UNIVERSITAT DE BARCELONA

UNIVERSITAT DE BARCELONA

FACULTAT DE FÍSICA

Applications of quantum machine learning in cybersecurity

MASTER THESIS

Author:
Álvaro Ari Huanay de Dios

Supervisor:
Artur Garcia Saez

Adversarial attacks on discriminative algorithms are highly used in the field of cybersecurity (e.g. on email-filtering or malware bypassing). As the automation of tasks is since now used more than ever, all current state-of the art attack and defence methods are frequently exploited by some sort machine learning action. This project aims to build a pipeline to bypass a discriminator (ResNet-18) using a Probabilistic Generative Model (Restricted Boltzmann Machine) by generating images as seemingly as possible to real hand-written numbers (MNIST dataset) trained by two different methods; classical machine learning and quantum-enhanced machine learning (classical machine learning with a final boost of quantum annealing) using D-Wave´s quantum computers. Quantum computing is proposed as an alternative to minimize more the free energy of the model at the end of the training. Computation of the loss function is proven to be easier with a quantum annealing machine rather than with fully classical methods. A better accuracy is expected by the quantum-enhanced model as well as a faster training. Quality of the images generated by each technique is compared and possible applications in the field of cybersecurity using PGMs are proposed besides of discussing physical requirements.

18th July 2022

Acknowledgements

Many thanks to the QUANTIC team
for their inconditional support and
helping me to see things from other
points of view in moments of uncertainty.

Also special thanks to my supervisor,
for opening me the door to the
exciting field of quantum computing
applied to cybersecurity.

Contents

1 Introduction and motivation

Computer science started to be part of society´s daily basis since the early 1960s. Its great utility in applications such as information storage, automation of tasks, and lately the creation of the internet to produce, share, and communicate content were the triggers that strengthened a rapid merge between humanity and this new discipline. However, society's accelerated dependence on computer systems puts people's safety at risk. Since 1970s-1980s they began to appear the first trojans and worms forcing cybersecurity to be taken more seriously. As time goes by, cyber threats become more sophisticated and difficult to identify. With the rise of machine learning new counter measures were created and slowed down the frenetic advance of cyber threats for a short period of time before attackers started to learn ML techniques as well and mastered how to exploit current defense models. These attackers realized that almost all defence techniques are based on Neural Networks which are vulnerable to small local perturbations (1) . This project aims to provide a probabilistic generative[1] model (PGM) as a tool to bypass a classifier generating hand-written digits and subsequently to use it for future defence models since they are proven to be more robust against adversarial attacks on discriminative algorithms[2] (1), (7), (8). A Restricted Boltzmann Machine is trained classically and later compared with a hybrid training that consists of a classical training complemented with quantum-enhanced sampling using a quantum annealing (QA) machine by D-Wave. Each model is compared to determine which has a better performance in replicating the introduced dataset. Sampling methods such as Markov Chain Monte Carlo (MCMC) can have very long mixing times and quantum sampling could provide an alternative to conventional techniques for sampling to draw representative samples of a Boltzmann distribution (2). Finally, possible applications in the field of cybersecurity with this model are proposed and physical requirements discussed to build such defences boosted by quantum computers.
The pipeline of this paper consists of a Restricted Boltzmann Machine for generative purposes and a Residual Neural Network of 18 layers (ResNet-18) for the discriminative stage of the project to evaluate the efficiency of both RBMs reproducing the MNIST dataset[3].

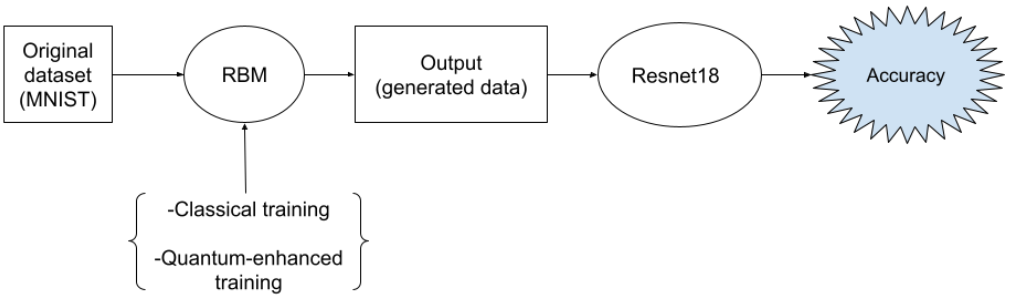


Figure 1: Pipeline of the simulation.

[1]Generative models include the probability distribution of the data itself $P(X, Y)$. A generative model can be used to generate random instances of an observation and target (x, y), or of an observation (x).

[2]Discriminative algorithms model a direct solution. For example, linear regressions create a decision boundary and take decisions based on it. Some examples are Artificial Neural Networks or Decision Trees.

[3]Currently MNIST is the only commonly image dataset used that can easily be implemented to a current quantum processing unit (QPU) (1). This is why the MNIST is used as input dataset for this project.

2 Restricted Boltzmann Machines

The pipeline of this project starts with a Restricted Boltzmann Machine. Restricted Boltzmann Machines consist of stochastic binary variables arranged into a visible layer and a hidden layer, where inter-layer connections are allowed but intra-layer connections are forbidden conforming an undirected bipartite graph (2).

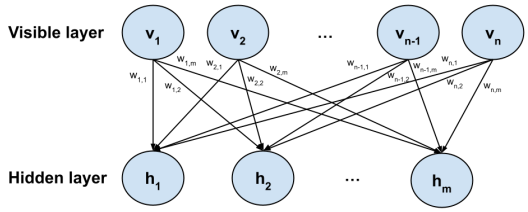


Figure 2: General representation of a Restricted Boltzmann Machine.

This restricted configuration allows for more efficient training algorithms in RBMs such as the gradient-based contrastive divergence (CD), persistent contrastive divergence (PCD), or parallel tempering (PT) algorithms. RBMs belong to the family of generative stochastic artificial neural networks that can learn a probability distribution over its set of inputs and reproduce it. RBMs can be used in many fields such as classification, feature learning, dimensionality reduction, or even in many body quantum mechanics. They can also be trained in either supervised or unsupervised ways. Its joint probability distribution is defined by a Gibbs distribution

$$P(v,h) = \frac{1}{Z} e^{-E(v,h)} \tag{1}$$

With an energy functional

$$E(v,h) = -\sum_{i=1}^{n} b_i v_i - \sum_{j=1}^{m} c_j h_j - \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} v_i h_j \qquad\qquad v_i h_j \in \{0,1\} \tag{2}$$

Where n (m) is the number of visible (hidden) nodes and v (h) represent the visible (hidden) nodes of the RBM. The terms w_{ij} b_i, c_j represent the weights, visible and hidden biases respectively that are represented by real values. The normalization constant of equation 1

$$Z = \sum_{\{v_k\}} \sum_{\{h_l\}} \exp\left(\sum_k b_k v_k + \sum_l c_l h_l + \sum_{kl} w_{kl} v_k h_l \right) \tag{3}$$

is known as the partition function of the system. Because of the bipartite graph structure, the forward and reverse conditional probability distributions for the RBM are both

$$P(h_j = 1|v) = \sigma\left(c_j + \sum_i w_{ij} v_i \right) \tag{4}$$

$$P(v_i = 1|h) = \sigma\left(b_i + \sum_i w_{ij} h_j \right) \tag{5}$$

Where σ stands for the sigmoid function and acts as an activation function which is used to add non-linearity in the machine learning model. These mathematical statements represent the building blocks of an RBM. The next step is to make the RBM learn so it can generate results as similar as possible to the data introduced (i.e., MNIST dataset).

2.1 Minimization of free energy in a RBM

The goal of the generative training is to determine the weights (w_{ij}) and biases (b_i, c_j) that minimize the negative log-likelihood (a.k.a loss function) of the model. PGMs can use different variants of gradient descent for training. These variants are used to compute a specific term deeply treated in this document; the negative phase (thermal equilibration) which is the reason why PGMs are robust against attacks enabled by local manipulations, because it involves global, long-range connections of the data learned (1). The ideal function to minimize is the Kullback-Leibler (KL) divergence which is a measurement of similarity between the probability distribution that the model learns ($p_\theta(x)$) and the probability distribution of the observed data ($p(x)$) to know how much information is lost when the model is being trained. The parameters that minimize the KL-divergence are the same as the parameters that minimize negative log-likelihood. This means that the loss function can be minimized and get the same parameters models (b_i, c_j, w_{ij}) that would have been gotten by minimizing the KL-divergence (52). The loss function of the model is expressed in the following form

$$L_\theta(T) = -\frac{1}{T}\sum_{x^{(i)}\in T} log\left(p_\theta(x^{(i)}\right) \tag{6}$$

where θ are the corresponding variations of the visible (b_i), hidden (c_j) biases and weights (w_{ij}) respectively and x corresponds to the input data distribution (MNIST dataset).
Coming back to equation 1, the probability distribution that the RBM learns is $p_\theta(x) = \frac{1}{Z_\theta}\sum_h \exp -E_\theta(x,h)$, being the partition function $Z_\theta = \sum_{x,h}\exp -E_\theta(x,h)$, and the free energy of a configuration of visible units

$$e^{-F_\theta(x)} = \sum_h e^{-E_\theta(x,h)} \tag{7}$$

Where $F_\theta(x)$ is defined as the free energy of the system.
With equations 1, 2, 3 and 7, the loss function defined at the beginning becomes

$$L_\theta(T) = \frac{1}{|T|}\sum_x F_\theta(x) + log\left(\sum_x e^{-F_\theta(x)}\right)$$

with its derivation demonstrated in appendix A. Thus, to minimize the loss function, it is needed to minimize its free energy as well. This last expression is more convenient to use to compute the gradient of the loss function. The first term is over the training dataset which is easily computed. However, the second term is over all possible neuron configurations and is intractable to compute classically. For N binary neurons the number of possible configurations is 2^N. Additionally, the gradient of the loss function can be obtained performing the chain rule. The complete derivation is in appendix B

$$\frac{\partial L_\theta}{\partial\theta} = \frac{1}{|T|}\sum_{x^{(i)}\in T}\frac{\partial F}{\partial\theta} - \sum_x p_\theta(x)\frac{\partial F_\theta(x)}{\partial\theta} = (\langle x_i\rangle_{data} - \langle x_i\rangle_{model})$$

The first term $\langle x_i\rangle_{data}$ is called the positive phase and can be efficiently computed from a classical computer using equation 4 because comes from the evaluations on the training set (2). Nonetheless, the second term $\langle x_i\rangle_{model}$ (negative phase), is the term that becomes intractable as the number of visible and hidden nodes increase because it requires knowledge of the overall distribution $p_\theta(x)$ (or analogously of the partition function Z_θ) (1). Note that the derivation of the gradient of the loss function has been done for a generic parameter θ and it is dependent on three variables, b_i, c_j, w_{ij}. Therefore, the actual gradients are

$$\frac{\partial L_{\{b_i,c_j,w_{ij}\}}}{\partial w_{ij}} = \langle v_i h_j\rangle_{data} - \langle v_i h_j\rangle_{model} \tag{8}$$

$$\frac{\partial L_{\{b_i,c_j,w_{ij}\}}}{\partial b_i} = \langle v_i \rangle_{data} - \langle v_i \rangle_{model} \tag{9}$$

$$\frac{\partial L_{\{b_i,c_j,w_{ij}\}}}{\partial c_j} = \langle h_j \rangle_{data} - \langle h_j \rangle_{model} \tag{10}$$

Being $\langle \cdot \rangle_{model}$ the difficult term to treat (classically)

$$\langle v_i h_j \rangle_{model} = \frac{1}{Z} \sum_{\{v_k\}} \sum_{\{h_l\}} v_i h_j \exp\left(\sum_k b_k v_k + \sum_k c_l h_l + \sum_{kl} w_{kl} v_k h_l \right) \tag{11}$$

And respectively for $\langle v_i \rangle_{model}$ and $\langle h_j \rangle_{model}$. As seen in Eq. 11, the exact value is intractable to compute classically if desired. Therefore, a sampling of the overall distribution is needed to make an approximation of the true value. There are two ways to solve this, one is by means of classical heuristics, and the other is by quantum annealing sampling.

2.2 Classical sampling

To evaluate the gradient of the loss function correctly (Eqs. 8-10) an approximation as good as possible of the negative phase is needed because computing $\langle \cdot \rangle_{model}$ exactly is not feasible. In this document, parallel tempering is explained as an alternative approach for the approximation of $\langle v_i \rangle_{model}$. The $\langle h_j \rangle_{model}$ is computed from Eq. 5 using the configuration already minimized by parallel tempering (configuration with lower free energy), and $\langle v_i h_j \rangle_{model}$ is obtained doing the outer product of these last two vectors $\langle v_i \rangle_{model} \times \langle h_j \rangle_{model}$, where each row i (j) represents a determined node in the RBM, and computing its average.

2.2.1 Parallel tempering (PT)

When complex potential energy surfaces are encountered (i.e., with high energy barriers), traditional metropolis MC can be inefficient blocking the model to compute a good value of the gradient of the loss function. As a proposition, an algorithm which has inspiration in the dynamics of physical processes is presented. It is called parallel tempering (PT).
Parallel tempering is a simulation method typically used in problems where sampling or optimization processes are necessary. In this situation, it is going to be used as a sampling tool. PT is used to find the lowest free energy state configuration of a system of many interacting particles (the nodes of the RBM) at low temperature. PT solves this problem by running several MC simulations (chains) in parallel at different temperatures (T_i), being $\beta_i = 1/k_B T_i$. The simulations at higher temperatures are able to explore the configuration space more freely, crossing energy barriers and hopping among shallow energy minima. PT takes advantage of this by exchanging these higher temperature configurations with lower temperature ones if a certain condition is fulfilled, the metropolis criterion.

$$P(C_i \leftrightarrow C_j) = min\left(1, \exp\left[\left(\frac{1}{k_B T_j} - \frac{1}{k_B T_i}\right)[H(C_j) - H(C_i)]\right]\right) \tag{12}$$

Being $H(C_i)$ and $H(C_j)$ the energies for the chains with configurations C_i and C_j respectively, allowing the low-temperature simulation to sample configurations more efficiently than with local Metropolis only (43). After a swap move is applied, replica at β_i assumes configuration C_j with energy $\hat{H}(C_j)$ and the replica β_j gets the configuration C_i with energy $\hat{H}(C_i)$ being able the replica with lowest energy to sample better the energy landscape. This process is applied for 100 chains at the same time.

2.3 Quantum sampling

RBMs are slow to train and time-consuming due to the slow mixing of Gibbs sampling (2). Moreover, PT can stay at a local minima or not to sample the configuration with lowest energy state (the swaps depend on probabilities, see Eq. 12). Quantum annealing processes can serve as an aid to explore more efficiently the energy landscape thanks to quantum superposition and tunneling training the RBM more effectively which can lead to a better calculation of the gradients and thus to a better accuracy of the model.

2.3.1 Quantum annealing (QA)

Quantum annealing is an optimization process to find the global minimum of a given objective function (the Hamiltonian) by using quantum fluctuations. It is used mainly for problems where the search space is discrete with many local minima (55). Sampling problems are related to these optimization problems by sampling from many low energy states to try to characterize the shape of the energy landscape spectrum, i.e., the free energy of the RBM. This sampling is used to obtain the negative phases $\langle \cdot \rangle_{model}$ of Eqs. 8-10.

The process of quantum annealing can be explained at its fundamental with 1 single qubit. This qubit is in a superposition of states 0 and 1. After the process of annealing this qubit collapses to either one of these 2 states. This process can be understood with the following diagram seen in Fig 3

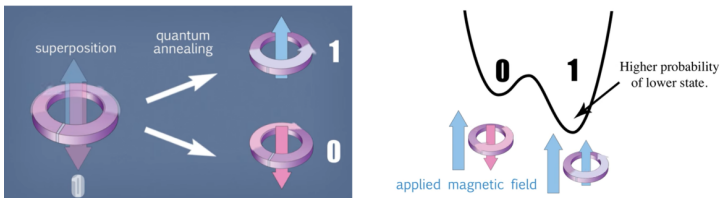


Figure 3: 1 qubit system collapsing to the ground state (55).

An energy barrier is raised in the middle to create a double well potential. The system can be controlled by introducing a bias (i.e., an external magnetic field) such that the probabilities of each state are not the same. This is done by minimizing more the energy of one state increasing the probability to collapse to this lower state due to the principle of minimum energy. For example, making more probable to collapse towards state $|1\rangle$ rather than $|0\rangle$ as seen in Fig 3.

To create a quantum computer that works by quantum annealing it is needed a high number of qubits and a way to make them work together. This is achieved by using an additional element, couplers. Couplers can make 2 independent qubits end up having the same final state or the opposite. The physical interpretation of couplers is the entanglement allowing to link 2 qubits at the same time. When two qubits are entangled, they can be thought of as a single object with four possible states ($|0,0\rangle$, $|1,0\rangle$, $|0,1\rangle$, $|1,1\rangle$). The relative energy of each state ($E_{|q_1,q_2\rangle}$) depends on the biases and couplings between qubits q_1 and q_2. For example, there could be a configuration where the state $|1,1\rangle$ is the one with lowest energy. The biases and couplings define the energy landscape, and the D-Wave quantum computer finds the minimum energy of that landscape (i.e., $E_{|1,1\rangle}$). The system gets more complex as qubits are added. For example, three qubits have eight possible states over which to define an energy landscape; four qubits have sixteen, five qubits thirty two and so on following the rule 2^N being N the number of qubits.

Therefore, as each qubit can have a bias applied and can interact with other qubit by means of couplers, one can build a system that reproduces the final Hamiltonian ($\hat{H}_f$) by linking several independent qubits together. Then, the quantum annealer solves the problem by finding the state with minimum energy moving the system from a fully quantum state towards a completly classical system (the system collapses). All of this happens in D-Wave´s chips around $20\mu s$.

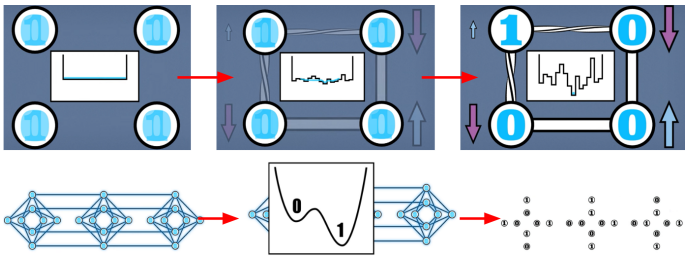


Figure 4: System collapsing after annealing process (55).

The Hamiltonian of the quantum annealing process is described by the following equation.

$$H_{ising} = -\frac{A(s)}{2}\left(\sum_i \hat{\sigma}_x^{(i)}\right) + \frac{B(s)}{2}\left(\sum_i h_i \hat{\sigma}_z^{(i)} + \sum_{i>j} J_{i,j}\hat{\sigma}_z^{(i)}\hat{\sigma}_z^{(j)}\right) \tag{13}$$

Being $\hat{\sigma}_{x,z}^{(i)}$ the Pauli matrices that operate on a qubit $q_{i/j}$ and h_i, $J_{i,j}$ are the qubit biases and coupling strengths respectively. The term $s = t/t_f$ is a factor that equals 1 when time t reaches t_f (total annealing time). The Hamiltonian can be decomposed in two terms; the initial Hamiltonian (first term, H_i) and the final Hamiltonian (second term, H_f). Final Hamiltonian contains the qubits, biases and couplers that have correspondence with the nodes, biases and weights of the RBM under study. The lowest energy state of the final Hamiltonian is the answer to the problem under study.

In quantum annealing the system starts at $t = 0$, being $A(0) \gg B(0)$ and at the lowest energy eigenstate of H_i. As it anneals, it introduces the H_f slowly enough to satisfy the conditions of the adiabatic theorem and the initial Hamiltonian is slowly reduced until the Hamiltonian contains only the $B(s)$ term. Ideally, the ground state of the initial Hamiltonian has been maintained through the annealing process such that at the end of the task the system collapses at the ground state of the problem Hamiltonian. In reality, the probability of staying in the ground state can sometimes be small due to perturbation effects such as thermal fluctuations from external sources or accelerating too much the annealing process. However, the low energy states returned still are useful. The most difficult problems to solve in terms of quantum annealing are those with the smallest minimum gaps[4] (29). The use of D-Wave´s annealer for the training of the RBM is as follows:

1. The RBM energy functional 2 is used as the final Hamiltonian $\hat{H}_f$.
2. Quantum annealing is run N times and the sample averages are taken for each negative phase respectively ($\langle v_i \rangle_{model}$), ($\langle h_j \rangle_{model}$) and ($\langle v_i h_j \rangle_{model}$) (2).
3. The negative phases $\langle \cdot \rangle_{model}$ are introduced into the gradient of the loss function (Eqs. 8-10) for each iteration during the training.

[4]Minimum gap is the minimum distance between the ground state and the first excited state of the system as the annealing process is undertaken.

2.4 Embedding the RBM into the Quantum Processor Unit (QPU)

To proceed to the sampling in the D-Wave computer, the concept of Q matrix is needed first. If x is defined as the binary vector that concatenates the visible node vector and the hidden node vector, the energy equation 2 can be expressed in the following form (2).

$$E = \beta x^T Q x \tag{14}$$

Where Q is the $(n+m) \times (n+m)$ matrix:

$$Q = \frac{1}{\beta_{eff}} \begin{pmatrix} B & W \\ 0 & C \end{pmatrix} \tag{15}$$

being n (m) the number of visible (hidden) nodes. B and C are diagonal matrices containing the biases b_i and c_j respectively and W represents the $n \times m$ squared matrix with all the elements corresponding to the weights between the visible and hidden nodes w_{ij}. The scale factor β_{eff} is set to 1.5 as a result of several studies performed in (2). Moreover, it is more convinient to work with "spin" variables rather than "binary" variables in D-Wave machines. Therefore, the transformation $x \rightarrow S = 2x - 1$ is applied and the Boltzmann distribution (Eq. 2) of the RBM becomes

$$E' = -\sum_{i=1}^{r} H_i S_i - \sum_{j=r+1}^{r+m} H_j S_j - \sum_{i=1}^{n} \sum_{j=r+1}^{r+m} J_{ij} S_i S_j \tag{16}$$

Where the visible nodes of the RBM are represented in this equation by the first r spin variables of the first term, the hidden nodes are represented by the next m spin variables of the second term, and the weights connection between the hidden and visible nodes are represented by the third term. Next, the RBM transformed to Ising model is embedded onto the D-Wave chip by mapping each visible node to a chain of vertical qubits (red dots), and each hidden node to a chain of horizontal qubits as seen in Figure 5.

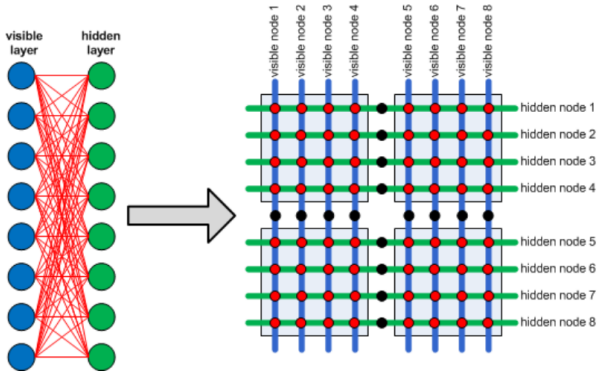


Figure 5: Embedding of a RBM onto a D-Wave QPU chimera graph (2).

Once the RBM is embedded on the QPU and sampling is done by the quantum annealer, the averages are computed and introduced into the gradients of the loss function.

- **Visible negative phase** $\langle v \rangle_{model}$. Where $\langle v_i \rangle_{model} = \frac{1}{N} \sum_{n=0}^{N-1} v_i^n$ being the dimension of the vector

$$\begin{pmatrix} \langle v_0 \rangle_{model} \\ \langle v_1 \rangle_{model} \\ \vdots \\ \langle v_r \rangle_{model} \end{pmatrix} \tag{17}$$

- **Hidden negative phase** $\langle h_j \rangle_{model}$. Where $\langle h_k \rangle_{model} = \frac{1}{N} \sum_{n=0}^{N-1} h_k^n$ being the dimension of the vector

$$\begin{pmatrix} \langle h_{r+1} \rangle_{model} \\ \langle h_{r+2} \rangle_{model} \\ \vdots \\ \langle h_{r+m} \rangle_{model} \end{pmatrix} \tag{18}$$

- **Weight negative phase** $\langle v_i h_j \rangle_{model}$. Where $\langle v_i h_j \rangle_{model} = \frac{1}{N} \sum_{n=0}^{N-1} v_i^n h_j^n$ with $i \in [0, r]$, $j \in [r+1, r+m]$ and being the dimension of the matrix

$$\begin{pmatrix} \langle v_0 h_{r+1} \rangle_{model} & \langle v_1 h_{r+1} \rangle_{model} & \cdots & \langle v_r h_{r+1} \rangle_{model} \\ \langle v_0 h_{r+2} \rangle_{model} & \langle v_1 h_{r+2} \rangle_{model} & \cdots & \langle v_r h_{r+2} \rangle_{model} \\ \langle v_0 h_{r+3} \rangle_{model} & \langle v_1 h_{r+3} \rangle_{model} & \cdots & \langle v_r h_{r+3} \rangle_{model} \\ \vdots & \vdots & \ddots & \vdots \\ \langle v_0 h_{r+m} \rangle_{model} & \langle v_1 h_{r+m} \rangle_{model} & \cdots & \langle v_r h_{r+m} \rangle_{model} \end{pmatrix} \tag{19}$$

Once these negative phases are computed, the gradient of the loss function can be easily obtained (Eqs. 8-10). The next step in the training is the optimization of the RBM to compute the new b_i, c_j, and w_{ij} updates.

2.5 Optimization of the RBM

To successfully make the RBM learn (i.e., minimize its free energy) it is also necessary to introduce an optimization algorithm that computes the new updates b_i, c_j, w_{ij} after the previous estimation the gradient of the loss function to update the configuration of the model. Adam optimizer (a.k.a. Adaptive moment estimation) algorithm which is an extension of Stochastic Gradient Descent (SGD) is used for this task. Some benefits of this model are that it requires little memory and it works very well for problems with large terms of data (48). This method takes momentum[5] and RMSprop[6] and makes them to work together. To implement Adam it is needed first to set at $t = 0$, $V_{dw} = V_{db} = V_{dh} = 0$, $S_{dw} = S_{db} = S_{dh} = 0$, and then on iteration t compute these values again using dw, dv, dh which are $\frac{\partial L}{\partial w_{ij}}$, $\frac{\partial L}{\partial b_i}$, $\frac{\partial L}{\partial h_j}$ respectively (previously computed in Eqs. 8-10 with parallel tempering or quantum annealing respectively) with the following formulas

$$V_{dw} = \beta_1 V_{dw} + (1 - \beta_1) dw; \qquad V_{db} = \beta_1 V_{db} + (1 - \beta_1) db; \qquad V_{dh} = \beta_1 V_{dh} + (1 - \beta_1) dh$$

This set of three equations is the momentum-like update with hyper parameter β_1 (momentum term). And the following three equations are

$$S_{dw} = \beta_2 S_{dw} + (1 - \beta_2) dw^2; \qquad S_{db} = \beta_2 S_{db} + (1 - \beta_2) db^2; \qquad S_{dh} = \beta_2 S_{dh} + (1 - \beta_2) dh^2$$

which represent the RMSprop-like update with hyper parameter β_2 (RMSprop term). Furthermore, in a typical implementation of Adam, bias corrections are needed. Therefore, to compute the real (corrected) values, the following formulas are used

$$V_{dw}^{corrected} = \frac{V_{dw}}{(1 - \beta_1^t)}; \qquad V_{db}^{corrected} = \frac{V_{db}}{(1 - \beta_1^t)}; \qquad V_{dh}^{corrected} = \frac{V_{dh}}{(1 - \beta_1^t)}$$

[5] Momentum is an extension to the gradient descent optimization algorithm that allows the search of a minima within an inertia in a determined direction (14)

[6] Root Mean Squared Propagation is an extension of gradient descent that uses a decaying average of partial gradients (15)

$$S_{dw}^{corrected} = \frac{S_{dw}}{(1-\beta_2^t)}; \qquad S_{db}^{corrected} = \frac{S_{db}}{(1-\beta_2^t)}; \qquad S_{dh}^{corrected} = \frac{S_{dh}}{(1-\beta_2^t)}$$

Now, after these corrections are computed, Adam optimization returns the RBM updates multiplied by the learning rate α and are added to the previous values w_{t-1}, b_{t-1}, h_{t-1}

$$w_t = w_{t-1} - \alpha \frac{V_{dw}^{corrected}}{\sqrt{S_{dw}^{corrected}} + \epsilon}; \qquad b_t = b_{t-1} - \alpha \frac{V_{db}^{corrected}}{\sqrt{S_{db}^{corrected}} + \epsilon}; \qquad h_t = h_{t-1} - \alpha \frac{V_{dh}^{corrected}}{\sqrt{S_{dh}^{corrected}} + \epsilon}$$

being w_t, b_t, h_t the actual updates at time t. The hyper-parameters used in this optimization process are $\alpha = 1 \cdot 10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \cdot 10^{-8}$. At this point, one optimization batch[7] is finished. To complete 1 training epoch, sampling, computation of gradients and updates must be repeated 1875 times. For a full understanding of the code and the functionality of the RBM it can be viewed in the GitHub repository (59). The implementation of the QA sampling process can be observed in the following Figure.

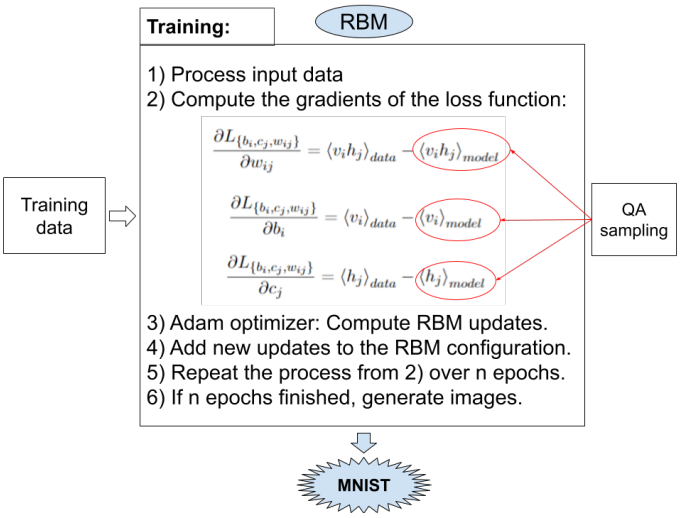


Figure 6: Pipeline of the generative training for a RBM with QA.

3 ResNet-18

The training of the generative model is finished. Now it is the moment to evaluate the performance of the RBM and it is done with the ResNet-18. The term "ResNet-18" stands for Residual Neural Network of 18 layers deep and it is mainly used for image recognition. A brief introduction is given in this paper as the goal of this project is to understand and focus on the RBM trained by two different methods and to benchmark its possible applications in cybersecurity.

Typical ResNet models are implemented with double or triple layer skips that contain ReLUs[8] and batch normalization in between (45). When dealing with problems of image recognition, researchers observed that after some depth in the network, the performance degrades. This problem is known as the vanishing gradient problem and can easily be explained by the chain rule. As the number of layers in the network increase, the number of elements that multiply the gradient of the loss do as well being each element usually less than unity. Hence, the gradient of the network diminishes stopping the NN learning.

[7] Batches are the number of iterations to finish 1 epoch (1875 batches per epoch).

[8] Rectified Linear Unit. It is defined as the positive part of its argument, i.e., $f(x) = x^+ = max(0, x)$.

3.1 Architecture and theoretical background

The solution to this problem comes with the invention of a "skip connection" between layers. These connections skip some layer in the NN and uses the output of one layer to feed the input of the next layer ahead (instead of the adjacent one). An image of the fundamental building block of a ResNet can be observed in Fig. 7. By just using vector addition of the identity function, the gradient of the loss is multiplied by one and its value is maintained in the earlier layers. Thanks to this, the gradient in previous layers can be preserved. The skip connection (a.k.a. residual block) can be represented mathematically as follows taking into consideration Fig. 7

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial H}\frac{\partial H}{\partial x} = \frac{\partial L}{\partial H}(\frac{\partial F}{\partial x} + 1) = \frac{\partial L}{\partial H}\frac{\partial F}{\partial x} + \frac{\partial L}{\partial H}$$

It can be seen that the first term vanishes before than the second one. Concatenative skip connections enable an alternative way to ensure feature reusability from the earlier layers to stabilize the training and converge the model (37).

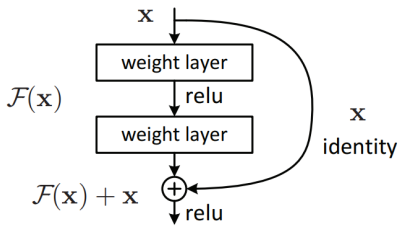


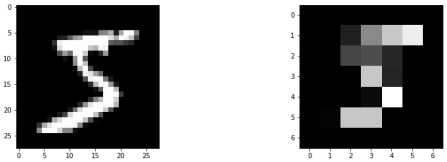
Figure 7: Canonical form of a ResNet (35).

4 Experimental setup

The main building blocks are ready, a RBM as a generative model and the ResNet-18 as a discriminative classifier. The generative model is trained by two different methods, classical training (CRBM) and hybrid training (HRBM), being the last method a classical training with a final boost of QA. Then, the generated images are evaluated by the ResNet-18, previously trained with MNIST as well. The natural datasize of MNIST is 28×28 pixels per image. The images are described by tensors whose elements are floats between $[0, 1]$. 1 means a white pixel and 0 a black pixel. The input datasize of the CRBM starts at 784 visible nodes ($28 \times 28 = 784$) for the visible layer and 300 hidden nodes (as hyper-parameter) for the hidden layer (see Fig. 2). The learning rate is the same as in section 2.5 and the number of epochs is still to be discussed in the results. Accuracies, performance and image quality are compared based on the two training methods.

4.1 Input dataset

The dataset used by the RBM is of two different sizes. One training dataset of size 28×28 to understand how the RBM works with full information, and another of size 7×7 (reduced) because the QPU of D-Wave´s quantum annealing machine cannot process images of bigger size. To evaluate correctly the accuracy of the RBM generating hand-written numbers, the ResNet has been previously trained with the MNIST dataset with sizes 28×28 and 7×7 returning an accuracy of 96.78% and 93.27% respectively. Therefore, errors that can be found in the evaluation are assigned explicitly to the RBM. An example of the images used to train the RBM and ResNet-18 can be observed in Fig. 8

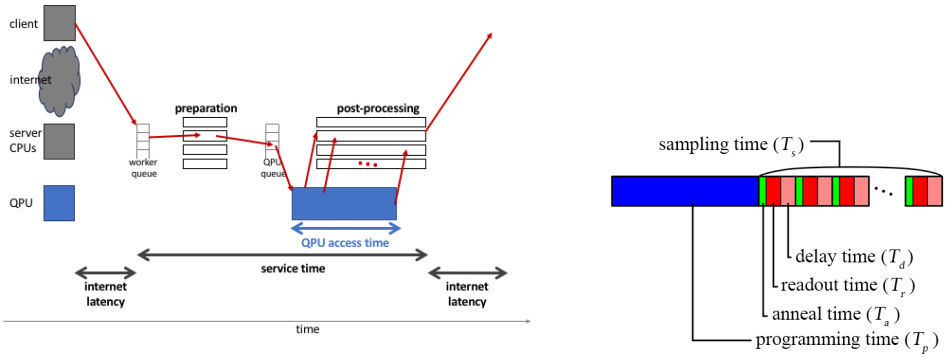


(a) Example 28×28. (b) Example 7×7.

Figure 8: MNIST image example of number 5.

4.2 Processing time of D-Wave´s systems

Before running all simulations, it is interesting to know how long does it take for the entire QA sampling process receive, compute, and return back the information. Figures 9a and 9b make a clear description of how a Quantum Machine Instruction (QMI) evolves.



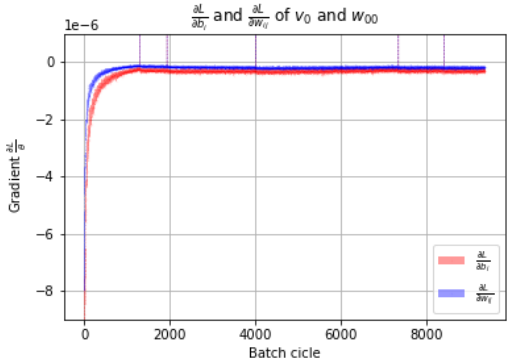
(a) D-Wave execution time for a single QMI. (b) QPU access time.

Figure 9: D-Wave processing time organization (31)

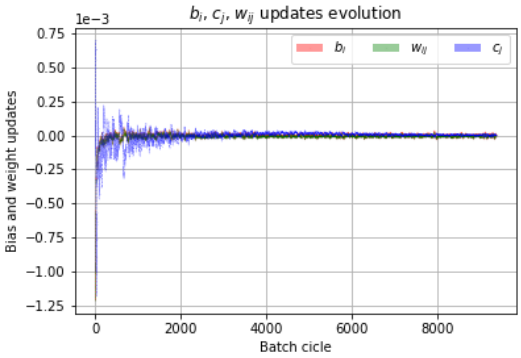
Fig. 9a describes how long does it take for a QMI to complete a single request. The first and last steps of the process consist of the internet latency of the client (that depending on the internet connection it is faster or slower). Next stage is the service time that consists of several steps starting at the worker queue in which all D-Wave clients wait in order to get their requests prepared for the QPU queue. The larger the queue, the larger the waiting time. Next comes the preparation step in which the QMI is prepared for the QPU queue to start the QPU access time. At this point, the process is better described in Fig. 9b. In the simulations run, 500 samplings were requested. The annealing time (green layer of Fig. 9b) takes around $20\mu s$ consuming a total of $0.01s$ for 1 single QMI request (without taking into consideration all the other previous steps such as delay time, readout time, programming time, internet lattency, queues...). Then, results from D-Wave are sent back to the client and the negative phases are computed locally with these sampled results. The negative phases are used to compute the gradients (Eqs. 8-10) and these are used to obtain the new bias and weight updates in the Adam optimizer step (see Fig. 6).
To summarize it briefly, the whole quantum-enhanced process builds the Q matrix, calls the sampler from D-Wave, sends the matrix to the sampler, obtains the free energy landscape of the RBM and returns results back to determine the $\langle \cdot \rangle_{model}$. This process is repeated iteratively and took an average of 9.87s. The complete process time range could be reduced by improving external circumstances such as greater internet connection (internet latency), a tinier worker queue size and with a better local computer speed.

5 Results

Before analyzing results, it must be taken into account that D-Wave cloud-based service provides only 1 minute of free service of total annealing simulations to anyone who wants to try their quantum computers. If that threshold of 1 minute is reached a fee must be paid in order to be able to continue with the simulation. To know how much time does a boost consume an example simulation of a HRBM of 50 h_{nodes} with 7×7 MNIST dataset and 5 training epochs has been run. The results showed that for a boosting training of 10.000 images the free trial consumed was of 68.11%. As 20 individual boosting simulations were run to obtain some statistical results, the boost had to be reduced to 1.000 images rather than 10.000 in order to be able to run them all. Results are provided in terms of plots (10, 11 and 12), tables (1 and 2) and images of handwritten numbers (13, 14 and 15).
To check if the RBM was working properly, plots of the gradient of the loss function (Fig. 10a) and of the bias and weight updates (Fig. 10b) were obtained for a CRBM. Both figures show that the evolution of the gradient and updates works pretty good for a low number of batches. These two plots are shown for just 5 training epochs because for 100 or 10 epochs the change of the gradient and updates would be almost imperceptible in an image. With 5 training epochs it can be seen that the most drastic change in values is accomplished after 2000 batch cicles ($\sim$ 1 epoch).

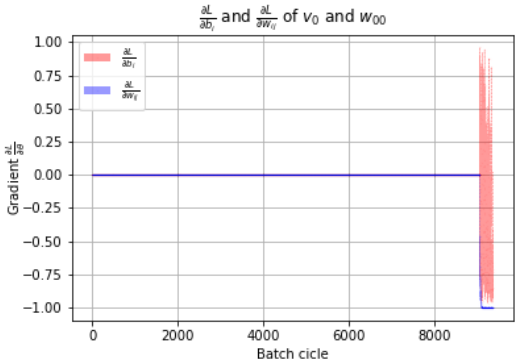


(a) $\frac{\partial L}{\partial b_i}$ and $\frac{\partial L}{\partial w_{ij}}$ of a CRBM with 5 epochs.

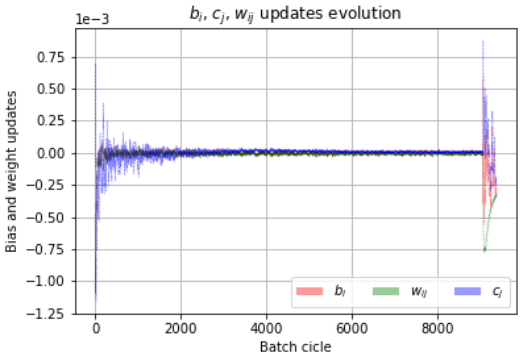


(b) b_i, c_j and w_{ij} updates per batch in a CRBM of 5 epochs.

Figure 10: CRBM evaluated for 5 epochs in the MNIST dataset 7×7 with 50 h_{nodes}.



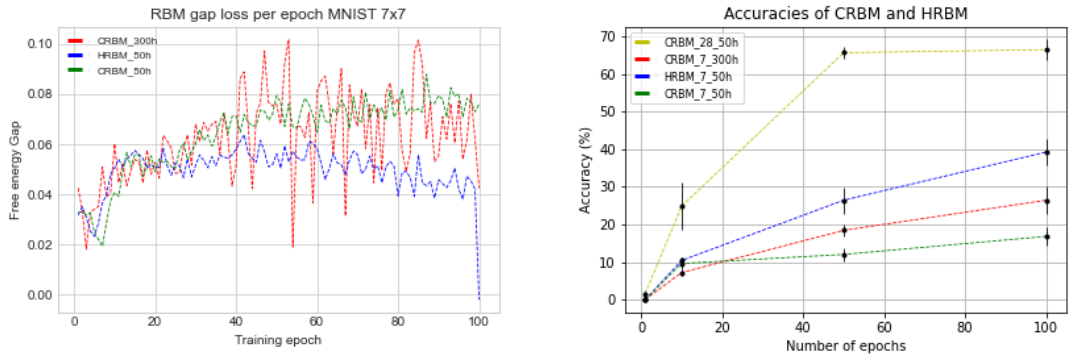
(a) $\frac{\partial L}{\partial b_i}$ and $\frac{\partial L}{\partial w_{ij}}$ of a HRBM with 5 epochs.



(b) b_i, c_j and w_{ij} updates per batch in a HRBM of 5 epochs.

Figure 11: HRBM evaluated for 5 epochs in the MNIST dataset 7×7 with 50 h_{nodes}.

Regarding Figures 11a and 11b it can be observed that the QA sampling boost makes a big step in the gradients and updates when it is started. This big step evidences that the annealer found a better minima in the energy spectrum of the RBM and hence changed the gradients and updates drastically. Therefore, the parallel tempering algorithm stuck in a local minima meaning that the energy spectrum of the model must have high barries which are more difficult to explore by parallel tempering than by QA. It can be noticed in both figures that once the annealing starts, a convergence towards zero is observed being in Figure 11b more noticeable than in 11a. Moreover, the complete annealing towards this aimed equilibrium is not possible to do because these last two plots correspond to the HRBM with the final boost of 10000 images consuming a 68.11% of the free trial. To observe a complete convergence in both Figures at least 5 minutes of annealing would be needed. Nonetheless, this abrupt change is of extreme interest because from both images it can be inferred that a considerably better configuration can be reached with a few samplings of the annealer. Hence with a few annealing samplings (1000 images) it can be enough to reach a better performance in the RBM (thus, lower free energy). Consequently, the following simulations are done with a final boost of 1000 images[9] rather than 10000.



(a) HRBM vs several CRBMs gap loss per epoch.

(b) HRBM vs several CRBMs accuracy.

Figure 12: Comparison of HRBM (blue) in terms of free energy difference and accuracy against CRBMs.

A good measure of well fitting is the free energy difference between some known (data) and unknown (validation data of the RBM) instances, i.e., $\Delta F(x) = F(x) - F_\theta(x)$. For that reason plots of Fig. 12a were created where many interesting things can be appreciated. Firstly, that in any case, gap between the known and unknown distributions is of order of a decimal in the worst case scenario (which is pretty good). Secondly, that the CRBM with 300 nodes (red) sometimes outperforms both models (green and blue) in free energy gap evaluation (manifesting the importance of a high h_{nodes}). Thirdly, observe that there is a correspondence in the gap loss value and the accuracy of the model in 7×7 datasets being the ones with lowest gap the ones with better accuracy. Fourthly (and most important), it can be observed clearly that at the end of the plot 12a the one with the lowest gap is the HRBM of 50 hidden nodes (blue). Hence, it is a very good indicator that with only 1000 annealing samplings in the final step of the training, the gap value reaches almost 0.
On the other hand, on Fig. 12b are shown the accuracies with its respective uncertainties. These values are obtained from Tables 1 and 2 in which several simulations were run in

[9] 1000 images for several reasons. Firstly, because otherwise there would not be enough time to run all the simulations (10000 images consume 40.87 seconds of the available time). Secondly, because several attempts were done to obtain results with more than 6000 images and the kernel frequently crashed for problems related to memory or error connections with D-Wave servers interrupting the flux of the simulations.

order to obtain statistical and reliable results. The most important aspect to note in this Figure is that in equal circumstances the HRBM (blue) outperforms the CRBM 50 h_{nodes} (green), and also outperforms the CRBM 300 h_{nodes} (red). Moreover, as expected the one with highest accuracy is the CRBM with 28×28 MNIST because is the one that receives the biggest ammount of information to learn (yellow). Nonetheless, it can be inferred from these results that when the embedding of the MNIST 28×28 into the QPU of D-Wave will be available, the results of HRBM will be considerably better than the yellow CRBM. Following the previous plots, tables with the accuracy computed for each simulation can be observed in 1 and 2 and images generated can be observed as well in Figures 13, 14 and 15 respectively. In these two tables the first column corresponds to the size of the MNIST dataset which in the classical RBM it is analyzed for both 784 and 49 input pixels respectively and in the hybrid RBM only 49 input pixels because of hardware constraints previously mentioned. The number of hidden nodes is also decreased (from 300 to 50) to see if as expected the accuracy would diminish. The next parameter analyzed is the number of epochs that ranged from 1 to 100 in 4 steps (1, 10, 50, 100) and run 5 times each. Then, average of each model was computed with its respective uncertainty (last column).

CRBM	h_{nodes}	Epochs	T1	T2	T3	T4	T5	$\langle Accuracy \rangle$
28×28	300	1	96%	100%	100%	100%	100%	$(99.20 \pm 0.80)\,\%$
		10	100%	96%	100%	96%	100%	$(98.40 \pm 0.91)\,\%$
	50	1	0%	4%	4%	0%	0%	$(1.60 \pm 0.98)\,\%$
		10	36%	44%	12%	16%	16%	$(24.80 \pm 6.37)\,\%$
		50	68%	68%	60%	68%	64%	$(65.60 \pm 1.60)\,\%$
		100	68%	64%	76%	60%	64%	$(66.40 \pm 2.71)\,\%$
7×7	50	1	0%	0%	0%	0%	0%	$(0.0 \pm 0.0)\,\%$
		10	8%	4%	12%	12%	12%	$(9.60 \pm 1.6)\,\%$
		50	8%	16%	16%	12%	8%	$(12.00 \pm 1.79)\,\%$
		100	20%	12%	12%	24%	16%	$(16.80 \pm 2.33)\,\%$
	300	1	0%	0%	0%	0%	0%	$(0.0 \pm 0.0)\,\%$
		10	4%	8%	8%	8%	8%	$(7.20 \pm 0.80)\,\%$
		50	12%	20%	20%	20%	20%	$(18.4 \pm 1.60)\,\%$
		100	20%	32%	36%	28%	16%	$(26.40 \pm 3.71)\,\%$

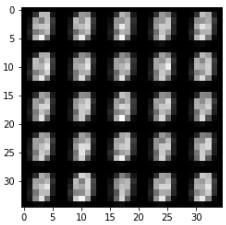
Table 1: Fully classical RBM trained with different datasets, hidden nodes and epochs.

HRBM	h_{nodes}	Epochs	T1	T2	T3	T4	T5	$\langle Accuracy \rangle$
7×7	50	1	0%	0%	0%	0%	0%	$(0.0 \pm 0.0)\,\%$
		10	8%	12%	12%	12%	8%	$(10.40 \pm 0.98)\,\%$
		50	16%	32%	20%	32%	32%	$(26.40 \pm 3.49)\,\%$
		100	40%	36%	32%	36%	52%	$(39.20 \pm 3.44)\,\%$

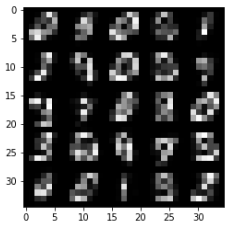
Table 2: Hybrid RBM trained with the reduced dataset, and 50 hidden nodes through different epochs.

Experiments with QA proved to be better than classical methods under same circumstances, improving up to a 22.4% in the best case scenario. It can be observed that reduction of v_{nodes} means less input information and more difficulty learning heavily reducing its accuracy. Also, as the number of h_{nodes} shrinks, the accuracy does as well. A higher number of h_{nodes} allows to distribute more and better the information of each image. Regarding epochs, it is clear that as its value increases, accuracy improves as well manifesting that the RBM is actually learning. Some RBM images generated are presented in

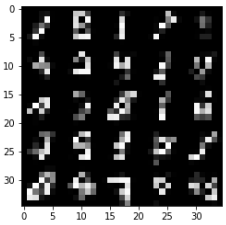
Figures 13, 14 and 15.



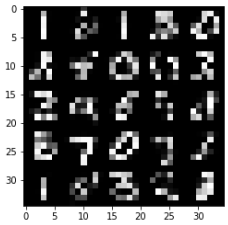
(a) $7x7$ MNIST 1 epochs.
CRBM accuracy 0%



(b) $7x7$ MNIST 10 epochs.
CRBM accuracy 4%

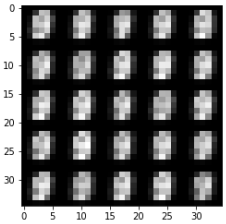


(c) $7x7$ MNIST 50 epochs.
CRBM accuracy 12%

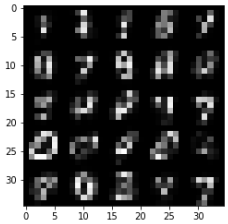


(d) $7x7$ MNIST 100 epochs.
CRBM accuracy 16%

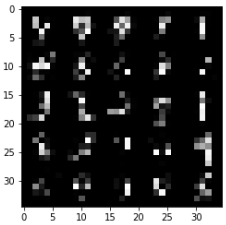
Figure 13: Numbers generated by the CRBM 7×7 $50h_{nodes}$ depending upon epochs.



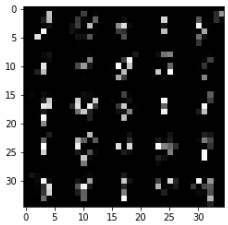
(a) $7x7$ MNIST 1 epoch.
HRBM accuracy 0%



(b) $7x7$ MNIST 10 epochs.
HRBM accuracy 12%

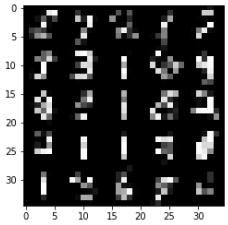


(c) $7x7$ MNIST 50 epochs.
HRBM accuracy 16%

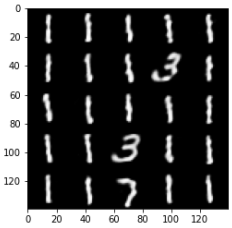


(d) $7x7$ MNIST 100 epochs.
HRBM accuracy 36%

Figure 14: Numbers generated by the HRBM 7×7 $50h_{nodes}$ depending upon number of epochs.



(a) $7x7$ MNIST 100 epochs. CRBM accuracy 36%.



(b) $28x28$ MNIST 10 epochs. CRBM accuracy 100%.

Figure 15: Numbers generated by the CRBM $300\,h_{nodes}$ depending upon the datasize, and epochs.

Some interesting outcomes can be obtained after checking results qualitatively and quantitatively. Firstly, observing the images generated by RBMs 7×7, a considerable decrease in quality and accuracy can be appreciated. Compare all figures of 13 and 14 with Fig. 15b ($28 \times 28 \rightarrow 7 \times 7$), dataset is reduced in size and the RBM looses a lot of input information. There is a decrease in the number of nodes of about 93.75% in the visible layer and of 83.33% in the hidden layer. This reduction makes it very difficult for the RBM to learn patterns accurately and to generate more complex numbers correctly. It is also noticeable that after reducing the v_{nodes}, if h_{nodes} are decreased as well, the images and accuracies worsen (compare Fig. 15a with 13 and 14, $300\,h_{nodes} \rightarrow 50\,h_{nodes}$). Secondly, it can be appreciated that for an HRBM of 100 training epochs (Fig. 14d), the most common number generated is 1 followed by 7 and 4 (compare with the CRBM under the same circumstances, Fig. 13d which has a wider spectrum of numbers). This is the reason why the accuracy of the HRBM is considerably higher, 1 is the easiest number to categorize for the ResNet, hence assigns a better performance to the HRBM. It is important to note that the numbers generated by the RBM are random, not requested, and from 5 simulation runs in model 14d, no 5 nor 8 were found for example. The most predominant numbers in 14d are 1, 7 and 4. Furthermore, this effect of non diversity can also be appreciated in the CRBM with full information as well (Fig. 15b), in which the most common number is 1 again.

Images generated by HRBM 100 epochs may be thought not to be as well as generated by the CRBM with 100 epochs (compare 14d with 13d). However, numerical results in the ResNet showed an improvement discriminating numbers when using 14d rather than 13d enhancing the probability of the most probable ones and lowering the probability of the complementaries. This discrepancy between quality of images and accuracy assigned to each RBM can be explained by the distribution of pixels intensity in the images. Pixel distribution is better spread by the HRBM (besides of generating more 1s) than by the CRBM allowing the ResNet to evaluate each image with higher confidence, giving the HRBM better accuracy, meanwhile even though the CRBM shows a diversified class of numbers for all epochs, it draws them foggier, making the ResNet not to be 100% sure about its decision thus lowering the accuracy of the CRBM. Note that the ResNet threshold confidence is set to $\geq 95\%$ otherwise the number generated by the RBM will not be considered as a good number and the accuracy of such RBM will decrease. The ResNet-18 assigns to each number generated by the RBM a probability $p(j) \in [0, 1]$ where $j \in [0, 9]$.

6 Conclusions

Several conclusions could be inferred from previous section. First of all, more analysis needs to be done with the HRBM mainly. CRBM works as expected (see Figures 10a and 10b), nonetheless, the HRBM shows a not fully convergent behaviour in 11a after 10000 annealing run images. The behaviour of the HRBM has been determined as convergent because the updates (Fig. 11b) were convergent, not the gradient of the loss (at first sight). Fig. 11a shows a scarce weak convergence at the very end of the plot. Nonetheless, it cannot be verified with full determination as there was no more available time from D-Wave´s side. Moreover, the variety of numbers generated needs to be studied more. Once the epochs are increased to 100 and QA boosting is implemented in the HRBM the diversity reduces considerably (compare Fig. 13d and 14d). Simulations were extended (out of the scope of this document) up to 10 runs in some cases and a clear predominance of the number 1 was observed. Theory and computations were repeated and examined deeply with no final incongruencies found in the implementation of the annealing or during trainings.

Additionally, it would be of high interest to have the possibility to afford for more time in D-Wave cloud service because with 1 minute only a boost of up to 17% of a single epoch can be performed (it is not possible to run 1 full epoch of QA). As a clarification, trainings were done up to 100 epochs ($\sim$ 5 hours per simulation) because for larger trainings the performance of the images generated was not better (they were basically the same in quality and accuracy compared with the ones of 100 epochs). The aim to use a D-Wave quantum computer was not only to increase the performance of the RBM but to accelerate the training time, theoretically achieved taking into consideration only the annealing times ($20\mu s$ per annealing) but not accomplished in reality because various impediments were encountered during training such as the internet latency and the information processing time of the problem sent to the servers (see Fig. 9a). It is true that the sampling is very quick but the transmission of information was what took the most putting into manifest the need of developing quantum communication protocols to send data faster or to develop quantum computers locally where all the software can be run in place instead of sending and receiving packets. Also, accuracy values are very low (see Tables 1 and 2) because the threshold settled in the ResNet-18 is too high ($\geq$ 95%). There were results where many generated examples such as 2 (or 6) correctly identifiable for the human eye had a probability of being correct of 49% (57%) being the other labels (complementary to 2 and 6) almost 0%. These correct numbers with $<$ 95% probability have been discarded because of this threshold. This limit must be treated carefully because the RBM accuracy depends heavily on this parameter. If a limit of 75% was applied instead, results would be much better numerically in tables and accuracy plots (but not graphically generating hand-written numbers).

Finally, as a brief summary to sum up all these conclusions into one single paragraph, several pros and cons can be addressed to enclose this section. As positive results, a good gap loss of the free energy was obtained for several RBM models. The accuracies were pretty high for such ResNet threshold barrier. The RBM could be successfully embedded into the QPU and the gradients of the loss function could be computed easier without need of classical approximation. Moreover, the annealing boost of D-Wave found a better configuration of the RBM reaching a lower minimum state energy. On the other hand, more study needs to be done to determine why the RBM does not generate with the same probability a 2 or a 7 even in the most ideal case scenario (Fig. 15b) in which almost all

numbers are 1 and there are only two 3 and one 7. Also, why the quality of the images is not as good as expected in some scenarios even if the accuracy increased considerably and the need of more D-Wave sampling time to demonstrate that 11a truly converges as time increases.

7 Outlook and possible applications

Two paths can be followed from this point. One is the track of more research understanding how the RBM evolves as parameters change and finding ways to improve its efficiency and quality, and the other is to use RBMs for several applications.

On one hand, simulations could be repeated with a different topology (Pegasus instead of Chimera) to see if quality of images generated improves, or even easier, repeat the process but with a higher number of samplings per annealing (1000 instead of 500) to look for differences in performance. Also, as stated in previous section, more analysis should be done in order to determine why 1 is more likely to be generated than 2 for example because numbers are equally distributed and appear with same frequency in the datasets.

On the other hand, generative models open the door to many possible applications. One possible use case is to apply this model as a defence mechanism for classifying emails, e.g., as a DBN[10] (See Fig. 16). DBNs are stochastic binary models and can be used to check if a determined word appears in a text (node i gets value 1) or not (node i gets value 0). For example, to know if a word appears in a text it can be easily checked by splitting the email into words and storing them into a list. If (for example) word "urgent" appears, node i gets value 1, if not, it gets value 0. Once the DBN is trained with a boost of D-Wave, it can be tested with another DBN trained classically (e.g., parallel tempering again) and compare which model has a better performance. For a DBN that recognizes 30 words it would be necessary 30 v_{nodes} plus 20 h_{nodes} (as hyper parameter) and 3 (to categorize) in the output layer being needed a total of 53 nodes (53 logical qubits). This model is affordable for current quantum hardware and could be proposed as a future exercise or continuation of this project. Nonetheless, this model would be too weak against current state-of-the-art email filtering models. If comparable performance is aimed, a minimum of 200 words per label (legitimate, spam, phishing) must be used. Repeating the process it would be necessary a minimum of 600 v_{nodes}, 300 h_{nodes} and 3 output nodes, summing up a total of 903 nodes (logical qubits) for the DBN. Unnafordable for current D-Wave systems because it would be necessary a minimum of 7224 qubits (see architecture of Fig. 5). Even though the second model is not feasible to implement currently, it would be interesting to see if the performance of the quantum enhanced model (HDBN) would be better than the classical one (CDBN) for only 30 words (53 logical qubits) before waiting for improvements in quantum hardware to run the experiment with 903 logical qubits.

[10] A Deep Belief Network is a generative graphical model that consists of stacking several RBMs together creating a network of various hidden layers plus an output classification layer.

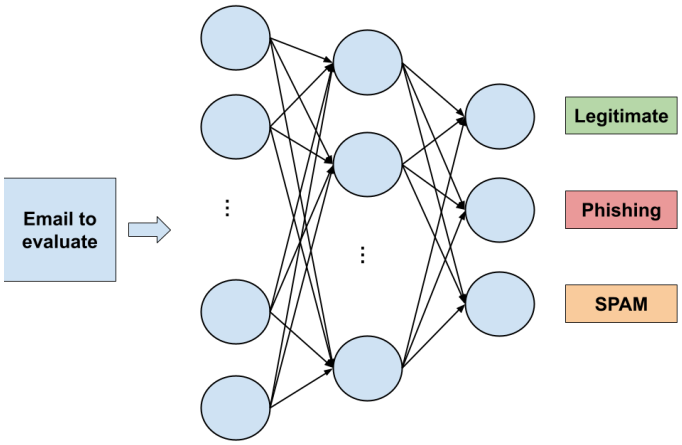


Figure 16: DBN evaluation model.

Many applications can be created from PGMs in the field of cybersecurity, RBMs and DBNs could also be used in networks for DDoS mitigation (22), malware classification (23) or to generate QR (Quick Response) codes for 2-Factor Authentication steps. A QR code needs a minimum of 76×76 pixels (49). Repeating the same procedure, the number of v_{nodes} needed would be $76^2 = 5776$ and taking into account the hidden layer would be approximately a total of 11500 nodes (logical qubits) which is not feasible either.

To summarize, PGMs are a very powerful tool which should be taken into account for current IT defence systems. As time goes on attack models improve in performance and speed. The merge of ML with quantum computing should be observed and treated carefully since now more than ever because in the wrong hands it can cause a great damage not only to companies but to governments as well if resilience counter measurements are not already prepared for the moment of the incidence. There is also much work to do in the foundational research area. For example, further investigation is needed to look for developments in quantum hardware (that may be enhanced by the use of photonic circuits) to not to reduce datasets and to start treating bigger images, texts and audios.

8 Bibliography

References

[1] Defence against adversarial attacks using classical and quantum-enhanced Boltzmann machines. By A. Kehoe, P. Wittek, Y. Xue, A. Pozas-Kerstjens

[2] Application of Quantum Annealing to Training of Deep Neural Networks By Steven H. Adachi, Maxwell P. Henderson.

[3] GitHub repository Energy-based models. By Alex Pozas-Kerstjens.

[4] Gradient Descent Algorithm — a deep dive. By Robert Kwiatkowski.

[5] Training and Classification using a Restricted Boltzmann Machine on the D-Wave 2000Q. By Vivek Dixit, Raja Selvarajan, Muhammad A. Alam, Travis S. Humble and Sabre Kais.

[6] Generative Modeling for Machine Learning on the D-Wave. By Sunil Thulasidasan.

[7] Adversarial Examples, Uncertainty, and Transfer Testing Robustness in Gaussian Process Hybrid Deep Networks By John Bradshaw, Alexander G. de G. Matthews and Zoubin Ghahramani.

[8] How Wrong Am I? - Studying Adversarial Examples and their Impact on Uncertainty in Gaussian Process Machine Learning Models. By Kathrin Grosse, David Pfaff, Michael Thomas Smith and Michael Backes.

[9] Comparison of quantum and classical methods for labels and patterns in Restricted Boltzmann Machines By Vivek Dixit, Yaroslav Koshka, Tamer Aldwairi, M.A. Novotny

[10] Quantum Machine Learning. By Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd

[11] Attacking the Quantum Internet. By Takahiko Satoh, Shota Nagayama, Shigeya Suzuki, Takaaki Matsuo, Michal Hajdušek, and Rodney Van Meter.

[12] MixDefense: A Defense-in-Depth Framework for Adversarial Example Detection Based on Statistical and Semantic Analysis By Yijun Yang, Ruiyuan Gao, Yu Li, Qiuxia Lai, Qiang Xu.

[13] A Gentle Introduction to Markov Chain Monte Carlo for Probability. By Jason Brownlee.

[14] Gradient Descent With Momentum from Scratch. By Jason Brownlee.

[15] Gradient Descent With RMSProp from Scratch. By Jason Brownlee.

[16] Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. By Jason Brownlee.

[17] Adam - Optimization. By Cornell University.

[18] Generative vs. Discriminative Algorithms By A. Aylin Tokuç.

[19] Generative Adversarial Nets By Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengi.

[20] Towards Evaluating the Robustness of Neural Networks By Nicholas Carlini, David Wagner.

[21] DeepFool: a simple and accurate method to fool deep neural networks. By Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Pascal Frossard.

[22] Restricted Boltzmann Machine based Detection System for DDoS attack in Software Defined Networks. By P.MohanaPriya and S.Mercy Shalinie.

[23] On the Evaluation of Restricted Boltzmann Machines for Malware Identification. By Kelton A. P. da Costa, Luis Alexandre da Silva, Guilherme Brandão Martins

[24] Application of Quantum Annealing to Nurse Scheduling Problem By Kazuki Ikeda, Yuma Nakamura Travis S. Humble.

[25] Ising/QUBO problem By blueqat.
[26] GitHub blueqat tutorials. By blueqat
[27] THE MNIST DATABASE of handwritten digits By Yann LeCun, Corinna Cortes, Christopher J.C. Burges.
[28] D-Wave documentation
[29] What is quantum annealing? By D-Wave
[30] D-Wave Workflow: Formulation and Sampling
[31] D-Wave Operation and Timing.
[32] Batch, Mini Batch Stochastic Gradient Descent By Sushant Patrikar.
[33] Applying deep learning and a RBM to MNIST using Python. By Adrian Rosebrock.
[34] Neural Networks and Deep Learning. Chapter 1: Using Neural Nets to recognize handwritten digits. By Michael Nielsen.
[35] Deep Residual Learning for Image Recognition. By Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun.
[36] Introduction to ResNets. By Connor Shorten.
[37] Intuitive Explanation of Skip Connections in Deep Learning. By Nikolas Adaloglou.
[38] GitHub repository ResNet-18 example. By Sebastian Raschka.
[39] Getting Started with D-Wave Solvers.
[40] A Path Towards Quantum Advantage in Training Deep Generative Models with Quantum Annealers. By Walter Vinci, Lorenzo Buffoni, Hossein Sadeghi, Amir Khoshaman, Evgeny Andriyash, and Mohammad H. Amin.
[41] A Parallel Tempering algorithm for probabilistic sampling and multimodal optimization. By Malcolm Sambridge.
[42] Parallel tempering: Theory, applications, and new perspectives By David J. Earlab and Michael W. Deem.
[43] Monte Carlo & Parallel Tempering. By John Gergely.
[44] Computational Approaches for Understanding Dynamical Systems: Protein Folding and Assembly. By Qinghua Liao.
[45] Understanding and visualizing ResNets. By Pablo Ruiz.
[46] End-to-end pre-trained CNN-based computer-aided classification system design for chest radiographs. By Yashvi Chandola.
[47] Residual Neural Network.
[48] Adam Optimization Algorithm. Overview of an effective optimization algorithm. By Kurtis Pykes.
[49] QR Code Minimum Size: How Small Can a QR Code Be? By Scott Schulfer.
[50] Quantum Annealing. By Alba Cervera-Lierta.
[51] Quantum Computing languages landscape. By Alba Cervera-Lierta
[52] Connections: Log Likelihood, Cross Entropy, KL Divergence, Logistic Regression, and Neural Networks. By Rachel Draelos.
[53] QR Code Security: What are QR codes and are they safe to use? By karspersky.
[54] Neural Networks series.
[55] Quantum Annealing series. By D-Wave.
[56] Adam Optimization Algorithm By Andrew Ng.
[57] Quantum Programming with D-Wave | Webinar
[58] What is an RBM (Restricted Boltzmann Machine)? By IBM.
[59] GitHub repository project. By Álvaro Huanay de Dios

A Loss function derivation

The loss function of the model can be expressed in the following form

$$L_\theta(T) = -\frac{1}{T}\sum_{x^{(i)}\in T} log\left(p_\theta(x^{(i)}\right)$$

Nonetheless, there is an easier way to represent it in order to compute the gradients of the loss function and minimize its free energy. Using equations 1, 2, 3 and 7 for the RBM model

$$L_\theta(T) = -\frac{1}{|T|}\sum_x log\left(\frac{1}{Z}\sum_h e^{-E_\theta(x,h)}\right) = -\frac{1}{|T|}\sum_x\left(log\left(\sum_h e^{-E_\theta(x,h)}\right) - log\left(Z\right)\right) =$$

$$= -\frac{1}{|T|}\sum_x\left(log\left(e^{-F_\theta(x)}\right) - log\left(\sum_{x,h} e^{-E_\theta(x,h)}\right)\right) = \frac{1}{|T|}\sum_x\left(F_\theta(x) + log\left(\sum_x e^{-F_\theta(x)}\right)\right) =$$

$$= \frac{1}{|T|}\sum_x F_\theta(x) + log\left(\sum_x e^{-F_\theta(x)}\right)$$

A new loss function equation is obtained. It can be separated into two terms labeled as the data term (easy to compute, that depends on variable $|T|$), and the model term that depends on all the possible neuron configurations (2^N, not easy to compute).

B Gradient of the loss function derivation

The derivation of the gradient of the loss function is a little more elaborate process. Using the chain rule from the last equation of appendix A

$$\frac{\partial L_\theta}{\partial\theta} = \frac{1}{T}\sum_x \frac{\partial F_\theta(x^i)}{\partial\theta} + \frac{\sum_x -\left(\partial_\theta F_\theta(x)\right)e^{-F_\theta(x)}}{\sum_x e^{-F_\theta(x)}} = \frac{1}{|T|}\sum_x \frac{\partial F_{\theta(x^i)}}{\partial\theta} - \frac{1}{Z_\theta}\sum_x e^{-F_\theta(x)}\frac{\partial F_{\theta(x)}}{\partial\theta} =$$

$$= -\sum_x \frac{1}{Z_\theta}\left(\sum_h e^{-E_\theta(x,h)}\right)\frac{\partial F_\theta(x)}{\partial\theta} + \frac{1}{|T|}\sum_x \frac{\partial F_\theta(x^i)}{\partial\theta} =$$

$$= \frac{1}{|T|}\sum_{x^{(i)}\in T}\frac{\partial F}{\partial\theta} - \sum_x p_\theta(x)\frac{\partial F_\theta(x)}{\partial\theta} = \left(\langle x_i\rangle_{data} - \langle x_i\rangle_{model}\right)$$

Where the terms of the last equality correspond to the positive and negative phases respectively. As can be seen, in the second term it is needed $p_\theta(x)$ (or equivalently the partition function Z_θ) with all the possible neuron configurations which as stated in section 2.1 is not possible to calculate classically. For that reason classical approximations are used instead and quantum annealing sampling is proposed as a solution in this document.