeion	10	1.045	1593

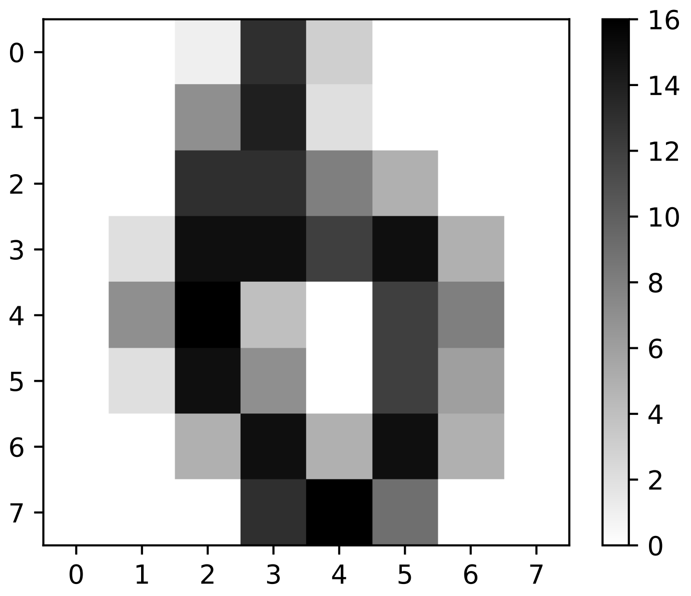

Fig. 1. An instance for class 6 in the Digits Dataset. Each pixel has a value varying from 0 up to 16. As can be seen, the low resolution can lead to misclassification, since this instance can be easily mistaken for an instance of class 0.

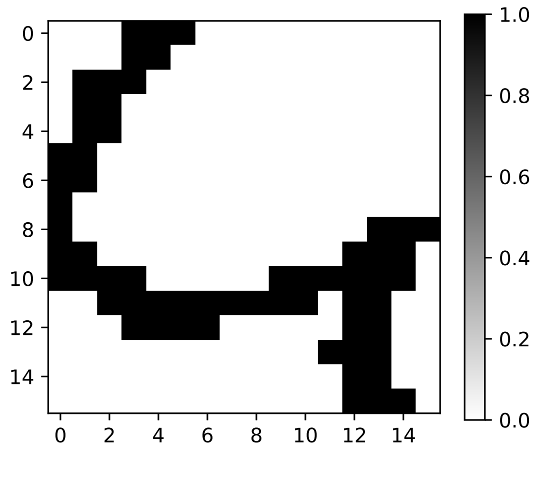

Fig. 2. An instance for class 4 in the Semeion Dataset. Each attribute, or pixel, has a binary value. These patterns have a bigger dimensionality, and thus are less likely to be misclassified, but also the complexity of the processing needed to build the quantum circuit is increased.

2.3 Performance Measures

When each class contains roughly the same number of instances in a dataset, it is known as a balanced dataset. In these cases, most of the performance measures are adequate, as long as there is no bias towards any class. However, depending on the application of the classifier and the relevance of any of the classes, some other performance measure may be chosen.

Table 2. Confusion matrix for a two-class dataset.

		Real Class	
		Positive	Negative
Predicted Class	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

The most common is accuracy, which measures the ratio of instances correctly classified to the total number of instances:

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \; . \tag{8}$$

A dataset is unbalanced when one or more classes is poorly represented in the dataset. Most classical performance measures produce a majority class bias in an unbalanced class problem. In these cases the True Positive Rate (TPR):

$$TPR = \frac{TP}{TP + FN} \; , \tag{9}$$

which is also known as Recall or Sensitivity can be used to measure the ratio of the number of positive instances correctly classified to the total number of positive instances.

We also keep track of the Positive Predictive Value (PPV) (also known as Precision):

$$PPV = \frac{TP}{TP + FP} \; , \tag{10}$$

which measures the ratio of the number of positive instances correctly classified to the total number of positive classified instances. With TPR and PPV the $F1$ score can be obtained, which is the harmonic mean of these performance measures:

$$F1 = 2 * \frac{PPV * TPR}{PPV + TPR} = \frac{2 * TP}{2 * TP + FP + FN} \; . \tag{11}$$

When classes contain insufficient instances to be partitioned in a traditional validation method, it is common to use all instances for training and testing process. Accuracy under this validation method is known as Resubstitution Error.

2.4 Quantum Classification Algorithm

The model used for the classification task is an implementation of the one described in [10]. In this model, an instance with binary attributes is encoded by means of a method called *hypergraph states generation subroutine* [10, 16]. The weight vector is randomly initialized, which will be updated accordingly to a set of hyper-parameters, which will regulate its rate of change. At the end of the execution of the dynamically generated quantum circuit through the process described in [10], a measurement is performed on the ancilla qubit, which will take the value 0 or 1. By means of several repetitions of the circuit, the proportion of measurements with result 1 over the total measurements can be obtained. The more measures are made, the closer the result is to the real one.

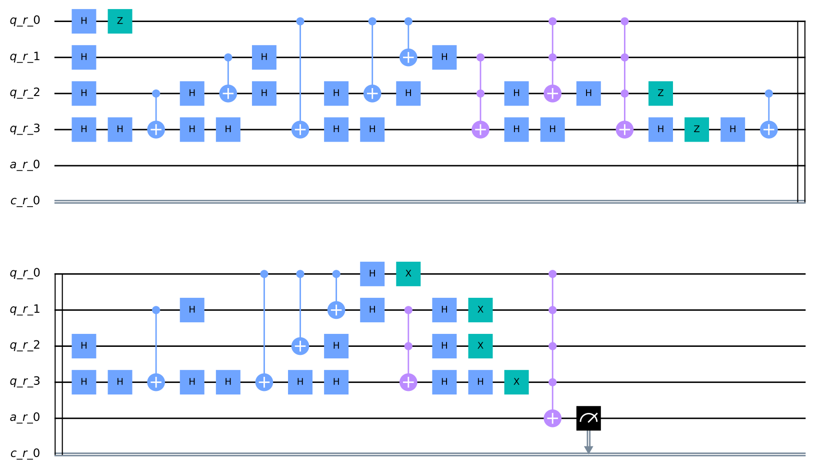

Fig. 3. Quantum circuit programmatically generated encoding both a pattern and a vector weight. The circuit is encoding an image of size 4x4 using 4 qubits for processing, one qubit as ancilla, and one classical bit storing the measurement. This relatively simple circuit already contains 34 layers of quantum gates, this can convey the magnitude of the circuits needed to process the 16x16 images. Currently bounded by the physical implementation of real quantum computers, the model is not limited by the pattern dimensions it can process.

This proportion, which we called *readout*, is compared with another hyper-parameter, which is called threshold. This threshold is used to assign the class. If the readout is less than the threshold the positive class is assigned, otherwise the negative class is assigned to the pattern in turn.

As it is a supervised classification task, during the training step, it will be evaluated if the assigned class is correct. If it is, we simply continue with the next pattern. In the case it is incorrectly classified, depending on its real class,

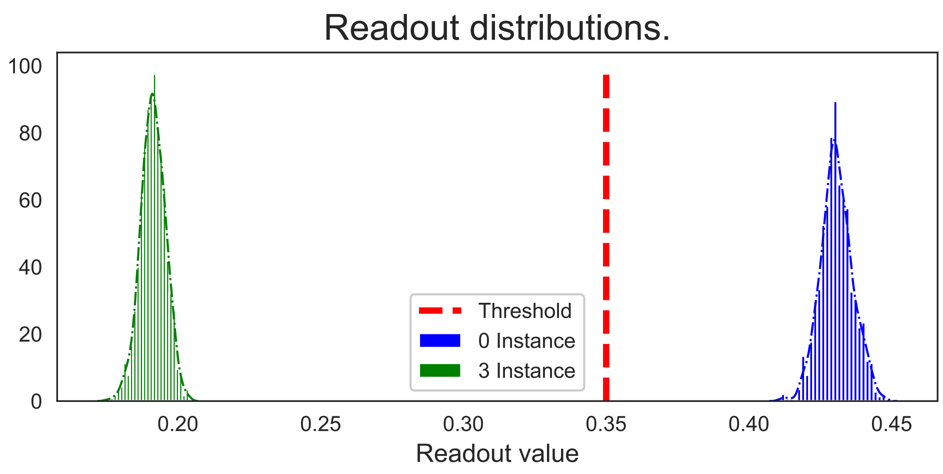

Fig. 4. Readouts distributions after a thousand iterations of the same circuit using the final weight vector for the 3/0 binary classifier. The distribution at the left is for an instance of the class 3 and is well below the threshold. The distribution at the right is for an instance of class 0, this time, above the threshold.

a hyper-parameter will be used, in a similar fashion to the traditional learning rate in neural networks, which defines the proportion of change in the weight vector. There is a learning rate for the positive class and another one for the negative class.

As usual, this procedure can be repeated for an arbitrary number of epochs, where one epoch means that the classifier has seen the entire training set. Or also, as it was done, the training can be finished earlier if a critical value has been reached in a certain metric that we seek to optimize.

The whole process, aiming for efficiency, is simulated using Qiskit [11], however, the entire process is suitable and ready to be executed on a real quantum machine.

Although we implemented and followed the model described by [10], we diverge from them in the sense that we do not test the model in an *ad hoc* dataset. Furthermore, the *ad hoc* dataset was split by means of a previously and arbitrarily selected weight vector. In other words, the classification task was already known to be solvable because it was constructed to do so, but in this work, we do not assume or fabricate such property and let the model converge to a solution.

3 Experimental Results and Discussion

In this section we present the experimental results of the classification model described above. The model was tested in both Optical Recognition of Handwritten Digits Data Set and Semeion Handwritten Digit Data Set. In case of Digits, Resubstitution Error and Hold-out were used as validation method. For Hold-out, we used the partition offered by the authors [15]. For Resubstitution Error, only the test set was used, acting as both training and test sets. This dataset requires processing, as each attribute has a value between 0 and 16, and the model only works with binary values a threshold was applied to binarize the patterns as follows:

$$\text{binarized pixel} = \begin{cases} 0, & \text{if original pixel value} < 10 \\ 1, & \text{otherwise} \end{cases} \tag{12}$$

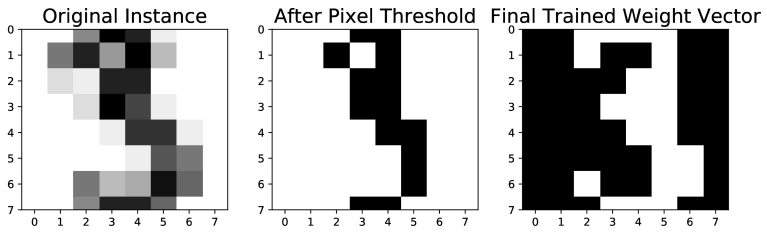

Fig. 5. At the left, there is an original instance from the Digits Dataset. In the middle, there is the result of the binarization threshold. And at the right, there is the final weight vector for the 0/3 classifier. The vector tends to take the form of sort of a mask for one of the classes.

This threshold is itself a new hyper-parameter that can be optimized. In the case of Semeion, Resubstitution Error was used due to the relatively small number of available patterns. This dataset does not need processing as the attributes are already binary. In each dataset, binary classifiers of two styles are trained and tested: class vs class, also known as One vs One (OvO), which in this case represents a balanced classes problem, and class vs the rest also known as One vs All (OvA) which represents an unbalanced class problem. The results are shown in several tables.

In Table 3, the diagonal represents the trivial classification of one single class. It is interesting to note although we can use the upper or lower half to classify the reflexive class, i.e. use the trained positive/negative classifier to try to classify the negative/positive problem, we did not get good results in this scenario. This makes sense when the vector weight for each binary classifier is inspected since it tends to take the form of a sort of mask resembling the instances of the negative class.

In Table 4, we keep track of some performance measures, including the Area Under the Curve (AUC). It is evident from the Recall measure that the quantum model can distinguish the positive class from all the other instances with good accuracy. The ratio of minority class against the rest is approximately 1:100. This result is promising for medical applications where datasets are generally heavily imbalanced. We show, in Table 5 the increase in accuracy performance when a different validation method is used. It is also notable the latent power of generalization because in this case the evaluation was recorded upon a test set containing instances not seen during the training set.

In Table 6, we show the usual metrics, where is notorious the benefit gained by the Hold-out method for this classification model in this dataset. We noted an improvement in seven out of the ten classes. This validation method gives us more confidence in the generalization power that this model might have.

In Table 7 and in Table 8, we give the already known performance metrics. Acceptable performance can be seen in almost every classification task, but it is notorious the decrease compared to the Digits dataset. This drop can be explained by the fact that each instance lives in a bigger dimensional space, and therefore the quantum circuit needed to process each pattern is also bigger and more complex quantum-gates-wise. Nevertheless, we must keep in mind although both datasets might seem similar they are in fact different and we should not expect the same performance in both of them.

Table 3. Resubstitution Error by balanced class classification.

	Digits Resubstitution Error									
	Negative Class									
	0	1	2	3	4	5	6	7	8	9
Positive Class										
0	1.0	0.725	0.794	0.714	0.671	0.769	0.353	0.834	0.678	0.589
1	0.377	1.0	0.660	0.849	0.749	0.780	0.898	0.833	0.609	0.759
2	0.833	0.710	1.0	0.730	0.804	0.894	0.631	0.764	0.564	0.820
3	0.933	0.753	0.833	1.0	0.780	0.887	0.807	0.676	0.624	0.652
4	0.974	0.567	0.837	0.758	1.0	0.785	0.792	0.780	0.749	0.764
5	0.886	0.785	0.830	0.715	0.779	1.0	0.953	0.695	0.800	0.640
6	0.635	0.807	0.843	0.807	0.682	0.785	1.0	0.883	0.540	0.828
7	0.983	0.653	0.803	0.776	0.875	0.797	0.905	1.0	0.728	0.832
8	0.752	0.480	0.706	0.792	0.783	0.651	0.830	0.597	1.0	0.666
9	0.564	0.812	0.815	0.696	0.883	0.527	0.678	0.813	0.788	1.0

4 Conclusions and Future Work

In this work, the performance of a fully quantum machine learning model in real datasets was tested. The evaluation is generally favorable, providing positive feedback that QML is promising.

Table 4. Resubstitution Error by heavily imbalanced class classification.

Digits Resubstitution Error					
Positive Class	Recall	Accuracy	Precision	F1	AUC
0	0.983	0.303	0.122	0.218	0.605
1	0.857	0.399	0.129	0.224	0.602
2	0.966	0.377	0.133	0.234	0.639
3	0.814	0.328	0.112	0.198	0.543
4	0.955	0.420	0.143	0.249	0.657
5	0.873	0.359	0.123	0.216	0.587
6	0.845	0.368	0.121	0.212	0.580
7	0.837	0.417	0.128	0.222	0.604
8	0.724	0.388	0.107	0.186	0.538
9	0.861	0.419	0.132	0.229	0.615

Table 5. Accuracy measure by balanced class classification.

Digits Hold-out										
	Negative Class									
	0	1	2	3	4	5	6	7	8	9
Positive Class										
0	1.0	0.844	0.833	0.900	0.896	0.833	0.924	0.831	0.732	0.731
1	0.969	1.0	0.852	0.863	0.818	0.826	0.914	0.853	0.775	0.779
2	0.895	0.824	1.0	0.841	0.810	0.860	0.877	0.828	0.740	0.843
3	0.883	0.819	0.730	1.0	0.920	0.854	0.964	0.823	0.851	0.719
4	0.933	0.815	0.899	0.840	1.0	0.876	0.790	0.819	0.819	0.811
5	0.933	0.892	0.793	0.802	0.856	1.0	0.964	0.872	0.811	0.765
6	0.827	0.809	0.843	0.925	0.825	0.953	1.0	0.902	0.839	0.739
7	0.974	0.878	0.876	0.859	0.794	0.869	0.977	1.0	0.813	0.871
8	0.889	0.682	0.860	0.826	0.814	0.823	0.842	0.804	1.0	0.824
9	0.849	0.850	0.907	0.746	0.822	0.850	0.955	0.835	0.836	1.0

With this work, we provided a real validation to the quantum model beyond the theoretical correctness and the usual proof of concept. This showed the potential real-world application of the QML in the near future.

Table 6. Accuracy by heavily imbalanced class classification.

Digits Hold-out					
Positive Class	Recall	Accuracy	Precision	F1	AUC
0	0.955	0.206	0.107	0.192	0.539
1	0.972	0.218	0.112	0.201	0.552
2	0.858	0.212	0.098	0.176	0.500
3	0.803	0.188	0.093	0.167	0.460
4	0.994	0.176	0.108	0.195	0.539
5	0.950	0.249	0.114	0.204	0.560
6	1.0	0.154	0.106	0.192	0.529
7	0.843	0.346	0.116	0.204	0.567
8	0.965	0.204	0.105	0.190	0.544
9	0.966	0.193	0.107	0.193	0.536

Table 7. Resubstitution Error by balanced class classification.

Semeion Resubtitution Error										
	Negative Class									
	0	1	2	3	4	5	6	7	8	9
Positive Class										
0	1.0	0.851	0.806	0.890	0.788	0.840	0.593	0.843	0.835	0.680
1	0.947	1.0	0.728	0.869	0.352	0.763	0.832	0.700	0.608	0.650
2	0.937	0.707	1.0	0.767	0.793	0.707	0.706	0.716	0.601	0.637
3	0.912	0.797	0.672	1.0	0.759	0.757	0.853	0.712	0.525	0.624
4	0.981	0.801	0.793	0.837	1.0	0.793	0.751	0.749	0.515	0.664
5	0.890	0.794	0.672	0.657	0.693	1.0	0.721	0.703	0.528	0.570
6	0.757	0.842	0.800	0.881	0.636	0.809	1.0	0.805	0.506	0.692
7	0.946	0.734	0.624	0.779	0.702	0.722	0.724	1.0	0.530	0.667
8	0.854	0.728	0.786	0.866	0.655	0.671	0.781	0.683	1.0	0.610
9	0.905	0.809	0.763	0.769	0.689	0.611	0.780	0.737	0.607	1.0

However, it is in its early stages and in order for it to be competitive against traditional Machine Learning, there is still a gap which this work seeks to bridge.

Some points that would contribute to moving the QML to more mature stages are an increment in the number of available qubits to perform computation,

Table 8. Resubstitution Error by heavily imbalanced class classification.

Semeion Resubtitution Error					
Positive Class	Recall	Accuracy	Precision	F1	AUC
0	0.770	0.193	0.090	0.161	0.449
1	0.962	0.230	0.113	0.202	0.555
2	0.849	0.222	0.1	0.178	0.500
3	0.937	0.212	0.106	0.191	0.534
4	0.807	0.183	0.092	0.166	0.460
5	0.937	0.202	0.105	0.189	0.528
6	0.925	0.197	0.105	0.189	0.520
7	0.873	0.230	0.102	0.183	0.516
8	0.864	0.210	0.097	0.175	0.502
9	0.873	0.211	0.100	0.180	0.506

reduce noise during the application of quantum gates, and mainly a feasible, scalable, and robust codification to map classical inputs to their quantum representation.

As future work, we would propose to extend the biclass to multiclass classification by means of the naive extension One vs One and One vs All as a baseline. A modification in the quantum circuit generation would be proposed to allow the coding of images at three channels depth, that is, in color. It would also be interesting to implement one of the proposals for quantum convolutional layers for features extraction.

References

1. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86, pp. 2278–2323 (1998)
2. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv: 1512.03385.
3. Yamamoto, A.Y., Sundqvist, K.M., Li, P., Harris, H.R.: Simulation of a multidimensional input quantum perceptron. Quantum Inf Process, 17, pp. 128 (2018)
4. Schuld, M., Sinayskiy, I., Petruccione, F.: Simulating a perceptron on a quantum computer. Physics Letters A., 379, pp. 660–663 (2015)
5. Henderson, M., Shakya, S., Pradhan, S., Cook, T.: Quanvolutional neural networks: Powering image recognition with quantum circuits, arXiv: 1904.04767 (2019)
6. Piat, S., Usher, N., Severini, S., Herbster, M., Mansi, T., Mountney, P.: Image classification with quantum pre-training and auto-encoders. International Journal of Quantum Information. 16, 1840009 (2018)
7. Cong, I., Choi, S., Lukin, M.D.: Quantum convolutional neural networks. Nature Physics, 15, pp. 1273—1278 (2019)

8. Zhao, J., Zhang, Y. H., Shao, C. P., Wu, Y.C., Guo, G.C., Guo, G.P.: Building quantum neural networks based on a swap test. Phys. Rev. A., 100, 012334 (2019)
9. Dang, Y., Jiang, N., Hu, H., Ji, Z., Zhang, W.: Image classification based on quantum K-Nearest-Neighbor algorithm. Quantum Information Processing, 17, 239 (2018)
10. Francesco, T., Chiara, M., Dario, G., Daniele, B.: An artificial neuron implemented on an actual quantum processor. NPJ Quantum Information, 5(1) (2019)
11. Aleksandrowicz, G., Alexander, T., Barkoutsos, P., Bello, L., Ben-Haim, Y., Bucher, D., Cabrera-Hernández, F.J., Carballo-Franquis, J.: Qiskit: An Open-source Framework for Quantum Computing (2019)
12. Qiskit.org: Hybrid quantum-classical Neural Networks with PyTorch and Qiskit, `https://qiskit.org/textbook/ch-machine-learning/machine-learning-qiskit-pytorch.html` (2019)
13. Jeswal, S.K., Chakraverty, S.: Recent developments and applications in quantum neural network: A review. Archives of Computational Methods in Engineering, pp. 1–15 (2018)
14. Zhou, J., Gan, Q., Krzyżak, A., Suen, C.Y.: Recognition of handwritten numerals by quantum neural network with fuzzy features. IJDAR, 2, pp. 30—36 (1999)
15. Alpaydin, E., Kaynak, C.: Cascading Classifiers. Kybernetika, 34, pp. 369–374 (1997)
16. Rossi, M., Huber, M., Bruß, D., Macchiavello, C.: Quantum hypergraph states. New Journal of Physics, 15, 113022 (2013)
17. UCI Machine Learning Repository: Semeion Handwritten Digit Data Set. `https://archive.ics.uci.edu/ml/datasets/semeion+handwritten+digit` (2019)