University of Zaragoza

Departament of Condensed Matter Physics

Undergraduate Dissertation in Physics

QUANTUM MACHINE LEARNING FOR DISEASE DIAGNOSIS

Author: JUAN MANUEL PÉREZ GARCÍA DE CARELLÁN

Director: Dr. DAVID ZUECO LAINEZ

Co-Director: Dr. RAFAEL DEL HOYO ALONSO

July 2023

Academic Year 2022–2023

I would like to express my gratitude to all my family and friends, from whom I have received unwavering support and encouragement throughout this four-year journey.

Also, I would like to give thanks to David, Rafael and Irene, for all the help they have provided me throughout the creation of this dissertation, and especially to Sebastián, for investing so much time and dedication in my learning experience.

Contents

Introduction

In our everyday lives, artificial intelligence (AI) has become deeply ingrained, seamlessly integrated into various aspects of our routines. From voice assistants guiding us through our schedules to personalized recommendations shaping our online experiences, AI has become a ubiquitous presence [6]. This normalization of AI reflects the immense strides we have made in computing capabilities and the relentless pursuit of scientific advancements.

The rapid evolution of computing power has fueled the curiosity and aspirations of scientists to develop algorithms and systems that continually push the boundaries of what AI can achieve. While classical approaches have yielded impressive results, the limitations of conventional computing models have led researchers to explore alternative avenues. It is within this context that the field of quantum computing has emerged, harnessing the principles of quantum mechanics to revolutionize AI.

Quantum machine learning (QML) has emerged as a promising paradigm, integrating the power of quantum computing with the capacity to tackle complex problems. Traditional neural networks, which form the backbone of classical machine learning, have paved the way for quantum neural networks (QNNs) that leverage quantum phenomena.

The application of QML extends beyond academic exercises, finding relevance in real-world challenges. From optimizing material design processes to enhancing financial forecasting , QML demonstrates its prowess in addressing non-academic problems. One area that holds significant interest is disease diagnosis. By leveraging the computational advantages offered by quantum systems, we can develop models capable of analyzing complex medical data with unprecedented accuracy and efficiency.

The intersection of quantum mechanics, AI, and healthcare opens exciting possibilities for disease diagnosis. By fusing the cutting-edge techniques of QML with the growing wealth of medical knowledge, we can develop innovative approaches that have the potential to revolutionize healthcare systems. Early and accurate disease diagnosis can significantly impact patient outcomes, enabling timely interventions and personalized treatment plans.

In this dissertation, we explore the field of quantum machine learning with a practical objective: to develop models that can classify various activities performed by individuals. The next step would be to utilize these models for early detection and prevention of muscular diseases, particularly those associated with poor posture, such as sarcopenia. Ultimately, this could contribute to improving the quality of life for individuals.

Objectives and outline

The primary objective of this dissertation is to study and implement quantum neural networks for the development of systems capable of performing machine learning tasks. Specifically, the focus will be on leveraging quantum neural networks to classify human activities in order to be able analyze postural abnormalities in individuals, with the ultimate goal of preventing the risk of associated diseases. By harnessing the power of quantum computing principles and combining them with machine learning techniques, we aim to create advanced systems capable of accurately assessing and identifying posture-related issues, leading to improved health and well-being.

To do so, we want to develop a deep understanding of supervised learning techniques through the utilization of artificial neural networks, while establishing connections with the fundamental principles of quantum mechanics and the time evolution of a quantum system.

First of all, we aim to gain a thorough understanding of classical neural network models. This entails delving into the inner workings of classical networks, understanding their mathematical foundations, and exploring their applications in real-world scenarios.

This research aims to incorporate insights from quantum mechanics into the study of neural networks, specifically focusing on quantum neural network models. By leveraging concepts from quantum computing and quantum information theory, we seek to explore how quantum-inspired architectures can enhance the capabilities of traditional neural networks.

Building upon the knowledge gained from classical and quantum neural network models, the next objective is to develop a hybrid neural network model. This hybrid architecture will combine classical and quantum elements to exploit the advantages of both paradigms. By integrating classical and quantum components, we aim to achieve enhanced performance and explore the potential for solving complex problems more efficiently.

To validate the effectiveness of the developed hybrid network, real-world datasets will be employed. The objective is to apply the hybrid model to a problem within a specific domain and evaluate its performance against established classical models.

Additionally, this research aims to compare two different implementations of neural network models. In addition to the hybrid network developed by Sebastián Roca [20], a PhD student working in the group of David Zueco, a model developed by the *Instituto Tecnológico de Aragón* (*ITAINNOVA*) will be utilized for comparative purposes. This comparison will provide valuable insights into the strengths and weaknesses of each approach, facilitating a comprehensive understanding of the hybrid model's advantages and limitations.

Finally, by integrating all these components, our objective is to derive meaningful conclusions from the obtained results, providing us with a deeper understanding of the efficacy of the methods employed in this dissertation. Additionally, potential improvements for the model will be proposed, aiming to pave the way for future studies in this direction and contribute to the advancement of artificial intelligence in the context of healthcare.

By accomplishing these objectives, this dissertation endeavors to contribute to the field of machine learning by advancing our understanding of supervised learning through artificial neural networks. Furthermore, it aims to explore the integration of quantum principles into neural network models, leading to the development of innovative hybrid architectures. The comparative analysis will provide valuable insights for selecting the most suitable model for specific applications, and the conclusions drawn will contribute to the broader body of knowledge in the field.

1 Time evolution in Quantum Mechanics

The dynamics for any quantum system are governed by a certain Hamiltonian operator, $\hat{\mathcal{H}}$ [1]. The state of the system can be described using a wavefunction Ψ, and its evolution is given by the time-dependent Schrödinger equation:

$$i\hbar\frac{\partial}{\partial t}|\Psi(t)\rangle = \hat{\mathcal{H}}|\Psi(t)\rangle, \tag{1}$$

where t represents time and $|\Psi(t)\rangle$ denotes the state vector of the quantum system.

From here, we would like to define the time evolution operator. We start by considering a system which at the moment t_0 is in the state $|\Psi, t_0\rangle$. We suppose that at a later moment t the system is described by the state $|\Psi, t_0; t\rangle$, $(t > t_0)$. We now want to see how state vectors evolve when time goes on, by defining the time evolution operator $\hat{\mathcal{U}}(t, t_0)$:

$$|\Psi, t_0; t\rangle = \hat{\mathcal{U}}(t, t_0)|\Psi, t_0\rangle, \tag{2}$$

which must satisfy physically relevant conditions.

The first comes from the normalization of the states, which is independent of time. This is because in general, we cannot expect the probability for the system being in a specific state to remain constant. For example, given an observable $\hat{A}$ and its set of eigenstates $\{a'\}$ and eigenvectors $\{|a'\rangle\}$, we can expand the state vector at time t_0 and time t in terms of them as follows:

$$\begin{aligned} |\Psi, t_0\rangle &= \sum_{a'} c_{a'}(t_0)|a'\rangle \\ |\Psi, t_0; t\rangle &= \sum_{a'} c_{a'}(t)|a'\rangle, \end{aligned} \tag{3}$$

where in general $|c_{a'}(t)| \neq |c_{a'}(t_0)|$. But we know

$$\sum_{a'} |c_{a'}(t_0)|^2 = \sum_{a'} |c_{a'}(t)|^2 = 1, \tag{4}$$

so the normalization of of states happens to be independent of time, leaving us with the result that the evolution operator must be unitary:

$$\begin{aligned} \langle\Psi, t_0|\Psi, t_0\rangle &= \langle\Psi, t_0; t|\Psi, t_0; t\rangle \\ &= \langle\Psi, t_0|\hat{\mathcal{U}}^\dagger(t, t_0)\hat{\mathcal{U}}(t, t_0)|\Psi, t_0\rangle \\ &\Rightarrow \boxed{\hat{\mathcal{U}}^\dagger(t, t_0)\hat{\mathcal{U}}(t, t_0) = \mathbb{I}} \end{aligned} \tag{5}$$

The second condition is the composition property, which implies the following, for given times $t_2 > t_1 > t_0$:

$$\hat{\mathcal{U}}(t_2, t_0) = \hat{\mathcal{U}}(t_2, t_1)\hat{\mathcal{U}}(t_1, t_0) \tag{6}$$

Since time is a continuous parameter we obviously have

$$\lim_{t\to t_0} |\Psi, t_0; t\rangle = |\Psi, t_0\rangle \;\Rightarrow\; \lim_{dt\to 0} \hat{\mathcal{U}}(t_0 + dt, t_0) = \mathbb{I} \tag{7}$$

So we can assume the deviations of the operator $\hat{\mathcal{U}}(t_0 + dt, t_0)$ to be of the order dt. We then set

$$\hat{\mathcal{U}}(t_0 + dt, t_0) = \mathbb{I} - i\Omega dt, \tag{8}$$

where Ω is a Hermitean operator that must have frequency dimension, so it must be multiplied by a factor before associating it with the Hamiltonian operator: $\hat{\mathcal{H}} = \hbar\Omega$, or

$$\hat{\mathcal{U}}(t_0 + dt, t_0) = \mathbb{I} - \frac{i\hat{\mathcal{H}}dt}{\hbar} \tag{9}$$

Finally, applying the composition property we get

$$\begin{aligned}\hat{\mathcal{U}}(t + dt, t_0) &= \hat{\mathcal{U}}(t + dt, t)\hat{\mathcal{U}}(t, t_0)\\ &= \left(\mathbb{I} - \frac{i\hat{\mathcal{H}}dt}{\hbar}\right)\hat{\mathcal{U}}(t, t_0),\end{aligned} \tag{10}$$

and if we reorganise the equation and expand the left hand side as the Taylor series we end up with the Schrödinger equation for the time evolution operator:

$$\begin{aligned}\hat{\mathcal{U}}(t + dt, t_0) - \hat{\mathcal{U}}(t, t_0) &= -i\left(\frac{\hat{\mathcal{H}}}{\hbar}dt\hat{\mathcal{U}}(t, t_0)\right)\\ &\Rightarrow \boxed{i\hbar\frac{\partial}{\partial t}\hat{\mathcal{U}}(t, t_0) = \hat{\mathcal{H}}\hat{\mathcal{U}}(t, t_0)},\end{aligned} \tag{11}$$

which coincides with the Schrödinger equation for the state vector of the quantum system (Eq. (1)) if we multiply both sides by the state vector $|\Psi, t_0\rangle$.

In quantum computing, when we talk about applying operations to the units of information (qubits), what is physically happening is that the system is being driven by some Hamiltonian engineered according to the technological implementation of each laboratory so every gate applied to any qubit can be described by acting with the time evolution operator onto the state of the system.

2 Machine Learning

To solve a problem on a computer, we need an algorithm, which is a sequence of instructions that should be carried out to transform the input to output. For example, one can devise an algorithm for sorting. The input is a set of numbers and the output is their ordered list. For the same task, there may be various algorithms and we may be interested in finding the most efficient one, requiring the least number of instructions or memory, or both.

For some tasks, however, we do not have an algorithm. For example, to tell spam emails from legitimate emails. We know what the input is: an email document that in the simplest case is a file of characters. We know the output should be a yes or no answer indicating whether the message is spam or not. But we do not know how to transform the input to the output. What can be considered spam changes in time and from individual to individual.

What we lack in knowledge, we make up for in data. We can easily compile thousands of example messages some of which we know to be spam and what we want is to "learn" what constitutes spam from them. In other words, we would like the computer (machine) to automatically extract the algorithm (learn) for this task [5].

Machine learning approaches are traditionally divided into the three broad categories: supervised learning, unsupervised learning and reinforcement learning. These categories provide a framework for understanding different approaches to solving learning problems and guiding the development of algorithms and techniques in the field of machine learning.

- **Supervised learning (SL):** [6] The available data consists of labeled examples, meaning that each data point contains features and an associated label. The goal of supervised learning algorithms is creating a function that maps feature vectors (inputs) to labels (output), based on example input-output pairs. A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples, called test samples.

- **Unsupervised learning:** In this case the set of data contains only inputs, and the algorithms find structure in the data, like grouping or clustering of data points. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data.

- **Reinforced learning:** [2] It is concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward. It differs from supervised learning in not needing labelled input/output pairs, and in not needing sub-optimal actions to be explicitly corrected. Instead, the focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).

2.1 Supervised learning

There is a vast number of algorithms that allow us to carry out supervised learning and the work they do could be summarised in the way we describe now [11].

Given a set of N training examples of the form $(\vec{x_1}, y_1), \cdots, (\vec{x_N}, y_N)$ such that $\vec{x_i}$ is the feature vector of the i-th example and y_i is its label, a learning algorithm seeks a function $g : X \to Y$ where X is the input space and Y is the output space.

Although machine learning and in particular supervised learning have been transformative in some fields, machine learning programs often fail to deliver expected results. Reasons for this are numerous: lack of (suitable) data, lack of access to the data, data bias, or overfitting are some examples. This final example merits a more thorough explanation to illuminate certain challenges that have arisen throughout the project.

2.2 Overfitting

In mathematical modeling, overfitting refers to "the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit to additional data or predict future observations reliably" [18].

An overfitted model is a mathematical model that contains more parameters than can be justified by the data. The essence of overfitting is to have unknowingly extracted some of the residual variation (i.e., the noise) as if that variation represented underlying model structure. In the specific case that concerns us regarding machine learning, overfitting occurs when a model begins to "memorize" training data rather than "learning" to generalize from a trend.

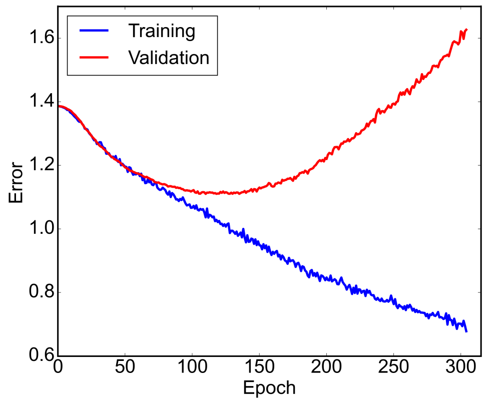


Figure 1: *Overfitting visualized in supervised training over the number of training cycles (epochs).*

Generally, a learning algorithm is said to overfit relative to a simpler one if it is more accurate in fitting known data (hindsight) but less accurate in predicting new data (foresight). The goal is that the algorithm will also perform well on predicting the output when fed "validation data" that was not encountered during its training. As shown in Figure 1, if the validation error increases while the training error steadily decreases, then a situation of overfitting may have occurred. The best predictive and fitted model would be where the validation error has its global minimum.

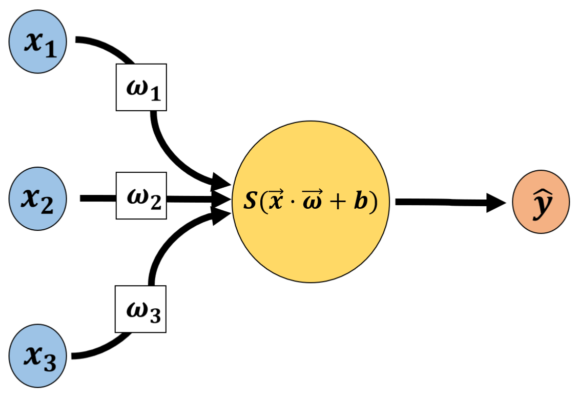
Performing machine learning involves creating a model, which is trained on some training data and then can process additional data to make predictions. In the field of machine learning, there are several types of models utilized for various tasks, including decision trees, support-vector machines, and regression analysis.

However, one particular model that has garnered significant interest and is especially relevant to our project is artificial neural networks. These networks have been widely employed and extensively studied due to their ability to learn complex patterns and relationships in data, making them a powerful tool in many applications.

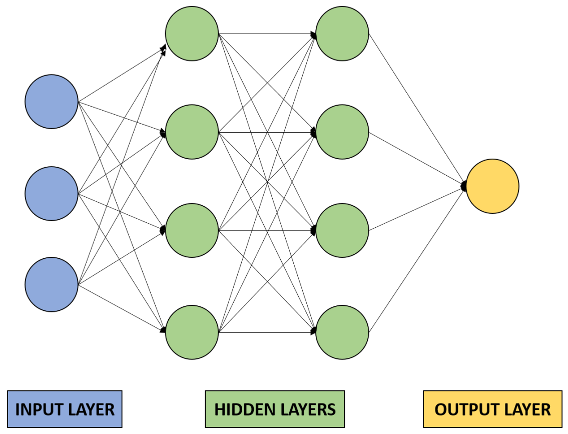
3 Artificial Neural Networks

Artificial neural networks are computational models inspired by the human brain, designed to process information and perform machine learning tasks, such as capture patterns and relationships in data, allowing them to perform tasks as for example classification or natural language processing.

The neuron is the fundamental processing unit found within a neural network. Similar to a biological neuron, these neurons have input connections through which they receive external stimuli, the input values. With these values, the neuron performs an internal calculation and generates an output value [8, 10].



(a)



(b)

Figure 2: (a) *Representation of a neuron and its internal calculations.* (b) *Sketch of a fully connected neural network made of several layers.*

Internally, the neuron utilizes n input values, which correspond to the number of features of our dataset, to generate a weighted sum of them. The weighting of each input x_i is determined by the weight ω_i assigned to each input connection. In other words, each connection that reaches our neuron has an associated value that determines how strongly each input variable affects the neuron. These weights are the parameters of our model. In fact, it can be said that what the neuron does internally is something like a linear regression model, where we have input variables that define a line or hyperplane that can be adjusted by varying the inclination using our parameters.

Additionally, linear regression models, and therefore neurons, include an offset term b that allows us to vertically shift the lines. This term is called bias and is essentially represented as another connection to the neuron, but the variable assigned to it is always set to one and can be manipulated by adjusting the value of b. In this way, our neuron now acts exactly like a linear regression model.

$$z = x_1 \cdot \omega_1 + x_2 \cdot \omega_2 + \cdots + x_n \cdot \omega_n + b = \vec{x} \cdot \vec{\omega} + b \tag{12}$$

However, the difference comes with one final step, which consists of passing the value of z defined in Eq. (12) to a non-linear activation function so that the neural network can successfully approximate functions that do not follow linearity or it can successfully predict the class of a function that is divided by a decision boundary which is not linear. There are many different possibilities and they depend on the specific problem. One commonly used example is the sigmoid or logistic function, defined by

$$\hat{y} = \sigma(z) = \frac{1}{1+e^{-z}}, \tag{13}$$

where σ denotes the sigmoid activation function and the output we get after the forward propagation is known as predicted value $\hat{y}$. Other examples of used activation functions are softmax, ReLU or tanh.

Continuing on, the learning algorithm can be broken down into two main components: backpropagation and optimization. These components are essential in the training process of the model, allowing it to adjust its parameters and improve its performance over time.

Backpropagation is short for backward propagation of errors, and it refers to the algorithm for computing the gradient of the loss function with respect to the weights and bias. To know an estimation of how far are we from our desired solution, a loss function is used. For regression problems, the mean squared error is generally chosen as the loss function. For classification problems, however, cross entropy is usually chosen.

We use cross entropy loss when adjusting the model weights during training. The aim is to minimize the loss. The cross entropy function, H can be calculated using the probabilities of the target distribution, P, and the approximation of the target distribution, Q, as follows:

$$H(P,Q) = -\sum_{\vec{x}\in\text{classes}} P(\vec{x}) \cdot \log Q(\vec{x}) \tag{14}$$

Following with our example, in multi-class classification, the raw outputs of the neural network are passed through the activation function, which then outputs a vector of predicted probabilities over the input classes. As the labels are one-hot, meaning that each label is represented by a vector of binary values, where only one element in the vector is set to 1 (hot) and the rest are set to 0 (cold), only the positive class keeps its term in the loss. In this manner, we can eliminate the elements of the summation that are zero due to the target labels, resulting in a simplified form of Eq. (14):

$$H(\vec{x}) = -log\ S(\vec{x}), \tag{15}$$

where $S(x)$ refers to the activation function.

The final step is optimization, which is the selection of the best element from some set of available alternatives, which in our case, is the selection of best weights and bias. One example of optimization algorithm (we will work with it later on) is the gradient descent, which changes the weights and bias, proportional to the negative of the gradient of the cost function with respect to the corresponding weight or bias. Learning rate (α) is a hyperparameter which is used to control how much the weights and bias are changed.

The weights and bias are updated as follows and the backpropagation and gradient descent is repeated until convergence:

$$\begin{aligned} \omega_i &= \omega_i - (\alpha \cdot \frac{\partial \mathcal{L}}{\partial \omega_i}), \\ b &= b - (\alpha \cdot \frac{\partial \mathcal{L}}{\partial b}), \end{aligned} \tag{16}$$

where $\mathcal{L}$ represents the loss function.

3.1 Convolutional Neural Network

Within neural networks, we find a specialized type called convolutional neural networks (CNNs). CNNs [12] are architectures specifically designed to process grid-structured data, such as images or time series data. The distinguishing feature of CNNs is the convolutional layer, which applies convolution operations using filters to extract local image features, such as edges or textures.

Convolution is a mathematical operation that allows the merging of two sets of information. In the case of CNNs, convolution is applied to the input data (image) to filter the information and produce what is called a feature map.

Specifically, each new pixel we generate is created by placing a matrix of numbers (feature detector or kernel) over the original image, where the values of each pixel of the matrix are multiplied and summed to obtain a new value for the pixel being filtered. This iterative process applied to all pixels in the image results in the creation of a feature map. The obtained result will depend on how we configure the parameters of our filter, allowing us to study a specific characteristic of the image.

For example in a black and white picture, if we have a 3x3 kernel and multiply each pixel value, including its neighbors, by 1/9, we will be calculating the average of the colors of the pixels. This operation, in essence, blurs the original image, a technique commonly used to implement effects seen in many editing software applications today. A sketch of the process is shown in Figure 3.

The remarkable aspect of CNNs is that they are capable, through machine learning, of learning the values of the kernel that enable the network to perform its task more effectively.

3.2 Long Short-Term Memory Network

Long Short-Term Memory (LSTM) networks [3] differ from traditional neural networks in their ability to handle sequential data and capture long-term dependencies. Unlike standard neural networks, which process input data independently at each time step, LSTMs introduce memory cells that can store and access information over multiple time steps.

The memory cells maintain an internal state and utilize various gates (input gate, forget gate, and output gate) to regulate the flow of information. These gates allow the network to

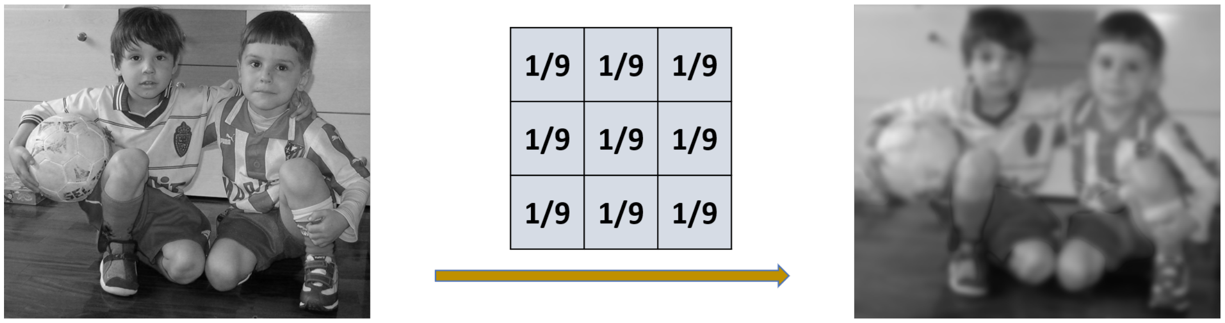


Figure 3: *Visualization of the blurring effect of the shown 3x3 kernel on a picture.*

selectively remember or forget information at different time steps, enabling the model to retain relevant information and discard unnecessary details.

The input data is sequentially processed through the LSTM layers, where each layer updates its internal state based on the current input and the previous state. This makes them well-suited for tasks such as speech recognition or time series analysis, among others.

3.3 Quantum Neural Network

As we already know, for supervised learning, given the input data and its labels, some function maps the input data into some space, usually referred as feature space. In the case of our quantum neural network, the data is encoded in a qudit together with the variational parameters that will be optimised in order to minimise a cost function. As in the classical analog, this cost function typically reflects how much the prediction made by our classifier deviates from the class to which each data point actually corresponds.

In this work we consider how the evolution of a d-level system can be used in the framework of supervised learning. Essentially, we will cover the ideas developed in [15, 20].

Our d-level system could be a spin molecular qudit, which can be controlled through resonant microwave pulses with superconducting resonators [16, 17]. The operations or gates that this type of architecture allows us to implement are rotations in the (x, y) plane relative to the two levels resonant with the control pulse, usually contiguous to each other due to selection rules. That is, using monochromatic pulses, we can control one transition at a time, provided that the anharmonicity condition is met (each of the transitions has a resonance frequency different from the others). The mathematical description of this operation is given in Eq. (17):

$$\hat{R}_{k,l}(\theta,\phi) = exp\left(-i\frac{\theta}{2}(cos\ \phi \cdot \hat{\sigma}^x_{k,l} - sin\ \phi \cdot \hat{\sigma}^y_{k,l})\right), \tag{17}$$

where the $\hat{\sigma}$ matrices represent the Pauli operators for each pair of levels k, l:

$$\begin{aligned} \hat{\sigma}^x_{k,l} &= |k\rangle\langle l| + |l\rangle\langle k| \\ \hat{\sigma}^y_{k,l} &= -i|k\rangle\langle l| + i|l\rangle\langle k|, \end{aligned} \tag{18}$$

where $k < l$ is assumed.

The evolution operator is then parameterized. For each rotation between two levels, we have two parameters (θ, ϕ) to optimize in order to minimize the cost or loss function. Thus, a layer of rotations involving all accessible levels in our molecule will comprise $n_p = 2(d-1)$ parameters.

The ansatz consists of applying L layers of our rotations with a certain encoding of the data to be classified and the parameters to be optimised. For each data point, a state will be generated and compared with reference states, corresponding to each of the different classes. The reference state with which the generated state has more overlap will correspond to the class assigned to the data point. The operator corresponding to this ansatz is:

$$\hat{U}(\vec{x};\vec{\varphi}) = \prod_{l}^{L}\left(\prod_{i}^{d-1}\hat{R}^{l}_{i,i+1}(\vec{x};\vec{\varphi})\right) \tag{19}$$

where the angles (θ, ϕ) from the definition given in Eq.(17) will be in general a function of the data points, $\vec{x}$, and the optimization angles $\vec{\varphi}$.

We can link this section with Section 1 about the time evolution in quantum mechanics. These ansatz rotations (which are unitary operators as we derived for time evolution operators), as described in Eq. (19), form the basis for the time evolution of our quantum system. By carefully controlling and manipulating the quantum states using these rotations, we can effectively map the input data into the feature space and optimize the variational parameters to minimize the cost function. The connection between the time evolution operator and the operations performed on our qudit highlights the fundamental role of quantum mechanics in our approach to supervised learning.

In this case, our goal is to ensure that points belonging to the same class generate wavefunctions that are close to each other in Hilbert space. To achieve this, we define a reference state for each class, and the cost function we aim to minimize is defined in Eq. (20).

$$\mathcal{L}(\vec{x};\vec{\varphi}) = \sum_{n=1}^{N}\left(1 - |\langle\psi(\vec{x};\vec{\varphi})_n|\psi_n^R\rangle|^2\right), \tag{20}$$

where N is the total number of data points, $|\psi(\vec{x};\vec{\varphi})\rangle = \hat{U}(\vec{x};\vec{\varphi})|0\rangle$ and $|\psi_R\rangle$ is the reference state corresponding to the $\vec{x}$ point.

The problem of finding the reference states consists on finding maximally orthogonal states, which is not trivial at all. For this purpose, we have resorted to the use of genetic algorithms, a type of algorithm in which an initial population, in this case a set of states that aspire to be our desired set, is modified generation after generation according to biological inspired rules. At the end of this process, the chosen set is the one which presents the best fitness, a function that measures in our case the orthogonality between all the pairs of states as well as the homogeneity of these overlaps [4]. This process is described in detail in [20] and implemented in the code developed by Sebastián Roca. However, it is not something that has been extensively explored in this study.

The loss function in Eq. (20) has a physical-geometric interpretation: we want to maximize the overlap between the state generated by our temporal evolution from a classical data point and the reference state assigned to the corresponding class. This overlap, also known as fidelity between quantum states, represents the proximity between both states in Hilbert space.

4 Analysis

4.1 Datasets

In the development of this study, two different datasets have been used. First, we used the MNIST (Modified National Institute of Standards and Technology) [7] digit set, which consists of an extensive collection of grayscale images representing handwritten digits. The goal of using this first dataset was to analyze the impact of the number of quantum layers and the dimension of the qudit on a dataset covering all digits (from 0 to 9), while verifying the correct functionality of the code.

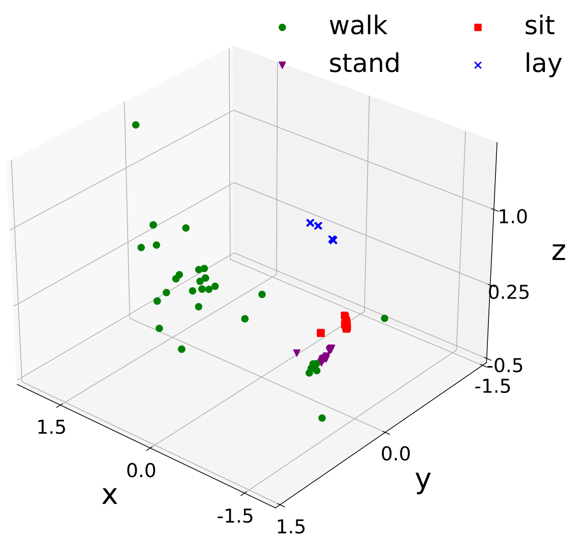


Figure 4: *Visualization of some of the input data points representeing accelerations, for 4 of the activities found in [19].*

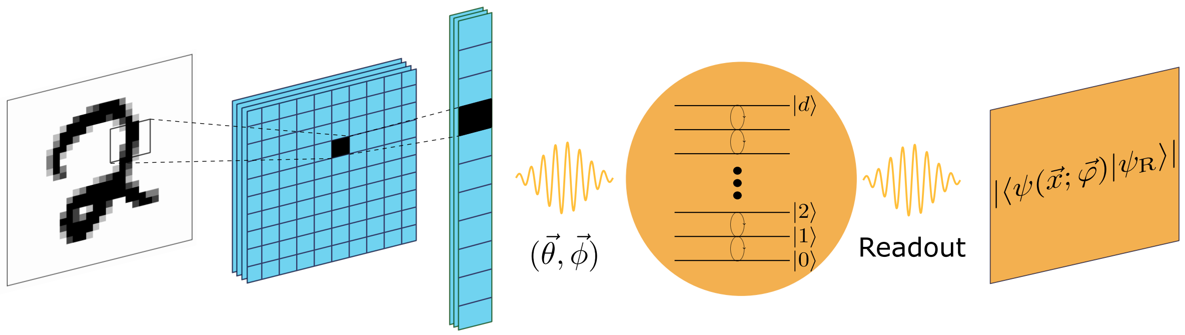
Subsequently, we focused on working with the dataset of greatest relevance for the study. This dataset was provided by the *ITAINNOVA* and was obtained from [19]. Specifically, from this dataset, we extracted the data related to the three angular velocities and the three accelerations (corresponding to the Cartesian x, y, and z axes) measured by a sensor positioned on the ankle of a group consisting of 17 distinct young individuals without any impairments. These subjects performed a total of 11 activities, with three attempts each. These activities include walking, jumping, falling in various forms, lying down, standing up and sitting down, among others.

The dataset consists of time series data with varying durations depending on the performed activity and individual. Activities that are static have a larger number of temporal sequences. Figure 4 shows the temporal distribution the of accelerations (in units of g, the acceleration due to gravity), for various activities, highlighting a higher dispersion of points corresponding to the walking activity compared to the other static activities.

4.2 Hybrid Neural Network

Now we present the hybrid model we have used for the process of supervised learning, which is summarized in Figure 5. The code by Sebastián Roca has been implemented in Python using PyTorch, a machine learning framework based on the Torch library.

First of all, the data is passed through three convolutional layers with different input and output dimensions. In each of them, and for all the remaining layers, the activation function is the ReLU (Rectified Linear Unit), which consists of a ramp function with unit slope and value



$(\vec{\theta}, \vec{\phi})$
$|d\rangle$
$|2\rangle$
$|1\rangle$
$|0\rangle$
Readout
$|\langle\psi(\vec{x};\vec{\varphi})|\psi_{\mathrm{R}}\rangle|$

Figure 5: *Visual depiction of the hybrid neural network structure.*

0 for negative values of x, as defined in Eq. (21).

$$\mathrm{ReLU}(x) \equiv x^+ = max(0, x) = \frac{x + |x|}{2} = \begin{cases} x & \text{if} \quad x > 0 \\ 0 & \text{if} \quad \text{otherwise} \end{cases} \tag{21}$$

After the last convolutional layer, two operations are applied to the output feature map: dropout and max pooling.

The dropout layer [9] randomly sets a fraction of the input elements to zero during training. This helps prevent overfitting by reducing the reliance on specific features and promoting generalization. Dropout acts as a regularization technique, forcing the network to learn robust representations by randomly "dropping out" some of the information. This helps to prevent the network from relying too heavily on specific features or correlations that may be present in the training data but are not representative of the underlying patterns..

Following the dropout layer, max pooling [13] is performed. This operation divides the feature map into non-overlapping regions and retains the maximum value within each region. By downsampling the feature map, max pooling extracts the most prominent features and reduces the spatial dimensions. It helps to capture the most relevant information while discarding less significant details, leading to more efficient computation and improved spatial invariance.

After the max pooling operation, the output tensor is further processed through subsequent layers in the neural network. First, the tensor is flattened to a 1-dimensional vector. This step is necessary to match the dimensions required for the subsequent fully connected layers.

Next, the flattened tensor is passed through two fully connected layers (each neuron in one layer is connected to every neuron in the subsequent layer), like the ones we introduced in Section 3 and are sketched in Figure 2. Lastly, the output of the last fully connected layer is passed to our hybrid layer, which incorporates quantum-inspired techniques we have previously mentioned in Section 3.3 and further processes the features extracted by the previous layers.

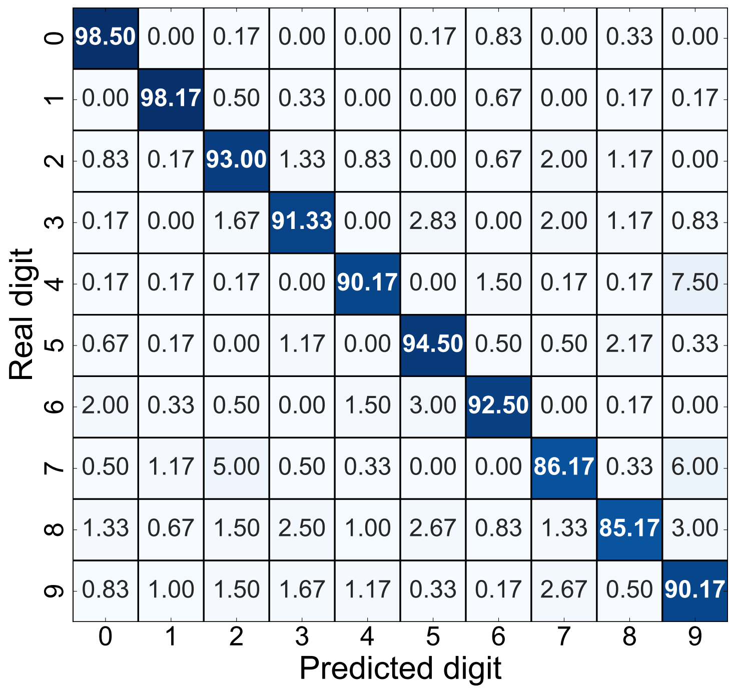
To summarize, the overall purpose of this sequence of operations is to transform the input data through convolutional layers, apply downsampling through max pooling, and then pass the processed features through fully connected layers and the hybrid layer.

Finally, once the data has passed through all the layers of the model, the optimization process takes place. In this regard, the code implementation utilizes the Adam optimizer, short for Adaptive Moment Estimation. The underlying principle of Adam is to dynamically adjust the learning rate for each parameter based on its gradient and the historical gradient magnitudes. The Adam optimizer combines the strengths of various optimization algorithms, including the one we described previously.

By adaptively modifying the learning rate, Adam ensures efficient parameter updates that are tailored to the specific characteristics of each parameter. This adaptability leads to faster convergence and improved overall performance during the training process.

4.3 Results

As mentioned above, we embarked on simulations using images of the digits 0 to 9. The results obtained after simulations with the purely classical model were very satisfactory, as can be seen in Figure 6. In this figure, the confusion matrix reveals a 94.19% accuracy in the test, highlighting the exceptional performance achieved.



	0	1	2	3	4	5	6	7	8	9
0	98.50	0.00	0.17	0.00	0.00	0.17	0.83	0.00	0.33	0.00
1	0.00	98.17	0.50	0.33	0.00	0.00	0.67	0.00	0.17	0.17
2	0.83	0.17	93.00	1.33	0.83	0.00	0.67	2.00	1.17	0.00
3	0.17	0.00	1.67	91.33	0.00	2.83	0.00	2.00	1.17	0.83
4	0.17	0.17	0.17	0.00	90.17	0.00	1.50	0.17	0.17	7.50
5	0.67	0.17	0.00	1.17	0.00	94.50	0.50	0.50	2.17	0.33
6	2.00	0.33	0.50	0.00	1.50	3.00	92.50	0.00	0.17	0.00
7	0.50	1.17	5.00	0.50	0.33	0.00	0.00	86.17	0.33	6.00
8	1.33	0.67	1.50	2.50	1.00	2.67	0.83	1.33	85.17	3.00
9	0.83	1.00	1.50	1.67	1.17	0.33	0.17	2.67	0.50	90.17

Figure 6: **Classical NN results for MNIST dataset, for $\alpha = 1.0 \cdot 10^{-5}$, epochs = 150.** *The confusion matrix provides insight into the accuracy of predictions for each digit.*

Prior to the analysis of the activity dataset, a study of the effect of quantum layers was carried out by comparing the classical and hybrid models, varying the number of quantum layers (L) and the qudit dimension (d).

The results, as evidenced in Figure 7, reveal a clear direct relationship between increasing the number of quantum layers or the dimension of the qudit and the improvement of the validation accuracy, although in all cases they do not surpass the results achieved by the purely classical model. Specifically, it appears that the effect of the qudit dimension enhances the results in a more noticeable manner.

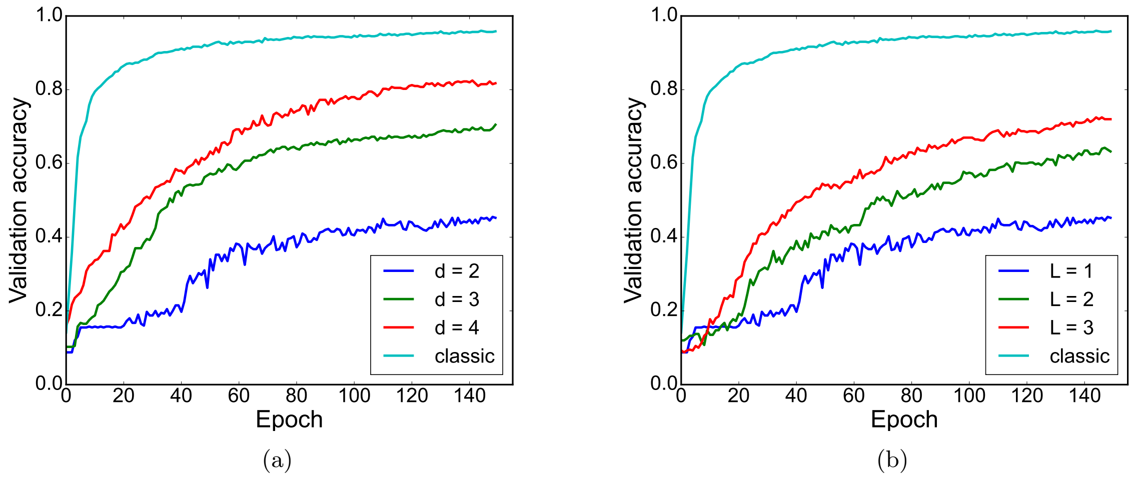


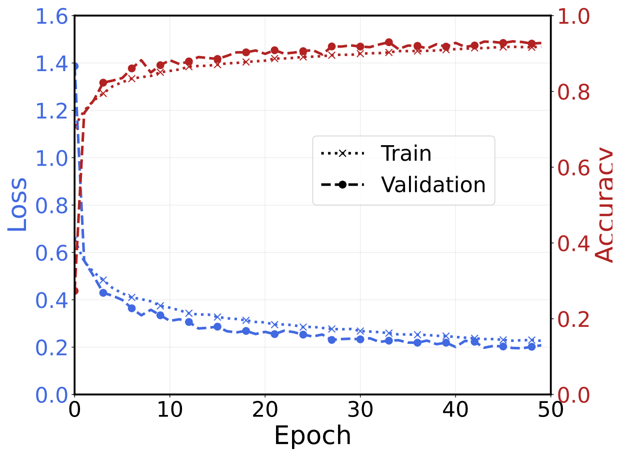
Figure 7: **Study of the influence of L and d for the MNIST dataset results, for $\boldsymbol{\alpha = 1.0 \cdot 10^{-5}}$, epochs = 150.** (a) *Evolution of the validation accuracy for several values of d and fixed $L = 1$.* (b) *Evolution of the validation accuracy for several values of L and fixed $d = 2$.*

After verifying the correct functioning of the code, we proceeded to perform machine learning using the activity dataset. In order to simplify the process, we decided to select only four activities out of the eleven available in the dataset. The choice was based on those activities that had a larger number of time sequences, which is equivalent to a larger amount of data, in the hope of obtaining more optimal results. After analyzing and filtering the dataset, we were left with the following activities: walking, standing, sitting and lying down.

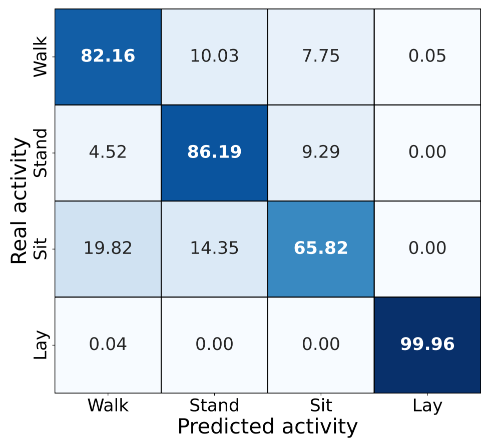
In addition to reducing the number of classes, we also reduced the number of features and the time duration of the 4 chosen activities. The features were reduced from six (corresponding to three angular velocities and three accelerations) to only three accelerations. This decision was based on the shared characteristic of three out of the four selected activities, namely their static nature. Consequently, the angular velocities did not provide significant relevant information. The time duration of the activities that had more time sequences was shortened to ensure that all the data describing human activity, which would be fed into the neural network, had uniform length as per the requirement.

One of our main goals is to compare the results obtained from the hybrid network with those obtained from the model used by the *ITAINNOVA*, which was a LSTM network.

In order to do the comparison, we first modified the code provided by the *ITAINNOVA* to accept the modified dataset. We conducted tests to find the optimal learning rate for the model and then visualized the results in a confusion matrix. This matrix, along with the evolution of metrics for a learning rate of 0.0025, can be observed in Figure 8, where a test accuracy of 83.53% was achieved.



(a)



(b)

Figure 8: **LSTM results for the activity dataset for $\alpha = 2.5 \cdot 10^{-3}$, epochs = 50.** (a) *Evolution of the metrics.* (b) *Confusion matrix.*

Regarding this obtained result, attention should be drawn to Figure 4. In it, it seems clear that the presented classes are easily separable at first glance. However, it is worth noting that the image only displays a small subset of temporal sequences compared to the entirety of the data (more than 850 points per activity and individual, compared to the 30 points per activity drawn in the Figure). If we could visualize all the data used in the same manner, we would observe how the points from different classes cluster together more, making the classification process more challenging.

Also, in that same Figure, it is evident that the “lay” activity is notably separated from the rest, and this is likely due to the similarity in ankle position among the other three activities, making recognition more difficult.

Moving on to the hybrid model, similarly to the example of the digits, we conducted a study on the effect of the qudit dimension and the number of quantum layers on the outcome. The main challenge we encountered was the limited amount of data compared to the available MNIST dataset. This resulted in simulations being highly sensitive to repetitions, and the results differed significantly between two identical configurations. To address this issue, we performed statistical analysis on the data by repeating the same configuration multiple times.

We then took the average of these repetitions to obtain the evolution of the metrics, as implemented in the code provided by the *ITAINNOVA*. This approach allowed us to mitigate the variability and obtain a more reliable assessment of the performance.

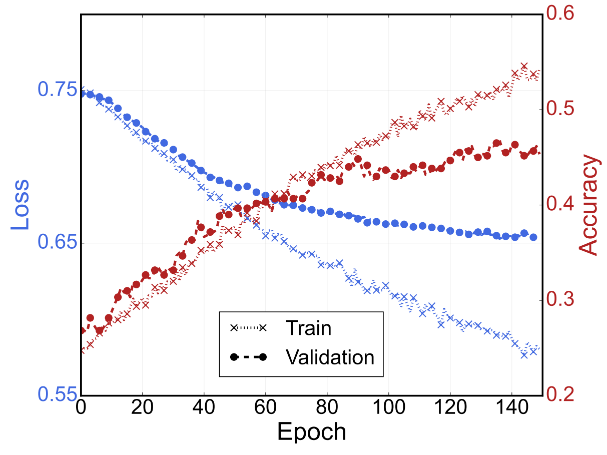
The next step, once again, involved conducting a learning rate (α) sweep for various (d, L) pairs in order to find the one that yields the best learning outcome. As shown in Table 1, in all cases, we obtained lower accuracy compared to the *ITAINNOVA* results (85.53%).

L	**d**	α	**Accuracy (%)**
1	2	$1.0 \cdot 10^{-4}$	68.33
1	3	$7.5 \cdot 10^{-5}$	68.33
1	4	$6.0 \cdot 10^{-5}$	68.33
2	4	$5.0 \cdot 10^{-5}$	71.67
3	4	$5.0 \cdot 10^{-5}$	63.33

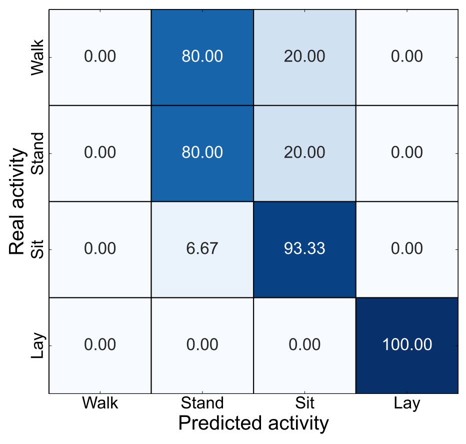
Table 1: **Hybrid NN results for the activity dataset, epochs = 150**. *List of the best test accuracies achieved for different L and d combinations, and their corresponding α value.*

The hybrid model achieved the best results with a combination of 2 layers and the dimension of the qudit equal to 4.

Based on the Table, it appears that increasing the dimension of the qudit did not impact the test accuracy, unlike increasing the number or layers, which did cause and impact, but it differs from what we could have expected from the results that we got with the MNIST dataset. The reason for choosing $d = 4$ to study the accuracy is that 4 is also the number of classes (activities), and defining maximally orthogonal reference states for each class is straightforward: if the number of classes matches the number of levels of the qudit, the most immediate option is to assign each class a basis vector.



(a)



	Walk	Stand	Sit	Lay
Walk	0.00	80.00	20.00	0.00
Stand	0.00	80.00	20.00	0.00
Sit	0.00	6.67	93.33	0.00
Lay	0.00	0.00	0.00	100.00

(b)

Figure 9: **Hybrid NN results for the activity dataset for $\alpha = 6.0 \cdot 10^{-5}$, epochs = 150, $L = 1$, $d = 4$**. (a) *Evolution of the metrics.* (b) *Confusion matrix.*

Finally, Figure 9 depicts the evolution of accuracies and losses over the epochs, along with the confusion matrix for a chosen pair $(L, d) = (1, 4)$. The confusion matrix suggests that the machine learning process may have encountered a local minimum and struggled to accurately differentiate the "walk" activity, probably due to the big dispersion of the points describing the acceleration at each time step. However, similar to the results obtained by *ITAINNOVA*, the "lay" activity was correctly classified, while the remaining three activities exhibited some degree of misclassification, indicating promising signs of overall correct classification.

Finally, it is worth noting that the code from *ITTAINOVA* achieves higher accuracies and lower losses, and in addition to that, convergence is reached more quickly, placing the hybrid model behind the LSTM model.

5 Conclusions

Through the development of this Undergraduate Dissertation we have presented an example of quantum machine learning on a human activity recognition dataset.

In the first part of the dissertation, we laid the foundation for temporal evolution in quantum mechanics by describing the operator $\hat{\mathcal{U}}(t, t_0)$. We then delved into the popular topic of machine learning, specifically focusing on supervised learning, which is the main subject of this dissertation. Next, we explained the theoretical principles of the tool we utilized for this work: artificial neural networks. We provided an overview of two significant types of neural networks, CNNs and LSTMs, highlighting their relevance to our work. This led us to the central point of our study: quantum neural networks. By exploring quantum neural networks, we were able to establish connections between our previous knowledge of quantum mechanics and the temporal evolution operator, ultimately creating a powerful tool capable of performing machine learning tasks.

Finally, we applied the acquired knowledge to a real-world problem: activity recognition using supervised learning, utilizing data collected by an ankle sensor that captured the movements of multiple subjects. The analysis of the results involved comparing the LSTM model from *ITAINNOVA* with our hybrid model for a set of 4 chosen activities. This comparison revealed that the hybrid network performed worse in classifying the different activities. However, both models demonstrated high accuracy in classifying the “lay” activity compared to the other three activities (“walk”, “sit” and “stand”). Following this comparison, we conducted a brief study on the impact of qudit dimension and the number of quantum layers in the hybrid network on classification accuracy. The best result was obtained for $d = 4$ and $L = 2$, likely due to the ease of creating reference states when the qudit dimension matches the number of classes.

The intersection of the conducted study with advancements in healthcare opens up opportunities for diagnosing diseases related to postural abnormalities. However, this study merely serves as a foundational basis that could be further advanced. Possible future advancements in addressing the problem would involve increasing the dimensionality. As explained in the previous section, we have limited ourselves to a small portion of the dataset obtained from [19] and explained in [14]. The next step would be to increase the number of sensors in other parts of the body and incorporate camera images. All of this could be combined with a more advanced neural model for time series analysis, with a growing emphasis on quantum aspects as progress is made in quantum computing.

References

[1] Claude Cohen-Tannoudji, Bernard Diu, and Franck Laloë. *Quantum Mechanics*. 2nd ed. Vol. 1. Wiley, 1977.

[2] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. "Reinforcement learning: A survey". In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285.

[3] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9 (1997).

[4] Melanie Mitchell. *An introduction to genetic algorithms.* MIT press, 1998.

[5] Ethem Alpaydin. *Introduction to machine learning.* 2nd ed. MIT Press, 2010. Chap. 1, pp. 1, 3.

[6] Peter Norvig Stuart J. Russell. *Artificial Intelligence: A Modern Approach.* 3rd ed. Prentice Hall, 2010.

[7] Li Deng. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[8] Jeff Heaton. *Introduction to the Math of Neural Networks.* Heaton Research, Inc., 2012.

[9] Li Wan et al. "Regularization of Neural Networks using DropConnect". In: *Proceedings of the 30th International Conference on Machine Learning.* Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. PMLR, 2013, pp. 1058–1066.

[10] Michael A Nielsen. *Neural networks and deep learning.* Vol. 25. Determination press San Francisco, CA, USA, 2015.

[11] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning.* 2nd ed. MIT Press, 2018.

[12] Rikiya Yamashita, Richard Kinh Gian Do Mizuho Nishio, and Kaori Togashi. "Convolutional neural networks: an overview and application in radiology". In: *Insights Imaging* 9 (2018).

[13] Jason Brownlee. "A Gentle Introduction to Pooling Layers for Convolutional Neural Networks". In: *Deep Learning for Computer Vision* (2019).

[14] José Núñez-Martínez et al. "UP-Fall Detection Dataset: A Multimodal Approach". In: *Sensors* (2019).

[15] Adrián Pérez-Salinas et al. "Data re-uploading for a universal quantum classifier". In: *Quantum* 4 (2020), p. 226.

[16] Alberto Castro et al. "Optimal control of molecular spin qudits". In: *Physical Review Applied* (2022).

[17] Victor Rollano et al. *High cooperativity coupling to nuclear spins on a circuit quantum electrodynamics architecture.* 2022.

[18] *Definition of overfitting by http://oxforddictionaries.com/.*

[19] *https://sites.google.com/up.edu.mx/har-up/.*

[20] *The original work made by S. Roca, D. Zueco et al. has not been published yet. Nonetheless, all the code and methods have been shared with the student for the development of this dissertation.*