On a quantum inspired approach to train machine learning models

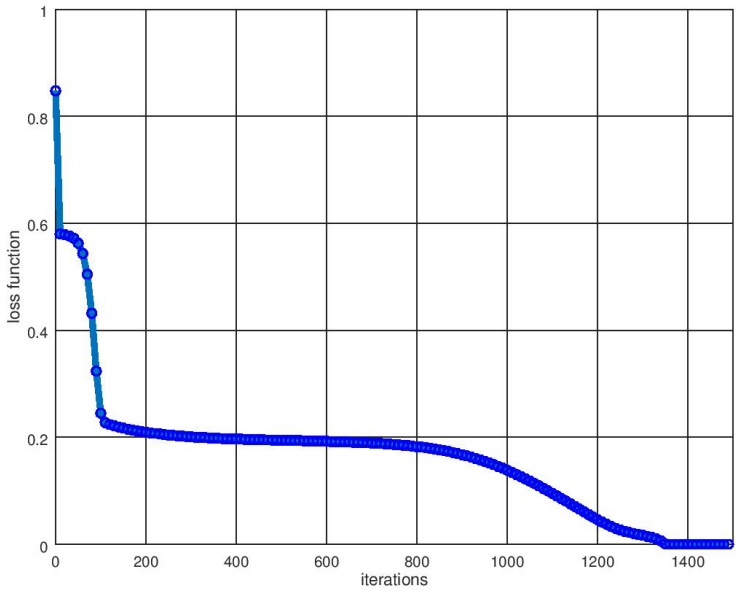
Jean Michel Sellier[1]

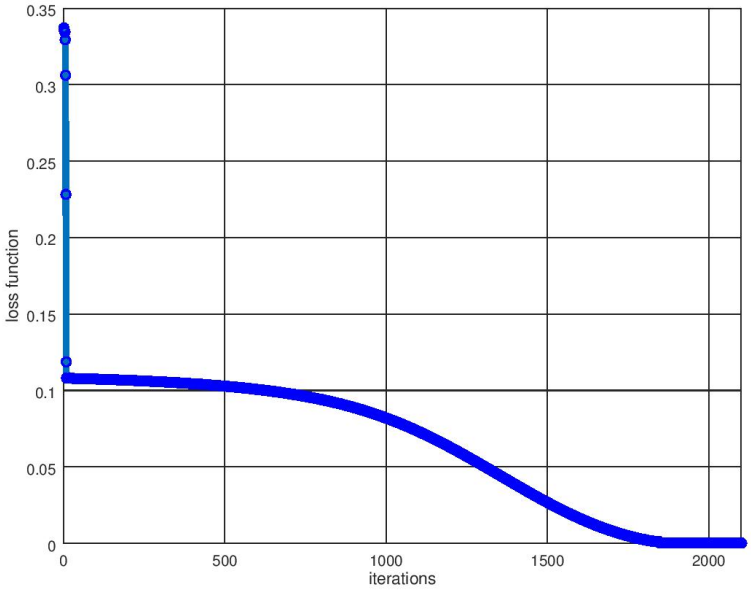
[1]Ericsson

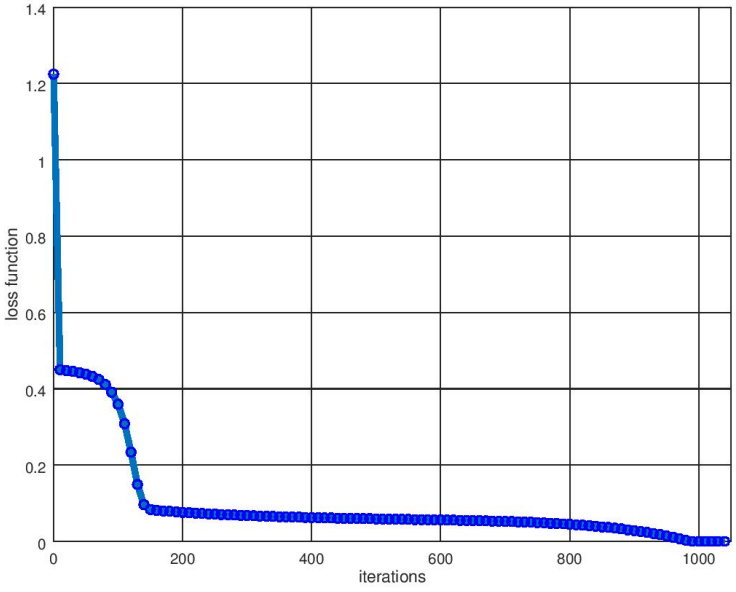
August 30, 2023

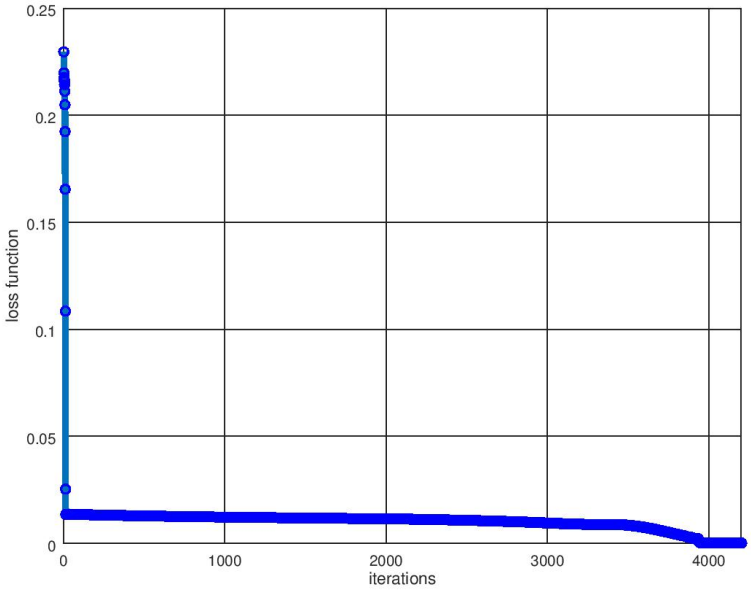
Abstract

In this work, a novel technique to train machine learning models is introduced, which is based on digital simulations of certain types of quantum systems. This represents a drastic departure from the standard approach which, to these days, is based on the use of actual physical quantum systems. Thus, to provide a clear context, a proper introduction to the field of quantum machine learning is first provided. Then, we proceed with a detailed description of our proposed method. To conclude, some preliminary, yet compelling, results are presented and discussed. Although at a seminal stage, the author firmly believes that this approach could represent a valid and robust alternative to the way machine learning models are trained today.









On a quantum inspired approach to train machine learning models

Jean Michel Sellier*,**

***Abstract*—In this work, a novel technique to train machine learning models is introduced, which is based on digital simulations of certain types of quantum systems. This represents a drastic departure from the standard approach which, to these days, is based on the use of actual physical quantum systems. Thus, to provide a clear context, a proper introduction to the field of quantum machine learning is first provided. Then, we proceed with a detailed description of our proposed method. To conclude, some preliminary, yet compelling, results are presented and discussed. Although at a seminal stage, the author firmly believes that this approach could represent a valid and robust alternative to the way machine learning models are trained today.**

***Index Terms*—Quantum machine learning, Quantum inspired methods, Machine learning, Quantum computing, Optimization problems, Artificial neural networks**

I. The necessity for quantum inspired machine learning techniques

In 1965, Gordon Moore, co-founder of Intel Corporation, was asked to contribute to the 55-th anniversary issue of Electronics magazine with a prediction on the future of the semiconductor industry over the next ten years. His response was a brief, yet very influent, article entitled "Cramming more components onto integrated circuits" [1]. In that work, Moore speculated that by 1975 it would be possible to contain as many as $65,000$ components on a single chip. More generally, he also predicted that the number of transistors that can be fit on a computer chip would double every two years. In turn, the computing power of the average chip would increase accordingly and the cost to produce such device would come down with time. Moore's Law was correct and served as a guide for innovation for over 50 years, helping to create effective roadmaps in the field of semiconductor devices for decades, a fact that surprised Moore himself, as mentioned in an interview in 2015. Historically, the first Intel microprocessor, the Intel 4004, had only $2,300$ transistors while, in 2019, the same company managed to pack over 100 million transistors in a square millimeter (the smallest transistors can now reach active lenghts of around 1nm). Today one can certainly debate on the validity of Moore's law but, without any doubt, it is certain that it will cease to be a sustainable road at some point. In fact, there are experimental indications to believe that we might already have reached the physical limitations of silicon based chips. For instance, in the process of miniaturization, quantum effects are now so prominent that, eventually, they prevent the device to work properly, a problem that has shown to have no simple solution. A further emerging factor threatening the future of Moore's Law is also represented by the growing costs related to manufacturing. Today semiconductor process technology is complex, three-dimensional, and devices require multiple exposures of silicon wafers. Consequently, for either physical or economical reasons, we have reached, or we are going to reach soon, a computational plateau in terms of the number of transistors that one can fit on a single chip. In other words, we might already be living in the post-Moore's Law age, or very close to it, and, without a clear and practical alternative, engineers can no longer increase the computing power of chips. As a direct consequence, the increase in processing power is already slowing down, with no clear sign that this trend will change anytime soon.

Interesting enough, in spite of the current situation with the manufacturing of modern semiconductor devices, computational practitioners seem to happily continue to require immense tasks to be performed, e.g., in the field of machine learning (ML) with the treatment of huge amounts of data. Clearly, a solution is required and many believe it could be provided by the emerging field of quantum machine learning (QML) [2], coming from the combination of ML and quantum computing (QC). However, QML comes with its own set of difficulties as well. For instance, a clear and generally accepted definition is still to be provided, although it seems that, today, the vast majority of the community is moving in one specific direction based on the use of specific and peculiar physical hardware such as quantum bits, or qubits, to obtain computational advantages in ML tasks. All sort of advantages are expected, for instance, execution speedups, reduction of memory consumption, etc. The problem with this approach, though, is that practical and reliable quantum computing hardware is not at reach yet (most likely, not any time soon) and for good reasons. These conclusions are supported by experimental facts which are clearly and thoroughly described in [2], [3] and, if quantum effects must be exploited in some way, a different approach might be needed (at least at this very preliminary stage of this technology). For instance, some proposed QML algorithms are based on the use of typical quantum effects such as entanglement, coherent transport, and tunnelling [2]. Now, it is very well known that such physical systems are extremely difficult to maintain in practical experimental settings as they require (expensive and cumbersome) cryogenic facilities to cool these systems down close to the absolute zero (i.e., -273.16 Celsius degrees). Moreover, even supposing that such temperatures would be easily reachable, it is well-known that decoherence

Paper prepared on the 17/08/2023.

*Ericsson, Global AI Accelerator, Montréal, Québec, Canada, jean.michel.sellier@ericsson.com

**This work was funded by and supported by the Ericsson Global Artificial Intelligence Accelerator in Montréal. The author thanks Maria Anti for her continuous support and enthusiasm.

eventually enters in the game and destroys the "quantumness" of the system, rendering it to a classical one (therefore losing any eventual quantum advantage). If one has to concretely utilize quantum states to, say, train neural networks, much more sophisticated technological advancements are going to be required which will have to be based on less brittle physical systems [3].

For the reasons depicted above, the main objective of this paper is to introduce an alternative paradigm which, by utilizing only off-the-shelf digital technologies, is capable of providing practical quantum machine learning capabilities. The solution presented in this work is rather simple, effective, and does not exploit any physical quantum systems at all. In more details, the main focus on this method is to train any kind of artificial neural networks (ANN) to perform predictions once provided with a certain amount of data (i.e., the data set).

In practice, in such context, it is always possible to depict a quantum system which can be simulated on a digital machine and, consequently, which can train an ANN by simply reaching its point of minimum energy by evolution in time; in other words the point of minimum energy reached by the system after some time represents the solution of the training problem (i.e., it can final weights and biases of the ANN). In fact, physical systems always evolve in a way that reduces their internal energy, a well-known lesson of Physics. The philosophy is, therefore, drastically different: rather than building physical systems, one simulates such systems. Obviously, to make simulations fast and useful, we must introduce some sort of approximation which, in this work, is based on the density functional theory (DFT), a very well known approach among computational quantum chemists. As it will be (preliminary) shown in this work, this actually represents a practical and realistic way to obtain the quantum state corresponding to the minimum energy of a system. As a matter of fact, in this work, we can show that it is possible to train ML models in practice, in a way that is unprecedented though (to the best of the author's knowledge). This, of course, opens the way towards practical QML in spite of the current lack of quantum computing devices.

In the next section, we start by showing the explaining the field of QML, its current status, and what one could expect from it in a not-too-distant future. We then introduce our suggested approach which, in this work, we will refer to as quantum inspired machine learning (QiML). Afterwards, we present some preliminary, yet compelling, results obtained by applying QiML to several different ML problems. Some conclusions are discussed at the end.

II. Quantum machine learning

Although a definition of QML is yet to come, broadly speaking, one could consider it as the point of convergence between ML and QC, with the hope that this combination can provide some advantages, e.g., in terms of memory usage, execution speed, accuracy, etc. Therefore, in the following sections, we start by introducing the main tenets of both ML and QC, and discuss their possible combination afterwards.

A. Principles of machine learning

The theory of ML [4] plays an important role in both artificial intelligence and statistics. In a broad sense, it can be considered as the discipline which provides computing devices with the ability to learn without having to be explicitly programmed. In practice, the input-output relation of a computer program is derived from a set of training data.

Traditionally, in ML, the term "learning" is usually divided into three different classes: supervised, unsupervised and reinforcement learning. Being very well documented by others, we limit this description to a few comments on these approaches:

- *Supervised learning* is achieved by means of an annotated dataset $\{(x_i, y_i)\}$ for $i = 1, \ldots, N$ which, usually, is provided by a human. Each element x_i is called an input or a feature vector, while y_i, the labels, represent the ground truth utilized by the learning algorithm to build knowledge. Thus, the main purpose of this class of approaches is to produce a model that, given an input vector x, can predict the correct value y. A good example of application is represented by the task of pattern recognition.
- *Unsupervised learning* is obtained by means of a unlabeled dataset $\{x_i\}$ for $i = 1, \ldots, N$. The main goal for the methods in this class is to use the data to extract some meaningful property out of it. Good examples of applications are provided by clustering, dimensionality reduction and generative models.
- In *reinforcement learning*, an agent is immersed in a given environment which can be probed by performing a certain set of actions. The agent is rewarded or punished according to the series of actions it performs. The final goal of the agent is to learn a policy/strategy which is considered optimal when it maximizes the expected average reward.

Although these three classes represent a rich family of powerful and interesting methods, in this work we focus on supervised learning problems only. Training phases are often the most costly part of an ML algorithm and efficient training methods become especially important when dealing with *big data*, which makes it perfect for the approach suggested in this work.

B. Principles of quantum computing

In a general sense, QC can be considered as the discipline which tries to harness the dynamics of physical quantum systems to achieve computational advantages [5]. While many different paradigms exist, the main one in use is certainly the gate or qubit paradigm. A foundational concept of such approach is represented by the quantum bit or, shortly, qubit. In this context, a bit becomes a physical system with two distinct states and a qubit can be seen as a quantum system having two distinct (orthogonal) states. These states are usually labeled as $|0\rangle$ and $|1\rangle$ (Dirac notation), and a qubit can hold one bit of information (just as a classical bit) but, also, a superposition of them, i.e., the state:

$$a\,|0\rangle + b\,|1\rangle$$

where a and b are two complex numbers. One should note that, so far in this discussion, no detail on how to physically realize a qubit is provided. The (input) data is stored in the shape of a register of qubits and processed by applying an ordered set of quantum gates which transform the information stored on the qubits in different ways, depending on the gate applied. The (output) information is, then, extracted by measuring the state of each qubit. Again, one should note that no detail is provided on how these quantum gates are built or on how the extraction of the final outcome (i.e., measurements) is performed.

From this perspective, one can classify QML algorithms in four main families depending on the way they store and process the data. The classification can be: *1)* QQ, where both the data and the way it is processed are of quantum nature, *2)* QC, where the data is stored as a superposition of states (therefore, quantum) while it is processed by a classical algorithms, *3)* CQ, where the data is stored classically but processed by some quantum algorithm, and finally *4)* CC, i.e., both the data and the algorithm are classical [6].

While QC comes with great promises [5], it also is affected by complex technical issues which need to be solved before real QC devices, and not just expensive experimental settings, can come to life. For instance:

- *Decoherence*: to scale properly, the physical system needs to be in the total absence of decoherence effects. Without this hallmark, only a small number of qubits can be built and effectively utilized in practice.
- *Measurements*: any measurement performed destroys the wave function of the quantum system being measured (a puzzling effect known as the wavefunction collapse). While measurements are absolutely necessary to probe the solution embedded in the physical system, an unreasonable number of them might be required to achieve good accuracy [2].
- *Entanglement*: it is an extremely complicated state of matter, difficult to create, and even more difficult to maintain or process.

Therefore, not all methods proposed in the literature will eventually see the light since many of them are likely to be affected by one, or more, of the issues above. A practical example of what is meant is reported in the next section for the sake of clarity. For very clear details on these technical difficulties, the Reader is, once again, invited to read [3].

III. The quantum random access memory

The quantum random access memory (QRAM) is an hypothetical quantum device to store information, suggested in [7]. In more details, the QRAM is theoretically supposed to encode information in superposition of states, by storing N d-dimensional vectors into $\log(Nd)$ qubits in $\mathcal{O}(\log(Nd))$ [1] time by making use of the so-called 'bucket-brigade' architecture. However, in practice the number of physical resources needed to build an actual QRAM scales as $\mathcal{O}(Nd)$ [2]. Moreover, experimental evidences show that:

- It is not clear if it has to be error corrected. In the positive case, though, it has already been shown that an exponential number of components would be required.
- The number of measurements to reconstruct the solution can grow exponentially.
- QRAM do not provide any advantage when the data is not uniformly distributed.

This, in turn, implies that any suggested algorithm based on the use of QRAM is going to be affected by the same issues above and, therefore, it is not going to work in practice. This is the case, for instance, for the HHL method, also known as quantum linear system algorithm (QLSA) [8]. The QLSA is an algorithm which is supposed to solve systems of linear equations such as $AX = B$ with A an $N \times N$ real matrix, X and B two N-dimensional real vectors. The best classical algorithm known, in practical cases, is the well known QR-factorization method which solves a system in $\mathcal{O}(N^3)$ steps. The QLSA promises to solve linear systems in logarithmic time instead. In order to obtain the quantum speedup, though, the following conditions must hold:

- The matrix A must be sparse.
- The classical data must be loaded in quantum superpositions in logarithmic time on the QRAM.
- The output needs an exponentially growing number of measurements to obtain a reasonable accuracy.

Clearly, these limitations, along with the issues affecting the QRAM itself, prevent the development and practical use of such algorithm [2].

We are now ready to introduce our suggested quantum inspired machine learning method.

IV. Quantum inspired machine learning

The solution suggested by the author of this work provides a different paradigm: to avoid the difficulties connected to the use of physical quantum systems, simulations of a certain class of quantum systems, which can be performed on (commercially available) digital computers, are utilized instead. In practice, given an artificial neural network (ANN) to train on a certain data set, we can show that it is always possible to depict a corresponding physical system which can perform the training process by reaching the point of minimum energy of the system itself, by a simple evolution in time. In other words, one exploits the fact that physical systems evolve towards states which minimize their energy and, then, use these final states to extract the solution of the training problem, i.e. it provides the final weights and biases for the initial ANN.

Obviously, not every physical system can be efficiently simulated on a digital machine and therefore, to reach practical and reliable training of ANNs, one is forced to introduce some approximations. In our specific case, it is inspired by the field of density functional theory (DFT), which is a very well known theory among computational quantum chemists (see, e.g., [9] and [10]). In practice, this allows a practical and efficient way to compute the quantum state corresponding to the minimum energy of the system. In fact, in this paper, we are able to show that this approach is capable of training ML models in

[1] The notation $\mathcal{O}(f(n))$ means that the asymptotic scaling of the algorithm is upper-bounded by a function $f(n)$ of the number n of parameters characterizing the problem, i.e., the size of the input.

a way that is unprecedented and this despite the current lack of any actual quantum computing device.

Before proceeding with more technical details, and for the sake of clarity, the Reader should note that the aim of this work is not to obtain any quantum advantage in terms of execution speed or memory usage (as usually promised by many different works in the field of QC) but it is, instead, to introduce a novel technique to obtain qualitatively different ML models. In practice, it is known that QML could, potentially, provide ML models capable of recognizing novel kind of patterns (i.e., quantum) which are difficult (or practically impossible) to recognize by means of classical approaches [11]. The goal of the training technique presented here is to provide a practical way to recognize those quantum patterns without the need of any physical quantum system. To the best of the author's knowledge, this is the first time that approximated digital simulations of quantum systems are proposed as a practical way to train ANNs. Certainly, this represents an important departure from the currently explored paradigm of QML. Consequently, for the sake of clarity, this work does not suggest the construction or utilization of any physical quantum computing device.

This approach brings new and relevant advantages in practical applications:

- Because it utilizes simulated quantum states to train ML models, it is not affected by any of the issues faced by the QML community.
- It allows anyone to use commercially accessible computers to train models in a new qualitatively different way. The method is utlzable right away, there is no need to wait for the existence of any quantum computer.
- Although based on digital simulations of quantum systems, it exploits various quintessential quantum effects such as, e.g., the tunnelling effect. Current (classical) methods, usually some variant of the gradient descent method, can rapidly get stuck into energetic valleys of the loss function. This issue is completely avoided by the proposed approach.

In the next following sections, we introduce every detail required to understand the QiML method. To keep this presentation clear, thus, we start by introducing the problem of training ML models as an optimization problem, with a special focus on ANNs. We then proceed with a quick description of one of the most common solver for optimization pronlems, i.e., the gradient descent method. This allows, afterwards, to introduce its physical interpretation which, in turn, brings to the mathematical definition of the QiML approach.

A. Training ANN as an optimization problem

We start now by focussing on the problem of training an artificial neural network, although this does not represent a limitation of QiML and generalization to train different models is easily obtained. In this context, an ANN is treated a mathematical abstraction of biological neural networks and can be considered as a collection of connected computing units, or artificial neurons, which connection strength is represented by a number known as the weight; the more connections a network has, the more weights are necessary. In particular, feedforward ANNs are constituted of layers of neurons which transfer information from one layer to the next, i.e., from the input layer to the output layer through one, or more, hidden layers.

Every neuron in an ANN is characterized by a discriminant function and an activation function which acts on the discriminant. In more details, if a neuron has a set of inputs $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and a set of weights $\mathbf{w} = (w_1, w_2, \ldots, w_n)$, a common choice for the discriminant function is the quantity

$$z = \sum_{i=1}^{N} w_i x_i \tag{1}$$

(for simplicity we embed the bias of a neuron in the sum by enforcing the condition $x_1 = 1$). There are plenty of possible choices for the activation function, usually indicated as a general (non-linear) function $\sigma = \sigma(z)$.

Once the architecture of a network is defined (i.e., the number of layers, the number of neurons per layer and their connections, the discriminant and activation functions for each neuron), it is possible to mathematically express any ANN as a function of the type below:

$$y = y(\mathbf{x}; \mathbf{w}) \tag{2}$$

with $\mathbf{x}$ representing the input, $\mathbf{w}$ the set of all weights and y being an output value computed by the network (the Reader should note that the variables x and y can be scalars or vectors depending on the case, and one denotes vectors in bold style and scalars in italic style respectively; for instance, in the formula above, $\mathbf{x}$ is a vector and y is a scalar).

Thus, provided some sample set $(x_i; y_i)$, for $i = 1, \ldots, Ns$, usually called the data set, describing the computational goal to be achieved by the network, the problem of training an ANN consists of minimizing some error function, also known as the loss function, which formally reads:

$$E = E(y(\mathbf{x}; \mathbf{w}); (x_i; y_i)) \tag{3}$$

and which depends on the whole set of weights. In practice, this goal is accomplished by looking for the set $\mathbf{w}^*$ which minimizes the error function $E = E(\mathbf{w})$ (in practical cases, the error function is represented by an L_2 norm of some shape). Many different algorithms to reach this goal exist, one of the most popular being the gradient descent method. For the sake of clarity and completeness, in the following, we introduce its the main tenets in the context of training ANNs.

B. The gradient descent method

One of the simplest training algorithms is the gradient descent method, sometimes also known as steepest descent method. One usually starts with some initial guess for the weight vector, often random, denoted by $\mathbf{w}^{(0)}$. The weights are then iteratively updated so that, at the n-th step, one moves in the direction of the negative gradient (i.e., the direction in which the error decreases), evaluated at $\mathbf{w}^{(n)}$:

$$\mathbf{w}^{(n)} = \mathbf{w}^{(n-1)} - \eta \nabla E(\mathbf{w}^{(n-1)}), \tag{4}$$

and the gradient is re-evaluated at each step. The parameter η is called the learning rate, and provided its value is sufficiently small, one expects that the value of E decreases at each successive step, eventually leading to a weight vector at which $\nabla E = 0$ is satisfied.

The updating rule above is reminescent of the well-known Newton second law of Physics:

$$m\mathbf{a} = \mathbf{F} \tag{5}$$

where m is the mass of a particle with acceleration $\mathbf{a}$, and in the presence of an applied force $\mathbf{F}$ which can be written as the gradient of a potential $U = U(x)$, i.e., $\mathbf{F} = -\nabla_x U$, with x being the position.

In practice, by knowing that a is the second derivative of the position x, exploiting the finite difference approach for derivatives, and by integrating formula (5) with respect to time in the range $[t_0, t]$, one gets:

$$\mathbf{x} = \mathbf{x}_0 - \lambda \nabla U(\mathbf{x}_0) \tag{6}$$

with $\lambda = 1/m\,(t–t_0)^2$ (the approximation $mv = -\int_{t_0}^{t} \nabla U(x)dx \approx -\nabla U(x)(t-t_0) = -\nabla U(x)\Delta t$ has been introduced, which is a valid assumption for small temporal ranges). Clearly, formula (6) has the same mathematical shape of formula (4) utilized to update the weights (and biases) of ANNs. Consequently, in a broad sense, one could interpretate the gradient descent method as the simulation of a classical physical system which evolves in time and which, in turn, reduces its internal energy. For instance, it could be interpretated as a mass point falling in a N-dimensional energetic valley represented by the error function or, equivalently, as a set of N one-dimensional mass points each falling in a one-dimensional energetic valley; the two systems, eventually, have to deal with the same number of variables (weights to be trained).

In spite of the great success this method has obtained in many different tasks, it is also affected by certain drawbacks which are difficult to get rid of. For instance, one of its most troubling limitations, or any variant of it for that matter, is represented by the fact that the user is actually forced to come up with a suitable value for the learning rate parameter η, without any practical guiding principle at hand i.e., most often, randomly. Unfortunately, the problems with this method do not end here. For instance, a good initial guess for the weights of a specific learning problem is practically unknown and, in practice, must be set randomly. As a matter of fact, it is well-knwon by the ML practitioners how this can trigger the need for numerous tests to be performed eventually (until good convergence is obtained, essentially). Another well-known issue is represented by the so-called vanishing gradient. In that case, the gradient descent method takes many (too) small steps to reach the minimum and quickly becomes a very inefficient procedure. Similar comments could be provided for the issue of the so-called exploding gradient. The author firmly believes that the approach presented in this work can avoid this sort of issues.

In the next section, we are now ready to introduce the quantum counterpart of the gradient descent approach which, in turn, happens to provide a high-level guideline to understand our suggested approach.

C. Towards a quantum inspired approach

The convertion of a classical system into its quantum counterpart is usually performed by replacing the time-dependent classical equation, in our case equ. (6), with its corresponding Schrödinger equation evolving while keeping the same potential energy $U = U(x)$ [12]. This equation reads:

$$i\hbar \frac{\partial \Psi}{\partial t} = \hat{H}\Psi = \big(-\frac{\hbar^2 \nabla^2}{2m} + U(x) \big) \Psi \tag{7}$$

with $\hbar$ the reduced Planck constant, $\hat{H}$ the Hamiltonian of the system, which is equal to the sum of the kinetic operator and the applied potential. Therefore, instead of approaching this problem by means of the gradient descent method, which in this context roughly corresponds to the simulation of a classical many-body system, one could alternatively utilize simulations of corresponding quantum many-body systems. Consequently, this enables the presence of quantum tunnelling within a simulation, a phenomenon impossible to mimic in the context of classical systems (this is well-known, for instance, in the field of simulations of electrons in relatively semiconductor devices). In turn, tunnelling can (and does) enable a better search in the space of solutions since it is not affected by the typical issues of the gradient descent method (e.g., because of the presence of quantum tunnelling, the search for solutions will not get stuck locally in a potential valley representing some non-optimal local minimum).

Therefore, one can safely conclude that, for any given ANN with N hyper-parameters to be trained, there are N corresponding Schrödinger equations to be simulated, just like there are N updating equations in the gradient descent method. In a broad sense, this is very similar to what one observes in the simulation of chemical systems which is usually performed as a set of Kohn-Sham equations coupled through some potential (in the context of density functional theory, or DFT, [13], [14]). We describe the way these Schrödinger equations are concretely coupled in the next sections.

We now proceed with the description of the discretization scheme used in this work to numerically solve the Schrödinger equation.

D. The finite-difference time-domain method

The Schrödinger equation described in the previous section cannot be treated analytically due to the very complicated shape of the applied potential. Thus, one must recur to numerical methods to extract its time-dependent solution (i.e., the wave function $\Psi = \Psi(x, t)$). Many methods are nowadays available, and, for the sake of a complete validation process, the author of this work has selected the well-known finite-difference time-domain (FDTD) method (see for example [15]) because of its relative simplicity and accuracy. Obviously, this choice does not represent a limitation of the method presented in this paper since other methods could be utilized equivalently.

The FDTD method has a long history in the field of computational electrodynamics, and only recently it has found use in the simulation of quantum mechanical systems as well. In more details, in classical FDTD Maxwell equations are discretized on a grid and solved explicitly using leap-frog integration in time. Since the FDTD algorithm is explicit and local, it is relatively straightforward to implement and parallelize. In turn, applying FDTD to solve the Schrödinger equation yields to a method that is generally straightforward to implement and computationally efficient [15]. Although, non-uniform grids have been utilized for the electrodynamics FDTD method, which could be used for the Schrödinger equation as well, we do not use them in this work for the sake of keeping this presentation as simple as possible (in other words, it does not represent a limitation for this method).

We now introduce the mathematical details of the FDTD method applied to the Schrödinger equation. First, the complex valued wavefunction is split into real and imaginary components:

$$\Psi(\mathbf{x},t) = \Psi_R(\mathbf{x},t) + i\Psi_I(\mathbf{x},t), \tag{8}$$

with Ψ_R and Ψ_I the real and imaginary part of the wave function respectively. The Schrödinger equation then becomes (in a one-dimensional space):

$$\begin{aligned} \frac{\partial \Psi_R}{\partial t} &= -\frac{1}{2m}\hbar^2\nabla^2\Psi_I + V(x)\Psi_I, \\ \frac{\partial \Psi_I}{\partial t} &= -\frac{1}{2m}\hbar^2\nabla^2\Psi_R - V(x)\Psi_R. \end{aligned} \tag{9}$$

The time derivatives is approximated by using central finite differences, which provides the discretized evolution equations [15]. Due to the explicit discretization of the time derivative, there is an upper bound for the time step for stability which needs to be ensured and which depends on both the grid cell size and the maximum absolute value of the potential within the computational domain. Finally, this evolution method can be used to solve general time-dependent problems, including a time-dependent potential.

Although relatively simple, the Reader should note that this is a very robust and accurate method. As a matter of fact, it has been utilized to compute very sophisticated quantities such as the spectrum of atoms and their wave-functions [15]. Having introduced the tools and ideas needed to build and understand the method proposed in this document, we can now proceed with the description of the method itself.

E. Training ANNs by simulating quantum systems

In a previous section, it has been discussed how the problem of training a given ANN can actually be reduced to the problem of minimizing a corresponding error or cost function, which reads $E = E(y(\mathbf{x};\mathbf{w});(x^i,y^i))$, and where (x^i,y^i), for $i = 1,\dots,Ns$, represents the data set. In the case of the gradient descent method, for example, if the ANN has N hyper-parameters then a set of N corresponding updating rules or equivalently, evolution equations, must be solved such as the set of equations (4). Bearing in mind that these are essentially equations describing classical systems, one can logically observes that these equations could be replaced by a set of N Schrödinger equations, such as (7). In other words, they can be replaced by their quantum counterpart which can be solved numerically by applying, for instance, the FDTD method discussed in the previous section.

In practice, the set of N Schrödinger equations (in other words, the new update rules) reads:

$$\begin{aligned} i\hbar\frac{\partial \Psi_1}{\partial t} &= \big(-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial {x_1}^2} + U(\mathbf{x})\big)\Psi_1, \\ i\hbar\frac{\partial \Psi_2}{\partial t} &= \big(-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial {x_2}^2} + U(\mathbf{x})\big)\Psi_2, \\ &\dots \\ i\hbar\frac{\partial \Psi_N}{\partial t} &= \big(-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial {x_N}^2} + U(\mathbf{x})\big)\Psi_N. \end{aligned} \tag{10}$$

where $U = U(\mathbf{x}) = U(x_1,x_2,\dots,x_N)$. The system described by these equations can be interprated in physical terms as an ensemble of N quantum objects, e.g. electrons, which interact with each other through a given potential provided by the error function $U = U(x_1,x_2,\dots,x_N)$ (to be more precise the potential $U = U(\mathbf{x})$ should be multiplied by the elementary charge constant q). For completeness, it could alternatively be interpreted as a single N-dimensional quantum object (although we do not make use of this fact in this work).

The precise shape of the coupling term $U(\mathbf{x})$ still remains to be specified. In this work, the way to obtain a precise shape for it is provided by tailoring well-known techniques coming from DFT for the training problem at hand. In practice, one starts from the main problem of DFT, i.e. the simulation of the many-body Schrödinger equation which represents a daunting problem (even when approached by numerical techniques). The issues related to this problem are typically approached by introducing the so-called Kohn-Sham system which, in practice, consists of a system of N single-body Schrödinger equations which are coupled with each other by an exchange-correlation potential term[14], introduced as a mathematical approximation of the original problem [9]. Unfortunately, the exact mathematical shape of this newly introduced potential is unknown in practice, no general theoretical guidelines can be provided, and many different approximations are possible (with pros and cons). The Reader is suggested to read [16] to have a thorough view on the state of the art and the reasonings behind. The only prescription provided by DFT is that exchange-correlation potentials can be made dependent on the total density of the system only (rather than, for instance, the full set of wave functions) which allows to depict physically and chemically meaningful exchange-correlation terms. More specifically, in this work, a new coupling potential is proposed but with a different goal in mind (ML) compared to the main goal of DFT (Chemistry): in practice, we are not interested in obtaining very accurate simulations of chemical systems; we are simply interested in obtaining, in some efficient way, a good enough approximation of the quantum states so to minimize the given error function (in other words, the accuracy of this method is focusing on training an ANN, not on computing accurate chemical quantities which can

be computationally demanding). Therefore, one can depict simpler exchange-correlation potentials in this novel task.

In practice, for a Kohn-Sham set of (Schrödinger) equations issued from chemistry, the coupling for the i-th equation describing the i-th body, or electron, of the physical system has the following mathematical shape [9], [10], [14]:

$$V(x_i) = V_{ext}(x_i) + V_{xc}[\rho](x_i), \tag{11}$$

where the exchange-correlation potential depends on only one variable at a time, therefore completely avoiding the problem of simulating the many-body Schrödinger equation, and where $V_{ext} = V_{ext}(x)$ is a given external potential. Thus, in our specific case, we set $V_{ext}(x) = 0$ everywhere and, then, introduce the potential below:

$$U(x_i) = U(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_{i-1}, x_i, \bar{x}_{i+1}, \ldots, \bar{x}_N), \tag{12}$$

where the symbols $\bar{x}_i$, for $i = 1, \ldots, N$, represent the average position of the i-th body, or electron, according to its wave-function, i.e., in mathematical terms:

$$\bar{x}_i = \int_0^{L_X} x \Psi_i^2(x) dx \tag{13}$$

with L_X being the length of the (one-dimensional) spatial domain. The reason for such simple choice is quite intuitive. One observes that the potential being applied to this system essentially consists of one-dimensional quantum wells. As such, the one-dimensional quantum body is expected to fall into, or to be at least localized around, the bottom of those wells and, consequently, reducing the total energy of the system (thus providing the desired minimization scheme), until the final condition $U(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_N) = 0$ is met. Obviously, the potential suggested in (12) is of use for training ANNs only and would not be of any use if applied to obtain accurate simulations of actual chemical/physical systems; this is certainly acceptable since, once again, the purpose of this work is only to train ANNs (more realistic exchange-correlation potentials might happen to provide other advantages, but this is presently out of the scope of this work). Finally, it should be noted that the system of Kohn-Sham equations (10) coupled by means of the term (12) provides a simulation problem which can be tackled by a digital computing machine. In fact, every equation represents a one-dimensional quantum system which is known to be easy to simulate on even common personal computers [15].

It is now possible to list the steps to perform the training of a given ANN within the context of the tools presented so far; for every step, comments are added for the sake of clarity and completeness:

- An artificial neural network is specified along with a dataset and its corresponding error function, with N hyper-parameters to be computed to reduce the error (or, equivalently, the energy of the system). These N hyper-parameters are represented by the N average positions of the N wave functions described in the steps below.
- A one-dimensional spatial domain is specified by the user by means of its length L_X. Other hyper-parameters are defined as well, more specifically the number of spatial cells N_X, a time step Δt, along with a maximum number of steps to run ITMAX. Finally, a value RMAX is specified which represents the numerical range in which to look for the ANN weights and biases, i.e. [-RMAX, +RMAX].
- N wave-functions Ψ_i are prepared in the same initial conditions corresponding to a quantum object which average position is equal to $L_X/2$. In practice, this is a Gaussian wave function with mean value in $x = L_X/2$ and dispersion $\sigma = L_X/5$. The boundary conditions for such wave functions are 0 at $x = 0$ and $x = L_X$. This applies during the whole simulation of such quantum objects.
- Then a main loop over the number of time steps ITMAX is started. In this loop, several operations are performed, among which: *a)* the computation of the current average position for every wave function of the system, *b)* the computation of the applied potential for every wave function of the system, and *c)* the evolution of every wave function by means of the FDTD method. During this time dependent evolution, the total energy of the system, represented by the quantity $U = U(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_N)$, can increase or decrease (since the exploration of the solution space is quantum or, in other words, ergodic). Thus, the best solution found, corresponding to the average position $(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_N)$ which minimally reduce the potential (or, equivalently, the error function) U, is recorded for future use. It is important to note that, during this simulation, tunnelling effects will facilitate the search for the best solution (just like it would happen, for instance, in the simulated quantum annealing approach).
- The final best solution is then utilized to set the weights and biases of the given ANN and to perform inferences.

In the next section we present a number of validation tests which have been performed to validate the QiML approach along with a discussion on the results obtained.

V. Numerical validation

In order to numerically validate the approach suggested in these pages, several validation tests have been performed. Some of them are quite preliminary although very compelling. The tests selected are: *1)* the curve fitting problem, *2)* the reduced MNIST test, *3)* the standard MNIST test and, finally, *4)* a real life use case coming from the field of telecommunications. The reason for such selection is actually quite simple: *1)* the curve fitting is a quintessential test in ML which, in spite of its simplicity, embed the tenets of a proper ML problem [4]; *2)* the reduced MNIST provides a way to test generalization capabilities of training methods; *3)* the MNIST and CIFAR-10 data sets are considered standard tests to assess a training method and/or a ML model; and *4)* the telecommunication data set because it is constituted of realistic measurements. All these problems comes from both families of regression and classification problems.

For every test shown in this section, the activation function for the neurons in the hidden layers is represented by the hyperbolic tangent while the output activation function is either the softmax function or the identity function, depending

on if the problem involves classifications or regressions. The hyperparameters for the simulation of the quantum systems are $L_X = 10$ nm (total length of the physical one-dimensional domain), $N_X = 5000$ (where N_X is the number of cells for the spatial discretization), $\Delta_T = 0.1$ femtoseconds (simulation time step), $\epsilon_Q = 1.e-3$ (the simulation stops when the energy is below this value), and $R_{max} = 2.75$ (all weights are in the range $]-R_{max}; +R_{max}[$). When applicable, every network has been trained in (random) batches with 128 samples per batch. The architectures utilized for the various tests performed are described in the subsequent sections.

A. The first test: curve fitting

This first test focuses on the curve fitting problem which represents a quintessential problem of ML [4]. In fact, many of the relevant issues concerning the applications of ANNs can be introduced in this very simple context. In practice, this problem consists in finding the parameters (weights and biases) of an ANN such that it fits a function described through a finite set of points.

In more details, two different kinds of neural networks have been trained to fit two different functions, i.e., $f(x) = x^2$ and $f(x) = \sqrt(x)$, for x equal to 0.15, 0.60, and 0.80. These are used as the training set while the solution found is validated for the values $x = 0.30$ and $x = 0.70$. No pre-processing transform, such as normalization, etc., is applied. First, a mainstream ANN with one hidden layer is trained on these two sets of points (one set at a time). The network has one input neuron, 5 neurons in the hidden layer and, finally, one output neuron. All neurons are connected in a fully connected feedforward topology. The QiML method is then utilized for the training phase (which coincides with the fitting problem at hand). Furthermore, to show that QiML is agnostic to the model in use, a neural network made of spiking artificial neurons [17], or spiking neural network (SNN), is trained in exactly the same way on the same sets of points. Even in this case, the architecture is the same as before, i.e., one input neuron, 5 hidden neurons and one output neuron, with the only (important) difference that now neurons do not implement any activation function but, rather, they take a train of spikes as input and provide a train of spikes as output. The way these trains are computed is based on the well-known leaky-integrate-fire (LIF) model and any real number is encoded in a train of spikes through the frequency, or rate, of the spikes themselves [17].

In both cases, convergence using the QiML method is reached on the two different datasets. Furthermore, for the sake of comparisons, the same networks have been retrained with the gradient descent method on exactly the same data. The final learning curves are shown in Figs. 1 and 2. In more details, Fig. 1 shows the evolution of the ANN cost function w.r.t. time (i.e., in simulation steps or iterations, depending on the method), while Fig. 2 refers to the function $f(x) = \sqrt(x)$. Similar results are obtained for the SNN (not reported for a lack of space). From these plots, it is quite evident how the QiML method decreases more rapidly than the gradient descent method. This seems to be a characteristic

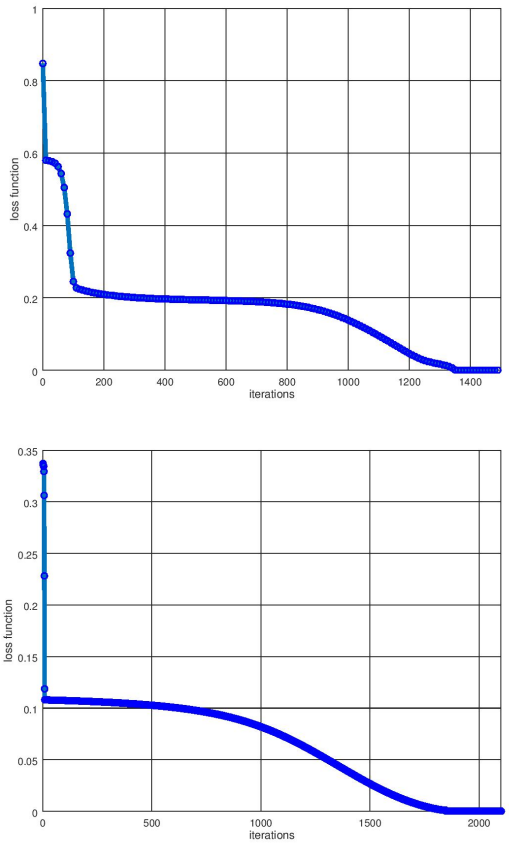


Fig. 1: Results for the curve fitting problem with ANNs for the function $f(x) = x^2$. Top: learning curve obtained using the gradient descent method. Bottom: learning curve obtained using the QiML method. Both: clearly, the two approaches explore the space of solutions in drastically different ways. Also, the QiML method seems to converge rapidly at the very beginning of the search.

of the method, since it has been consistently observed in other (different) tests as well. Finally, the Reader should note that the numerical experiments performed in this paper make use of multi-core CPUs only.

B. The second test: reduced MNIST

The purpose of this test, in reality a simple preliminary exploration, is to compare QiML and gradient descent in terms of generalization capabilities. It is based on the Modified National Institute of Standards and Technology (MNIST) data set which is a large database of handwritten digits. It is commonly used for training various image processing systems and also widely used for training and testing in ML. Specifically, it contains $60,000$ training images (resolution 28x28) and $10,000$ testing images [18].

Usually, in the field of ML, large and complex data sets are of high interest since real life applications rely on them. From a scientific perspective, though, small and simple data sets can be equally important since they provide a way towards training machines using only a small number of samples. In fact, it

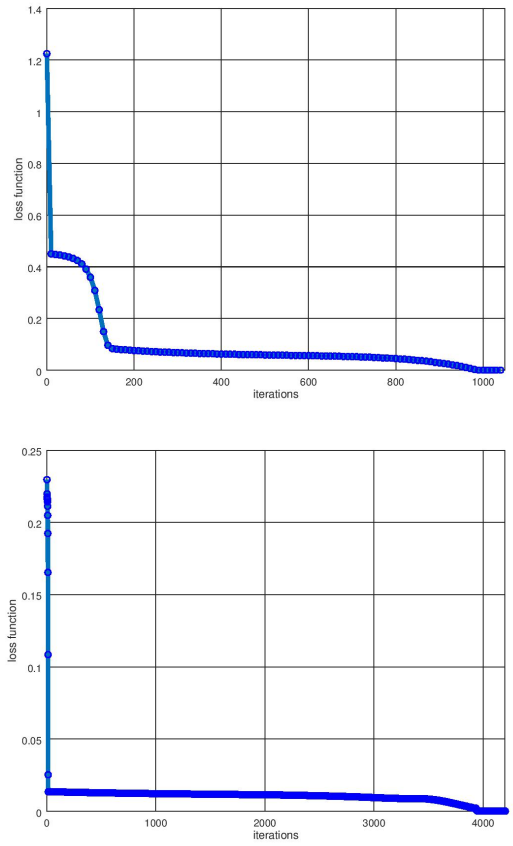


Fig. 2: Results for the curve fitting problem with ANNs for the function $f(x) = \sqrt{(x)}$. Top: learning curve obtained using the gradient descent method. Bottom: learning curve for the QiML method. Both: the two approaches, once again, explore the space of solutions in drastically different ways.

is well known that biological intelligence does not require to process the whole set of $60,000$ samples to recognize a handwritten digit. Therefore, tests with just a few samples can reveal both theoretical and applicative interesting insights. In particular, one trains an ANN using a tiny subset of the original MNIST data set, called reduced MNIST and denoted by RMNIST. The subset RMNIST/N, with N an integer number, is defined as the set which contains N samples of every digit. Thus, it contains $N \times 10$ elements in total. For instance, the set RMNIST/1 contains 1 sample per digit, i.e. 10 samples, the set RMNIST/4 contains 4 samples per digit, i.e. 40 samples, etc. Similarly, one can introduce the extended version of the subset RMNIST/N, indicated by MNIST/N-ext, and which contains the same elements of RMNIST/N plus a translation of every single element in the fourth direction (north, south, east and west) by 1 pixel. This allows a simple and natural extension of the original data set since a digit translated in one direction by one pixel is, obviously, still the same digit.

Both QiML and gradient descent are utilized for the training. In more details, the ANN consists of 784 input neurons (i.e., 28×28 pixels), 15 neurons in the hidden layer, with hyperbolic tangent, and 10 output neurons on which the

test	GD valid. err.	QiML valid. err.
RMNIST/1	0.59	0.47
RMNIST/4	0.36	0.29
RMNIST/1-ext	0.44	0.33
RMNIST/4-ext	0.19	0.15

TABLE I: Tests performed with reduced MNIST data sets RMNIST/1 and RMNIST/4 and their extended version as well. These results provide an indication that QiML might be able to generalize further than the gradient descent method.

architecture	GD + ReLU	QiML + SiLU
$784 - 300 - 10$	4.7	3.98
$784 - 1000 - 10$	3.8	3.06

TABLE II: Validation error for the standard MNIST test.

softmax function is applied. The training phase is very simple and consists of using the sets RMNIST/N and RMNIST/N-ext, with $N = 1$ and $N = 4$ until the best convergence is reached. The sets RMNIST/1, RMNIST/4, RMNIST/1-ext and RMNIST/4-ext are utilized for the training part while $10,000$ samples of the original MNIST data set are utilized for the validation part. The final results are shown in table I. Although one should consider this a very preliminary test, it seems to indicate that the QiML method could provide some advantages in terms of generalization capabilities.

C. The third test: standard MNIST

pre-processing: normalization, etc. architecture: ANN, number of hidden layers, etc.

Talk about convergence even before visiting a full epoch!!!!

The test described in this section is considered a de-facto standard in the field of ML to validate novel approaches. We now utilize the whole MNIST data set [18], described above, and train an ANN to recognize handwritten digits. As usual, the data is plit in two subsets, one for training purposes consisting of $50,000$ samples, and one for validation purposes and consisting of $10,000$ samples. Finally, the value for every pixel is rescaled so to be in the range $[0, 1]$. Two architectures for the network are tested which consists of *1)* 784 ($784 = 28 \times 28$) input neurons, 300 hidden neurons, 10 output neurons (for softmax operations), and *2)* 784 input neurons, 1000 hidden neurons and 10 output neurons respectively. The activation function in the hidden layer is the Sigmoid Linear Unit (SiLU) function which reads:

$$S(x) = \frac{x}{1 + e^{-x}}.$$

The outcomes of these numerical experiments are reported in table II and, for a fair comparison (the Reader should bear in mind that the QiML method is still in its infancy), the results published in [19] are reported as well for two networks with Rectified Linear Units (ReLU).

D. The fourth test: telecommunication use case

This final test involves the use of a real life data set coming from the field of telecommunications. Due to the nature of the data utilized, its specific details cannot be fully described

architecture	method	train. err.	valid. err.
$26-2-2-1$	GD	0.024	0.024
$26-2-2-1$	QiML	0.023	0.024
$26-2-2-2-1$	GD	0.022	0.023
$26-2-2-2-2-1$	QiML	0.023	0.023
$26-2-2-2-2-2-2-1$	GD	0.022	0.023
$26-2-2-2-2-2-2-1$	QiML	0.017	0.020

TABLE III

but it suffices to know that it consists of $420,000$ samples made of 26 features each. The goal is to predict one value corresponding to the 26 features at hand. The architectures tested consist of 26 input neurons, 2, 4 and 6 hidden layers (respectively) with 2 neurons each (hyperbolic tangent), and 1 output neuron. These tests have been performed with both QiML, in its infancy, and a gradient descent technique. The results, for comparison purposes, are reported in III where one clearly sees how the QiML is capable of providing good results although relatively recent.

VI. Conclusions and future works

The last decade has seen an increase of interest in the field of quantum ML. The decoherence-free approaches being explored today, such as QRAM, still face important technical issues and it is not clear when in the future these issues will be solved [3]. The QiML approach proposed in this work is certainly still in its infancy but it is already showing that it is possible to train ANNs by means of simple digital simulations of a certain class of quantum systems (i.e., electrostatically coupled one-dimensional electrons), as shown in the tests presented above. Certainly this approach can (and will) be improved in many different ways; in this article we have restricted ourselves to a proof of concept. For instance, some improvement could be obtained by:

- introducing LDA-type of exchange-correlation terms,
- introduction different techniques in the simulation of the Schrödinger equation,

and this will be the subject of further works. Once again, the choices made in this document are not limitations but they have been selected only for the purpose of simplifying the discussion.

Although more tests are certainly required (and will be performed) in the future, the following conclusions can be already drawn:

- During the various numerical experiments performed, it has been observed that some sort of mechanism for the ergodic exploration of the space of hyper-parameters seems to be taking place. In particular, the results obtained show that the final result does not depend on the initial guesses of the weights. Clearly, this is of extreme importance for ML practictioners since, in this way, one completely avoids the (time and resource consuming) typical cycle of trial-and-errors to find the best trained ANN, which, for classical methods, depends on the initial guesses.
- It can be easily shown that the QiML method scales linearly with the complexity of the problem at hand. In fact, it is an embarassingly parallelizable approach (various levels of linear parallelization schemes can be easily introduced in the system (10)). This can be of relevance for big models which, necessarily, require parallel computing devices.
- The QiML approach is model agnostic. In fact, we have been able to train ML models beyond the usual ANNs. For instance, it is possible to train spiking neural networks which can have recurrent connections. This could be of extreme importance for future models which could use different types of artificial neurons.

The author firmly believes that this work can pave the way towards a different and, possibly, better kind of machine learning approach.

References

[1] G. E. Moore, "Cramming more components onto integrated circuits", intel.com, Electronics Magazine, 19 April, (1965).
[2] C. Ciliberto et al., "Quantum machine learning: a classical perspective", Proc. R. Soc. A 474, (2018).
[3] M. Dyakonov, "The case against quantum computing", IEEE Spectrum, 15 Nov., (2018).
[4] C.M. Bishop, "Neural networks for pattern recognition", Oxford University Press, (1995).
[5] M.A. Nielsen, I.L. Chang, "Quantum computation and quantum information", Cambridge University Press, (2010).
[6] L. Buffoni, F. Caruso, "New trends in quantum machine learning", Europhysics letters, Vol. 132, Num. 6, (2021).
[7] V. Giovannetti, S. Lloyd, L. Maccone, "Quantum random access memory", Phys. Rev. Lett. 100, 160501, (2008).
[8] A.W. Harrow, A. Hassidim, S. Lloyd, "Quantum algorithm for linear systems of equations", Phys. Rev. Lett. 103, 150502, (2009).
[9] R.M. Martin, "Electronic structure", Cambridge University Press, (2013).
[10] M.A.L. Marques et al., "Fundamentals of time-dependent density functional theory", Springer, (2012).
[11] J. Biamonte et al., "Quantum machine learning", Nature, 549, pp. 195-202, September (2017).
[12] A. Messiah, "Quantum mechanics", Dover Publications, Vols. 1 and 2, (2014).
[13] P. Hohenberg, W. Kohn, "Inhomogeneous electron gas", Phys. Rev. 136, B864, November (1964).
[14] W. Kohn, L.J. Sham, "Self-consistent equations including exchange and correlation effects", Phys. Rev. 140, A1133, November (1965).
[15] N. Bigaouette et al., "Nonlinear grid mapping applied to FDTD-based, multi-center 3D Schrödinger equation solver", Comp. Phys. Comm., 183, pp. 38-45, (2012).
[16] N. Mardirossian, M. Head-Gordon, "Thirty years of density functional theory in computational chemistry: an overview and extensive assessment of 200 density functionals", Molecular Physics, Vol. 115, No. 19, pp. 2315-2372, (2017).
[17] W. Gerstner et al., "Neuronal dynamics: from single neurons to networks and models of cognition", Cambridge University Press, (2014).
[18] http://yann.lecun.com/exdb/mnist/, last visited Aug 2023.
[19] Y. Lecun et al., "Gradient-based learning applied to document recognition", Proceeding of the IEEE, Vol. 86, pp. 2278-2324, (1998).