UNIVERSITÀ
DEGLI STUDI
FIRENZE

Scuola

di Ingegneria

Corso di Laurea
Magistrale in
Intelligenza Artificiale

Modelli generativi quantistici: dalle gan ai modelli diffusivi quantistici

Quantum generative models: from quantum gan to quantum diffusion models

Candidato:
Paolo Marino

Relatore:
Prof. Filippo Caruso

Anno Accademico 2023-2024

Contents

Abstract

Generative models are gaining in popularity and are increasingly used. Deep Learning is growing very fast and along with this speed of growth it is bringing with it increasingly attractive results but also new problems. Training a complex neural network requires more and more resources and training time, so efforts are focusing on how to achieve optimal results while using as few resources as possible. Quantum computing could be the answer.

This thesis explores the intersection of quantum computing and generative models exploiting quantum properties to reproduce probability distributions and the speed-up that a quantum computer could provide. In particular, in this work, an attempt was made to combine classical machine learning to try to implement quantum versions of generative adversarial network and diffusion models.

1 Introduction

Problem statement The aim of this work is to combine the use of quantum computing with the use of generative models to reproduce statistical properties exhibited by particle images.

Everything can be interpreted as a mathematical function or probability distribution, even the values taken by the pixels of an image, and it is on this that generative models are based. What these models try to do is to try to reconstruct the probability distribution of a certain input data in order to recreate it as closely as possible.

Generative models have improved dramatically over the years, and many different techniques have been used to improve in this area: from recurrent networks, autoencoders, variational autencoders, through to the most widely used models in this area today, namely gan and diffusion models. Generative models have improved dramatically over the years, and many different techniques have been used to improve in this area: from recurrent networks, to autoencoders, to variational autencoders, to the most widely used models in this area today, namely gan and diffusion models. Today, these state-of-the-art models are used in a variety of tasks that are always gaining in popularity, such as image, video and music generation. In this work, I decided to combine quantum mechanics with the most commonly used generative models, i.e. gan and in particular diffusion models, to see if it is possible to achieve any results in tasks such as image generation even with limited resources, and if there may be some form of advantage in using quantum technology.

To do this, I decided to build on existing work on quantum gan and quantum diffusion models by trying to implement my own version.

1.1 Classical Machine Learning

At the beginning of this paper, I want to introduce briefly classical machine learning, so that I can then make it clearer how the quantum version can be introduced into this paradigm.

Machine learning is a discipline that consists, in short, of making predictions about data by understanding certain patterns. Basically, it consists of a set of specific techniques to solve different kinds of problems, for example regression problems or categorical problem.

Regression problem: Predicting a continuous numerical value e.g. predicting price fluctuations based on certain variables. For example include predicting prices (of homes, stocks, etc.), based on some variables. it's possible to use linear: suppose that we wish to estimate the prices of houses (in dollars) based on their area (in square feet) and age (in years) price $= w_{\text{area}} \cdot \text{area} + w_{\text{age}} \cdot \text{age} + \epsilon$, it's possible to express it as $\hat{y} = w_1 x_1 + \cdots + w_d x_d + b$ where $x_1 ... x_d$ are the variables, $w_1 ... w_d$ are the coefficients of **weights** and b called the bias [5].

Categorical problem: Predict to which class a certain data item belongs. E.g. given a number of pictures of dogs, cats or horses predict to which class each picture belongs. The first step is to convert the classes into numeric vectors, an example of encoding is one-hot encoding which consists of generating n-dimensional vectors (where n = number of labels), which inserts one at the category instance and zero elsewhere. In our case we would have three-dimensional vectors of the type $y \in \{(1,0,0),(0,1,0),(0,0,1)\}$[5]

Machine learning is typically divided into two phases: a training phase in which the model is trained to recognise certain patterns in the data using a loss function, which is a mathematical function expressing the distance between the result obtained and the expected result, and an optimiser, which is used

to optimise the parameters to reduce the loss. Obviously, the choice of the right loss and the right optimiser depend on the specific problem being addressed, as do the components of the model. For example, in the case of a regression problem, the objective could be to minimise a loss function like this: $l^{(i)}(w,b) = \frac{1}{2}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$ where $\hat{y}^{(i)}$ is the target and $y^{(i)}$ is the predicted loss. There are several algorithms for optimising parameters, one of which is called gradient descent, which consists of taking the derivative of the loss of a mini-batch of elements in the dataset and calculating the derivative with respect to the model parameters, multiplying the derivative by a certain small value called the learning rate and subtracting this by the current parameter values

$$(w,b) \leftarrow (w,b) - \frac{\eta}{|B|}\prod_{i\in B_t}\frac{\partial l^{(i)}}{\partial(w,b)}.$$

The second phase is called the testing phase and is used to see if the model is able to generalise the results obtained in the training phase even with data it has not previously seen.

There are several branches of this discipline which is very broad, so I would like to focus on a specific part which in turn includes generative models (e.g. diffusion models), i.e. deep learning: the typical neural network is called a multilayer perceptron, which consists of a series of interconnected layers of neurons, where each first layer takes the raw data, processes it and gives it as input to the next layer. So on until the final output is obtained

In contrast, in the case of a classification problem we have many outputs corresponding to the various classes to be expected.

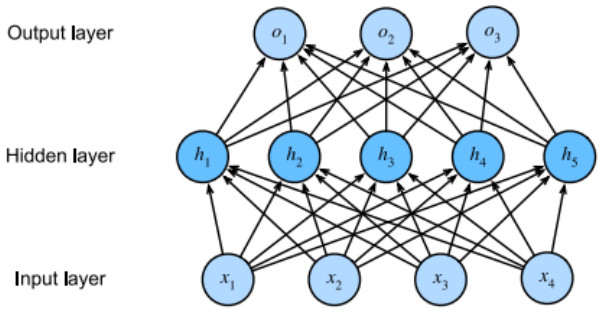


Figure 1: Example of a classical multilayer percerptron, formed by 3 layers where the first takes the initial data (input layer), the second, called the hidden layer, further processes the data while the last output layer processes the data until it takes on the size of the output [5].

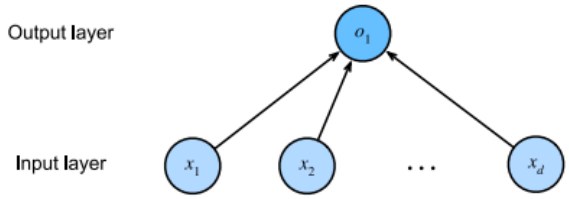


Figure 2: Example of a multilayer percerptron with one output [5].

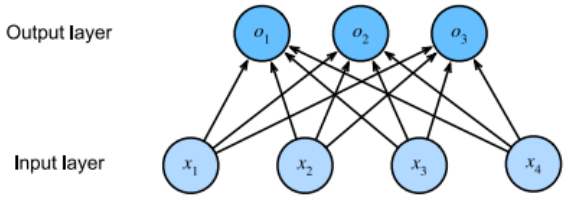


Figure 3: Example of a classical multilayer percerptron with multiple outputs [5].

The last key step for the proper functioning of a neural network is the use of a function that adds non-linearity. This is a key concept in understanding the su ccess that neural networks have had. As explained above, a multilayer perceptron consists of several hidden layers and an output layer, which can be expressed with the following affine equations

$$H = XW^{(1)} + b^{(1)}$$
$$O = HW^{(2)} + b^{(2)}.$$

Where W is the weight matrix and b the bias vector. However, putting several such hidden layers together means putting many affine functions one after the other, but the sum of many affine functions is nothing but an affine function

$$O = (XW^{(1)} + b^{(1)})W^{(2)} + b^{(2)} = XW^{(1)}W^{(2)} + b^{(1)}W^{(2)} + b^{(2)} = XW + b.$$

This is a problem when dealing with non-linear problems as not all variables are linearly related to the final output. If we take the example of predicting the cost of a house based on square metres, we have a linear relationship so there is no problem, but often it is not so simple. This is why activation functions are added at the end of each layer. For instance, a popular choice is the ReLU (rectified linear unit) activation function $\sigma(x) = \max(0, x)$. When the input is negative, the derivative of the ReLU function is 0, and when the input is positive, the derivative of the ReLU function is 1, so the derivative vanish or let the argument through.

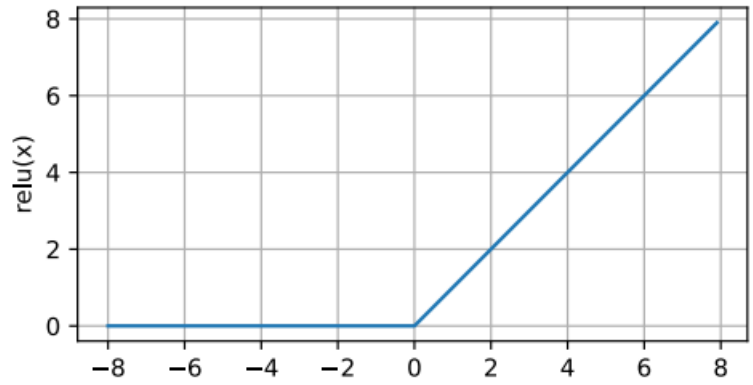


Figure 4: ReLU function: negative inputs are converted to zero while psoitive inputs retain their values [5].

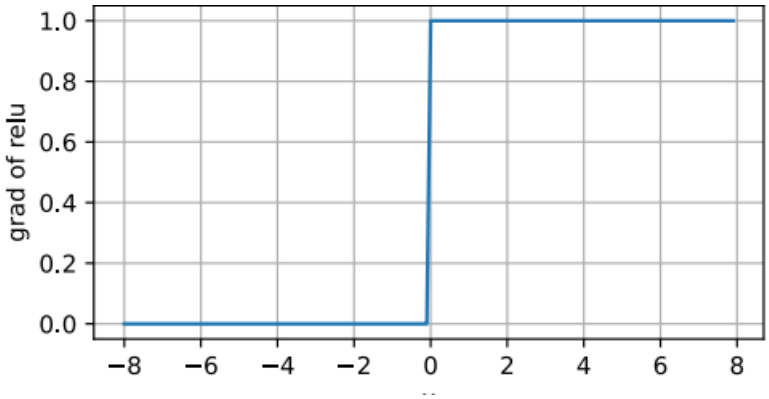


Figure 5: derivative of ReLU function: all negative values obviously remain zero while positive ones have derivative 1 [5].

1.2 Generative models

Until now, we have been talking about a family of supervised problems as they are trained knowing the answer to the problem. For example, during training, in the case of price prediction we already know the price to be predicted and in the case of classifying pictures of dogs and cats we already know whether a certain picture corresponds to a dog or a cat. This type of machine learning is called supervised machine learning. In this work we focus on a different family of unsupervised problems, i.e. problems that do not have 'answers' already given.

Generative models are a type of model used in machine learning to create new data that appears to come from a certain probability distribution. These models aim to capture the underlying structure of the training data and are capable of generating new examples that are statistically similar to the original data. For example, if we have a dataset of faces, the goal of these generative models is to learn how to generate data similar to that of the dataset. In particular, we will concenct two generative methods, the generative adversarial networks and diffusion models.

1.2.1 Generative adversarial networks

Generative Adversarial Networks (GANs) represent a revolutionary approach in the field of artificial intelligence and machine learning, particularly in the realm of generative modeling. Introduced by Ian Goodfellow and his colleagues in 2014 [2], GANs have rapidly gained prominence for their ability to generate highly realistic data, including images, audio, and text, that closely resemble authentic samples from a given dataset.

At the core of GANs lies a unique architecture comprising two neural networks: the generator and the discriminator. These networks engage in a captivating adversarial process, where the generator strives to produce synthetic data that is indistinguishable from genuine data, while the discriminator aims to differentiate between real and fake samples. Through iterative training, the generator progressively enhances its ability to fabricate realistic outputs, while the discriminator becomes more adept at discerning between real and synthetic data. This adversarial interplay ultimately leads to the refinement of both networks, resulting in the generation of remarkably convincing data distributions:

To learn generator's distribution p_g over data x, define noise variables $p_z(z)$, then represent a mapping to data space as $G(z, \theta_g)$ where G is a function represented by a multilayer perceptron with parameters θ_g.

$D(x)$ represents the probability that x came from the data rather than p_g. train D to maximize the probability of assigning the correct label to both training examples and samples from G and simultaneously train G to minimize $log(1 - D(G(z)))$.

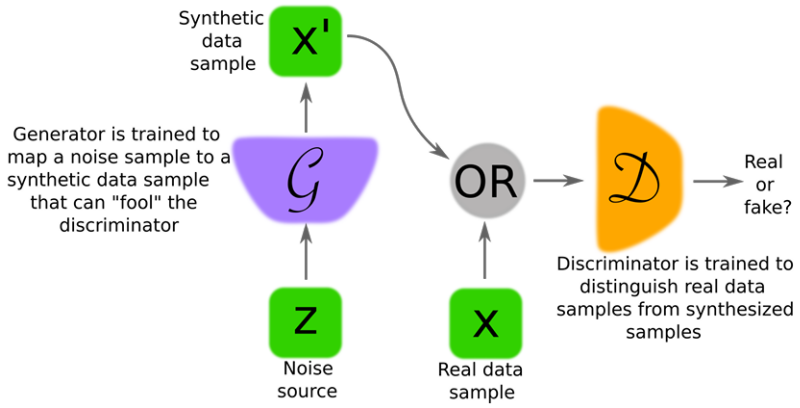


Figure 6: Scheme of GAN network [11].

D and G play the following two-player minimax game with value function $V(G, D)$:

$$V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

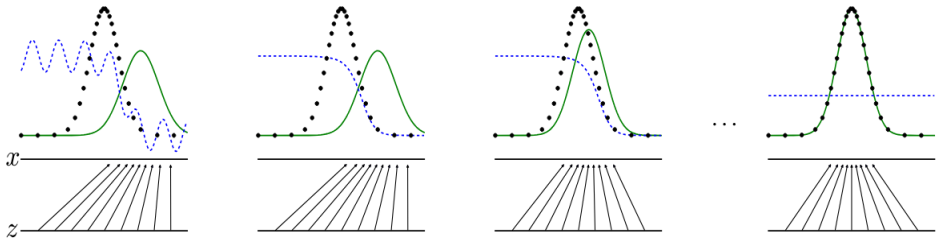


Figure 7: Distributions of real data, fake data and discriminator[2].

- discriminative distribution D, blue, dashed line
- real distribution p_x black, dotted line
- generative distribution p_g green, solid line

D converge to $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_g(x)}$, if $p_g = p_{data}$ the discriminator is unable to differentiate between the two distributions: $D(x) = \frac{1}{2}$.

Advantages and Disavantages. G must not be trained too much without updating D, in order to avoid "the Helvetica scenario" in which G collapses too many values of z to the same value of x to have enough diversity to model p_{data}. the main disadvantages are:

- difficulty in converging both models
- mode collapse, i.e. tendency to generate samples that are too similar to each other
- the discriminator may converge too quickly to zero, thus limiting the generator's ability to generate realistic samples

The advantages are that Markov chains are never needed, only backprop is used to obtain gradients, no inference is needed during learning, and a wide variety of functions can be incorporated into the model

Obviously, there are various methods for implementing the generator and discriminator architectures. Usually in the case of the generator, a multilayer percerptron is used, which returns as output an integer that can be 0 (fake) and 1 (real). While in the case of the generator one can always use an mlp that generates as output a tensor of the same size as the desired image. In the case of image generation, it has been seen, that using convolutional networks by downsampling and then upsampling and using other techniques such as batch normalisation gives better results and more stable training.

There are also variants of the gan for example the conditional gan that feeds the network with also the class or the wgan designed to be easier to train, using a different formulation of the training objective that does not suffer from the vanishing gradient problem.

1.2.2 Diffusion models

Diffusion models are generative models introduced by Sohl-Dickstein et al. [19] and used the firt time for image generation by Ho et al. [3] that works dirtying the image until you obtain pure noise and learning to recover data by reversing the noising process. It uses a Markow chain that progressively add noise in the forward process. According to a variance schedule $\beta_1, \ldots, \beta_T$

$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}), \quad q(x_t|x_{t-1}) = \mathcal{N}\left(x_t; (1-\beta_t)x_{t-1}, \beta_t I\right).$
Setting $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \frac{1}{T}\sum_{s=1}^{T} \alpha_s$ we have:
$q(x_t|x_0) = \mathcal{N}\left(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I\right)$

The image is spot-shifted until an isotropic Gaussian is obtained $p(x_T) = \mathcal{N}(x_T; 0, I)$ and we apply the backward process, trying to reconstruct the initial image:

$p_\theta(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \quad p_\theta(x_{t-1}|x_t) := \mathcal{N}\left(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\right)$

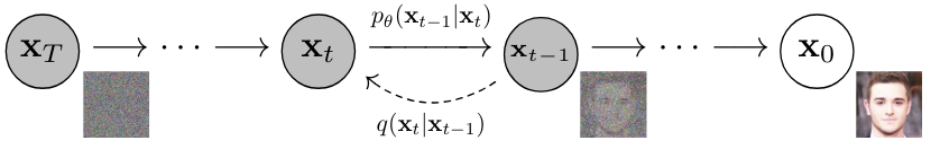


Figure 8: Diffusion Markow chain: the forward part dirties the image until pure noise is obtained, while the backward part from the noise tries to reconstruct the initial image [3].

We can summarize the algorithm in these 2 steps:

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Figure 9: Training and sampling algorithms used by the diffusion model. The former is used to train the model to recreate the probability distribution behind the original image, while the sampling algorithm takes randomly generated Gaussian noise as input and generates the desired image [3].

In the training phase, the initial image is taken and the forward step is applied in order to dirty the image with Gaussian noise $\epsilon \sim N(0, I)$ incrementally for t timesteps until the image tends to an isotropic Gaussian and then train a model to predict the added noise so that it can be used to reconstruct the image during the **sampling phase** trying to reconstruct the mean from the predicted noise.

Advantages and Disavantages. GANs are harder to train, more prone to mode collapse and other problems, and often have worse quality. Diffusion models have proven to be better than GANs at tasks such as image generation, synthesis and classification. On the other hand, diffusion Models take more time and larger datasets to train.

2 Quantum machine learning and NISQ era

2.1 Intro to quantum computing

From bit to qbit. A classical bit (cbit) is a basic unit of information that can exist in one of two distinct, or orthogonal, states. Examples of cbits include on-off switches and up-down magnets. In mathematical terms, the state of a cbit is typically represented by a logical binary digit, which can be either 0 or 1.

Alternatively, the state of a cbit can also be described using a two-dimensional "one-hot" amplitude vector. This vector contains a single "1" digit, whose position indicates whether the cbit is in the state 0 or 1. Specifically, when the cbit is in the state 0, the one-hot vector has a "1" digit in the first position, while when the cbit is in the state 1, the "1" digit is in the second position.

Cbit	Amplitude Vector
0	$\|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$
1	$\|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

Table 1: Cbit Amplitude Vectors

Ket and Bra notation. In quantum theory, Dirac's ket notation is commonly used to represent column vectors, denoted as $|a\rangle$, where a serves as an identifier for the vector. For instance, in the context of a single classical bit (cbit), the ket vector representing the value 0 is denoted as $|0\rangle$, represented as the one-hot amplitude vector: $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

On the other hand, Dirac's bra notation is used to represent row vectors. Given a ket $|a\rangle$, the corresponding bra $\langle a|$ is defined as the Hermitian transpose

of the ket vector $|a\rangle$, denoted as $|a\rangle = \langle a|^\dagger$.

The inner product between two kets $|a\rangle$ and $|b\rangle$ is defined as $|a\rangle^\dagger |b\rangle = \langle a| |b\rangle = \langle a|b\rangle$. This is known as Dirac's bra-ket notation for the inner product.

The kets $|0\rangle$ and $|1\rangle$ define an orthonormal basis for the linear space of two-dimensional vectors. They are orthogonal, i.e., $\langle 0|1\rangle = \langle 1|0\rangle = 0$, and have unitary norm, i.e., $\langle 0|0\rangle = \langle 1|1\rangle = 1$.

Throughout this discussion, all bra and ket vectors (not only $|0\rangle$ and $|1\rangle$) are assumed to have unitary norm, ensuring the condition $|| |a\rangle ||_2^2 = || \langle a| ||_2^2 = \langle a|a\rangle = 1$ for all kets $|a\rangle$ [20].

quantum bit and measurement. A quantum bit (qubit) is a two-level quantum system, such as the up-down spin of an electron or the vertical-horizontal polarization of a photon. Analogous to a random classical bit (cbit), whose state is defined by a probability vector p, the state of a qubit is described by a two-dimensional amplitude vector: $[|\psi\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} = \alpha_0 |0\rangle + \alpha_1 |1\rangle]$ There are two key differences between the state $|\psi\rangle$ of a quantum qubit and the state p of a random cbit:

- Unlike a probability vector p, the amplitude vector has complex entries α_0 and α_1. (Note that complex entries include real-valued entries as a special case.)
- The qubit state vector has the defining property of having unitary norm, i.e., $|| |\psi\rangle ||_2^2 = \langle \psi|\psi\rangle = |\alpha_0|^2 + |\alpha_1|^2 = 1$

We say that the qubit is in a superposition of states $|0\rangle$ and $|1\rangle$, with respective complex amplitudes α_0 and α_1. Mathematically, this implies that the state of a qubit is a vector that lies in a two-dimensional complex linear vector space, referred to as the Hilbert space of dimension two. The states $|0\rangle$ and $|1\rangle$ form the so-called computational basis of the Hilbert space

Being a two-dimensional vector, the state of the qubit can be equivalently

expressed as a superposition of any two orthonormal vectors forming a basis of the Hilbert space. An important example is given by the so-called diagonal basis, which consists of the two vectors $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}$

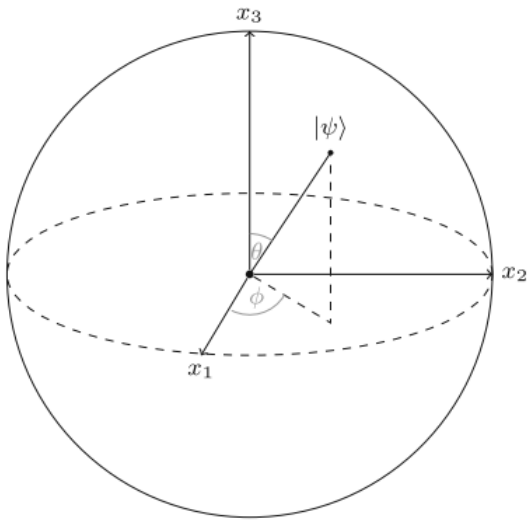


Figure 10: Bloch Sphere: The Bloch sphere provides a graphical representation of a quantum state of a two-level system (one qubit) [5].

Let us define a set of n qubits as separable if the qubits' joint state $|\psi\rangle$, a $2n$ dimensional vector, can be expressed as the Kronecker product of the individual qubit states. For two arbitrary qubit states, $|\psi_A\rangle = \alpha_{A0}|0\rangle + \alpha_{A1}|1\rangle$ for the first qubit and $|\psi_B\rangle = \alpha_{B0}|0\rangle + \alpha_{B1}|1\rangle$ for the second, the general form of a separable state with $n = 2$ is:

$$|\psi_A\rangle \otimes |\psi_B\rangle = \begin{pmatrix} \alpha_A 0 \cdot \alpha_B 0 \\ \alpha_A 0 \cdot \alpha_B 1 \\ \alpha_A 1 \cdot \alpha_B 0 \\ \alpha_A 1 \cdot \alpha_B 1 \end{pmatrix}$$

In terms of notation, a separable state in the form $|\psi_A\rangle \otimes |\psi_B\rangle$ can also be written as

$$|\psi_A, \psi_B\rangle = |\psi_A\rangle \otimes |\psi_B\rangle = |\psi_A\rangle\, |\psi_B\rangle\,.$$

A set of qubits is said to be entangled if its state cannot be written as the Kronecker product of the states of the individual qubits. Measurement outputs of entangled qubits exhibit statistical behaviors that cannot be explained by means of standard correlations of classical bits.

Important examples of entangled states for $n = 2$ qubits are given by the Bell states. For example, the so-called Bell pair

$$|\Phi+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

cannot be written as the Kronecker product of two individual qubit states. To see this, note that there is no choice for the two individual qubit states $|\psi_A\rangle$ and $|\psi_B\rangle$, which makes the tensor product $|\psi_A\rangle \otimes |\psi_B\rangle$ equal to the Bell state. Bell states are maximally entangled pairs of qubits, in the sense that any pair of entangled qubits can be obtained from a Bell pair (or another Bell state) through local operations on the two qubits.

A key fact about entanglement is that it does not survive a standard measurement (of all qubits). This is in the sense that, no matter what the quantum state is before the measurement, the post-measurement state is separable, since it is given by one of the vectors in the computational basis [5].

Quantum measurement. A von Neumann measurement in the computational basis, or standard measurement for short, takes as input n qubits in an arbitrary state $|\psi\rangle$ and it produces n classical bits (cbits) as the measurement's output, while leaving the qubits in a generally different state from the original state $|\psi\rangle$. Specifically, a standard measurement satisfies the following two properties:

- Born's rule: The probability of observing the n cbit string $x \in \{0,1\}^n$ is

$$\Pr[\text{measurement output equals } x \in \{0,1\}^n] = |\alpha_x|^2 = |\langle x|\psi\rangle|^2$$

- "Collapse" of the state: If the measured n cbit string is x, the post-measurement state of the qubits is $|x\rangle$.

Therefore, the probability of measuring a cbit string x is given by the absolute value squared of the corresponding amplitude $\langle x|\psi\rangle$. Moreover, a measurement randomly "collapses" state $|\psi\rangle$, which is generally in a superposition of computational-basis states, into a single computational-basis vector $|x\rangle$. In this regard, one can check that a system of n qubits that can only take states $|x\rangle$ in the computational basis behaves like deterministic cbits. This is in the sense that, by Born's rule, measuring a qubit in state $|x\rangle$, for $x \in \{0,1,...,2^n-1\}$, returns output x with probability 1, while leaving the qubits' state unchanged.

A standard measurement on n qubits can be implemented by applying a standard measurement separately to each individual qubit. Accordingly, each individual cbit x_k of a measurement output $x_0, x_1, ..., x_{n-1}$ corresponding to an integer $x \in \{0,1,...,2^n-1\}$ is produced by measuring the k-th qubit.

Quantum circuits. A quantum circuit contains n wires, one per qubit, and it describes the sequence of unitary matrices (U) applied to the n qubits with time flowing from left to right. A unitary matrix satisfies the equalities $UU^\dagger = U^\dagger U = I$, where I is a $2n \times 2n$.

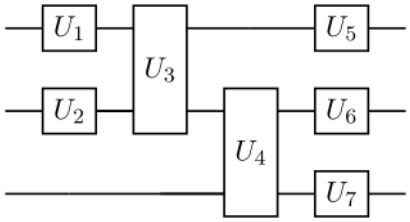


Figure 11: single-qubit gates U_1 and U_2 are applied to the first two qubits; then a two-qubit gate U_3 is applied to the first two qubits; and so on [4].

Quantum logic gates are realized by unitary transformations. Single-qubit gates are formally described by 2×2 unitary transformations, which can be represented as matrices. As an example, let's consider the X gate, which is the quantum equivalent of the classical NOT gate. The action of the X gate can be described as:

$$|0\rangle \rightarrow |1\rangle\,,$$

$$|1\rangle \rightarrow |0\rangle\,.$$

For example The X gate is represented by the matrix:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Using vector-matrix notation, it is straightforward to verify that applying the X gate to the state $|0\rangle$ results in the state $|1\rangle$, as follows:

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle .$$

A really important gate is the Hadamard gate H that creates superpositions of qubits. Its effect on the basis states $|0\rangle$ and $|1\rangle$ is given as

$$|0\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle),$$
$$|1\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

Operating on multiple qubits simultaneously is crucial in quantum computing. One of the fundamental gates in this context is the Controlled-NOT (CNOT) gate, representing a controlled operation where the state of one qubit controls the operation applied to another.

In the case of the CNOT gate, commonly referred to as the CX gate in some literature, it performs a NOT operation (equivalent to an X gate) on the target qubit if and only if the control qubit is in state $|1\rangle$; otherwise, the target qubit remains unchanged. The CNOT gate's behavior can be summarized as follows:

$$|00\rangle \rightarrow |00\rangle$$

$$|01\rangle \rightarrow |01\rangle$$

$$|10\rangle \rightarrow |11\rangle$$

$$|11\rangle \rightarrow |10\rangle$$

This behavior is accurately represented by the following matrix:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

The CNOT gate is pivotal in quantum circuits for implementing entanglement and quantum teleportation protocols, among other applications [5].

2.2 Quantum machine learning

Quantum Machine Learning is an interdisciplinary research area that combines principles from quantum computing with techniques from classical machine learning. The main goal is to leverage the unique properties of quantum mechanics, such as superposition and entanglement, to enhance the performance of machine learning algorithms on specific problems. Some examples include quantum classification algorithms, clustering algorithms, and **quantum generative models**. These algorithms consist of a **variational circuit** or **ansatz**, a **cost function** and an **optimizer**. During each iteration, the optimizer assesses the cost function with the current parameters and adjusts the parameters for the next iteration. This process continues until the algorithm converges to an optimal solution.

The hybrid nature of these algorithms stems from the fact that cost functions are evaluated using quantum resources while optimization is conducted through classical means. This combination of quantum and classical computation makes them highly versatile and suitable for a wide range of applications.

Variational algorithms typically follow a structured process to find optimal solutions.

Firstly, they initialize the quantum computer in a default state $|0\rangle$ and transform it to a desired state, referred to as the reference state $|\rho\rangle$, using a unitary reference operator U_R.

Then, they define a variational form $U_V(\vec{\theta})$, which represents a collection of parametrized states, to explore the search space from the default state to the target state $|\psi(\vec{\theta})\rangle$. The combination of the reference state and the variational form is called an ansatz.

Next, the problem is encoded into a cost function $C(\vec{\theta})$, typically a linear combination of Pauli operators, which is evaluated on a quantum system.

After evaluating the cost function, the classical optimizer analyzes the results and selects the next set of parameter values for the variational parameters.

This process iterates until the optimizer converges on an optimal solution,

which is determined by specified convergence criteria.

Finally, the proposed solution state for the problem is obtained by applying the optimized ansatz to the default state.

So, in short, one can construct a register (or ansatz) by applying a number n of qbits. The register can be defined as $|\psi\rangle = \sum_{i=0}^{2^{n-1}} \beta_i |i\rangle$ dove $\sum_{i=0}^{2^{n-1}} |\beta_i|^2 = 1$

Throughout this process, quantum resources are used for evaluating the cost function, while classical resources are employed for optimization tasks. This hybrid approach allows for the efficient exploration of complex optimization landscapes [5].

The circuits available in the short-medium term are called Noisy Intermediate-Scale Quantum (NISQ) circuits representing quantum computers with 50-100 qubits [7].

2.3 Encoding methods

When applying classical data to a quantum circuit, the key problem of data encoding arises to make them digestible by these circuits and leverage the properties of quantum physics to gain benefits. In the literature, standard encoding techniques have been defined, which can be divided into classes: basis encoding, amplitude encoding, and angle encoding.

2.3.1 Basis encoding

In this method, the classical bits are mapped into qubits using the states $|0\rangle$ and $|1\rangle$. For example, the word hello can be translated converting each ascii character into its binary form, for example the ASCII code for h would be 104 (binary: 1101000) and the corresponding quantum state is $|1101000\rangle$ [12].

2.3.2 Amplitude encoding

This encoding method uses amplitudes to encode classical information. Given a quantum state $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, classical information are encoded by adjusting amplitudes α and β. For example to encode real values $X = [1.2, 2.7, 1.1, 0.5]$, at first each term is square normalised $\frac{1}{10.19}$ and the state becomes $\frac{1.2}{\sqrt{10.19}}|00\rangle + \frac{2.7}{\sqrt{10.19}}|01\rangle + \frac{1.1}{\sqrt{10.19}}|10\rangle + \frac{0.5}{\sqrt{10.19}}|11\rangle$. The advantage of this method is that you can encode n pieces of information with $\log_2(n)$ qubits [12].

2.3.3 Angle encoding

Rotational gates can be used to encode classical information in quantum states and there are 3 types of rotation around the 3 axes R_x, R_y, R_z [12]:

$$R_x(\theta) = e^{-i\theta X/2} = \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix}$$

$$R_y(\theta) = e^{-i\theta Y/2} = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix}$$

$$R_z(\theta) = e^{-i\theta Z/2} = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}$$

So if I want to encode the whole 78 in a quantum state I treat 78 as an angle:

$$R_x(78) = e^{-i78X/2} = \begin{pmatrix} \cos(78/2) & -i\sin(78/2) \\ -i\sin(78/2) & \cos(78/2) \end{pmatrix}$$

$$R_x(78^\circ) \approx \begin{pmatrix} 0.766 & -0.642i \\ -0.642i & 0.766 \end{pmatrix} \approx [0.766 - 0.642i]$$

3 Quantum GAN

As described before, a GAN is a network that generates a probability distribution through a generator that must try to reproduce real data and a discriminator that must distinguish the real data from the generated data. This "challenge" between the two components can lead to the generation of data similar to the input data. In the case of quantum gans, the generator, the discriminator or both components can be quantum. For example in the latter case the data takes the form of an ensemble of quantum states generated by the system that the generator tries to match and the discriminator makes aribtrary measurements. An example is the qGAN implemented by Seth Loyd et al [9] where the aim is not to produce classical samples, but to create a quantum state which represents the data's underlying probability distribution.The quantum generator is trained to trasform a n-qubit input state $|\psi\rangle$ into a n-qubit output state $|G_\theta|\psi_{in}\rangle = \sum_{j=0}^{2^n-1} |\sqrt{p_\theta^j}|j\rangle$ where p_θ^j describes the probabilities of the basis state $|j\rangle$ [9]

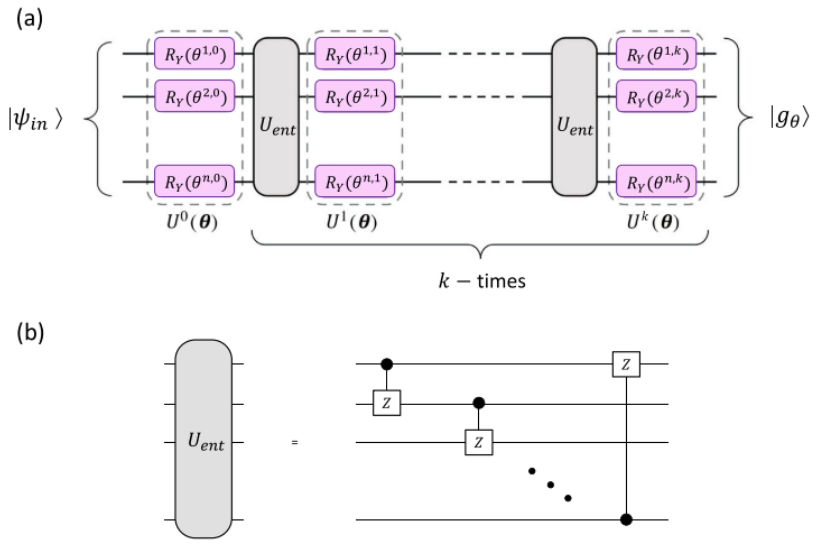


Figure 12: qGAN circuit: made of k + 1 layers of single qbit Pauli rotations and k entangling blocks U_{ent} [9]

The circuit above uses angle embedding around the y-axis followed by layers to create entangling

3.1 Quantum Gan implementation

Is it possible to apply quantum gan also for classical problems such as image generation. Due to the limited availability of resources (i.e. physical qubits), it is difficult to solve even classically seemingly trivial problems. For example, taken the mnist dataset consisting of the following digits

Figure 13: mnist images [15].

For my experiment, I decided to reproduce the quantum patch gan introduced in He-Liang Huang's paper [14]. What is done in brief is:

- Simplify the problem by taking only the digits 0 and 1 and reducing the resolution from the original 28 x 28 to a sub-resolution of 8x8
- Implementing a quantum generator
- Implementing a classical discriminator

For the realisation of the code I mainly used the libraries pennylane and pytorch, since pennylane allows a very good integration with pytorch, i.e. a

quantum circuit can be 'embedded' in a torchlayer so as to facilitate the cobination of classical and quantum layers or the use of classical optimisers. In this case, I used a series of generators, each generator having the task of taking care of a sub-part of the image. This is done because the use of a quantum circuit with so many qubits on a simulator tends to be very slow and computationally expensive, moreover real quantum computers have the problem of quantum noise which becomes more and more important as the number of qubits increases. Then a series of generators take as input the same latent vector trying to generate different parts of the image. Now let's get specific about how the various parts of the network are implemented

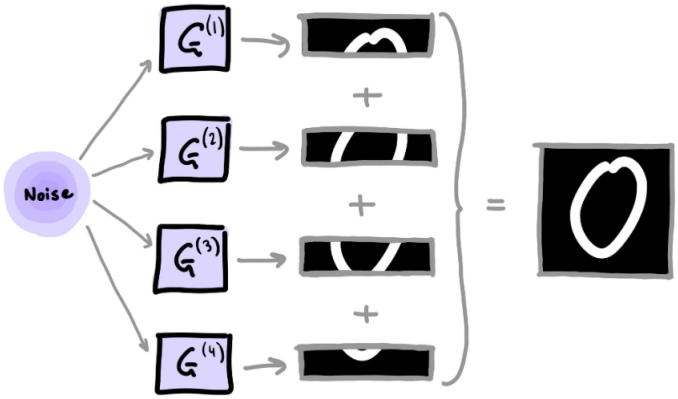


Figure 14: patch gan generators: break the image into 4 parts and apply a separate quantum circuit to each part [14].

Quantum generator As already mentioned what the single generator does is to take as input a state $|z\rangle$ and output a generated state $G_t(z)$. As a quantum encoding strategy, I used angle encoding, i.e. a rotation around the y-axis, so the state $|z\rangle$ is prepared like this

$$|z\rangle = (\bigotimes_{i=1}^{N_S} R_Y(\alpha_z(i)) \bigotimes_{k=1,k\in N-Ns}^{N-N_S} I_k)|0\rangle^{\otimes N}$$

Where N is the number of qubits and Ns is the number of ancilla qubits. Afterwards I apply the quantum circuit that consists of StronglyEntanglingLayers that consists of a rotation along the 3 axes for each qubit followed by layer control not to create entanglement between the various qubits.

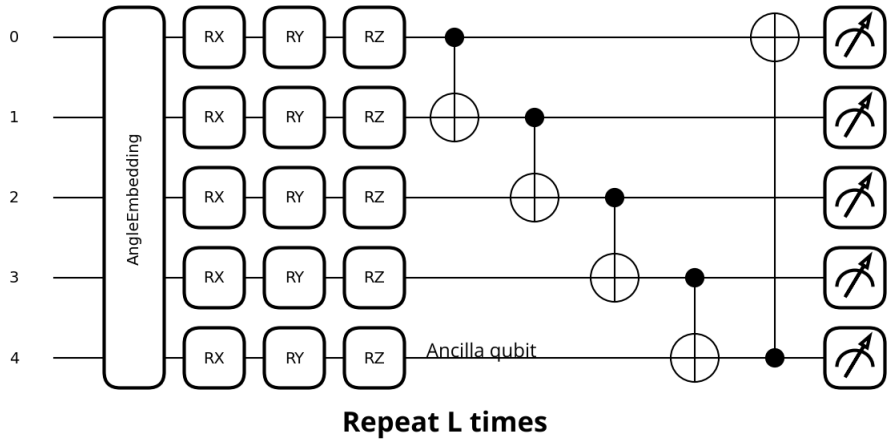


Figure 15: quantum gan circuit: composed of an angle embedding for encoding and followed by L layers of rotations around the 3 axes.

In this case, I chose to use 5 qubits of which one is an ancilla, 4 generators and 6 layers (by layers I mean repeating the same circuit L times), so each generator returns as output a tensor of batch size x 16 elements that put together go to reconstruct the 8x8 image. In this case, it is important to focus not only on the use of sub generators to avoid the noise problem due to the use of the few qubits, as mentioned earlier, but also on the use of the ancilla qubit. What is this ancilla qubit used for? The answer is to add non-linearity.

After applying the circuit $U_{G_t}(\theta)$ i get $|\Psi_t(z)\rangle = U_{G_t}(\theta)|z\rangle$. After measuring everything you take the partial measurement Π_A on the ancillary subsistem A $|\Psi_t(z)\rangle$. So the post measurement state is:

$$\rho_t(z) = \frac{\mathrm{Tr}_A(\Pi_A|\Psi_t(z)\rangle\langle\Psi_t(z)|)}{\mathrm{Tr}(\Pi_A \otimes I_{2^{N-N_A}}|\Psi_t(z)\rangle\langle\Psi_t(z)|)}$$

So the state obtained in this case is a non-linear map of $|z\rangle$ since both numerator and denominator contain a function of $|z\rangle$. Then the result of each generator is obtained by measuring $\rho_t(z)$ using a set of bases $\{|j\rangle\}_{j=0}^{2^{N-N_A}-1}$, then the result P(j) corresponds to $P(J=j) = \mathrm{Tr}(|j\rangle\langle j|\rho_t(z))$

where base $|j\rangle$ represents the j-th pixel value for the t-th sub-generator.

The result of the sub-generator system will be $G(z) = [G_1(z), \ldots, G_T(z)]$ where each sub-generator is defined as follows

$$G_t(z) = [P(J=0), \ldots, P(J=2^{N-N_A}-1)]$$

Discriminator The discriminator is classically implemented and is nothing more than a fully connected neural network (hence the classic multilayer perceptron seen before), which takes as input either the real data or the generated data G(z) and as output returns a single number the sigmoid activation function converts to a range between 0 and 1, where 1 stands for real i.e. D(x) and 0 stands for false D(G(z)). Thus, the more an input tends towards one of these two extremes, the more it is recognised by the network as true or false.

Loss function and optimization rule

The loss function is that of a classic gan where a min max game is played in which the discriminator tries to minimise its loss in distinguishing true from false examples while the generator tries to minimise its loss in generating examples similar to the true ones.

$$\min_{\theta}\max_{\gamma} L(D_\gamma(G_\theta(z)), D_\gamma(x)) := \mathbb{E}_{x\sim P_{\text{data}}(x)}[\log D_\gamma(x)]+\mathbb{E}_{z\sim P(z)}[\log(1-D_\gamma(G_\theta(z)))]$$

where $G_\theta(z) = [G_{\theta,1}(z), \ldots, G_{\theta,T}(z)]$. As an optimiser I used Stochastic Gradient Descent (SGD), as mentioned earlier, to optimise a neural network one subtracts for the gradient step $x \leftarrow x - \eta\nabla f(x)$: Given a dataset of n elements, a vector of parameters x and a loss function $f_i(x)$ with respect to the training example i I can express it as $f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$. At each iteration of stochastic gradient descent, we uniformly sample an index $i \in \{1, \ldots, n\}$ and compute $\nabla f_i(x)$ to update x $x \leftarrow x - \eta\nabla f_i(x)$

Training I trained the model on images scaled to 8x8 on a subset of 1 and 0 for 1150 iterations and these are some examples of images generated, in the image below we can see the trend of losses

Results From these examples, it can already be deduced with the naked eye that zeros are generated better than ones, and moreover, as interactions increase, images tend to generate only zeros, indicative of the problem gans often suffer from, namely mode collapse.

Figure 16: loss of the gan: the one in blue is the generator loss and the one in orange the discriminator loss

Figure 17: generated zeros

Figure 18: generated ones

To quantitatively evaluate this quantum gan, I decided to use a very popular index when it comes to measuring the accuracy of images generated by a gan or any other generative model. This process utilizes a pre-trained Inception v3 model, which is modified by removing its final layer. Instead of producing classifications, this altered model extracts abstract features from images, representing each image as a vector of 2,048 numbers. These numbers capture various aspects of the image as perceived by the Inception model. To calculate the FID score, both real and generated images are passed through this modified model. This results in two sets of feature vectors: one for the real images and another for the generated ones. These sets of vectors are then compared mathematically to determine how similar the generated images are to the real ones in terms of their abstract features. Essentially, the FID score uses the Inception model as an impartial judge to measure how closely the characteristics of generated images match those of real images, providing a quantitative measure of the generative model's performance.

INTERACTIONS	FID SCORE
500	199,42
1000	188,36

Table 2: Fid score after 500 and 1000 interactions

Obviously, as it is a measure of distance between the two distributions, the lower this value, the more similar the two distributions are and thus the images generated are likely to be

4 Quantum Diffusion models

The main topic of this work is the development of a quantum diffusion model, i.e. combining state-of-the-art generative models with quantum computation. As already seen quantum gans suffer from the same problems as classical gans such as unstable training which can be seen from the distribution of loss and mode collapse. For this it might be useful to exploit the ability of quantum models to generate probability distributions by combining this with the concept of noising and denoising of diffusion models.

The main advantage of using quantum technology is to limit the execution time of a diffusion model, which is the main problem when it comes to these models. In addition, quantum models can exploit unique properties to perform tasks that are impossible for classical computers and in this specific case analyse probability spaces inaccessible to classical computation.

However, in this work I have decided to direct my focus exclusively on classical problems such as simple image generation, to see if it is possible to find a suitable strategy to implement a quantum diffusion model that performs well in this task and that can perhaps be reused as a basis for solving more complex problems, particularly when dealing with quantum data.

In this case, since we are dealing with classical problems that have already found various solutions with more and more specific models and increasingly innovative solutions, I do not expect a quantum model with limited resources, and bearing in mind that this is one of the first approaches to the problem, to be able to match and surpass a classical model. However, it can be very useful today to be able to integrate a quantum model into a classical model that is already performing well, and as will be seen below, combining the two models can yield interesting results.

In this chapter I will give an introduction of how diffusion models are implemented in the state of the art, then I will give a theoretical overview of how the first quantum diffusion models were developed, and finally I will outline my own work and the choices that led me to make certain implementation choices.

4.1 Classical diffusion models and U-net

As seen above, a diffusion model consists of two steps:

- forward process: adding gaussian noise for each timestep
- backward process: removing gaussian noise for each timestep

In the backward process, a neural network is trained to predict the added noise and the loss between the predicted noise and the real one is calculated. Obviously the choice of the type of model is fundamental and recent studies have confirmed that the best performing model found so far for this type of task is the u-net. The u-net is a network used for image segmentation tasks, i.e. dividing images into portions of pixels that are easier to analyze and is mainly made up of convolutional layers. Convolutional layers are used for example in classification tasks in which it is necessary to predict which class an image belongs to, but when dealing with problems in which It is interesting to understand which class the individual pixels belong to, as for example in the analysis of biomedical images, networks that perform segmentation are used.

This type of architecture performs two types of operations, first downsampling and then upsampling. During downsampling, convolutions are applied by reducing the feature map and increasing the number of channels followed by ReLU activation functions, other operations such as batch normalisation and a maxpooling layer that takes the maximum value in each region of the image. We then take the resulting tensor with a very small feature map and many channels and perform the opposite upsampling operation that uses transposed convolutions by reducing the number of channels and increasing the feature map to regain the original image size. The latter process is accompanied by the addition of skip-connections that bypass one or more layers of a neural network, directly linking the input of one layer to the output of a subsequent layer, which serve to mitigate the problem of vanishing gradient in very deep networks.

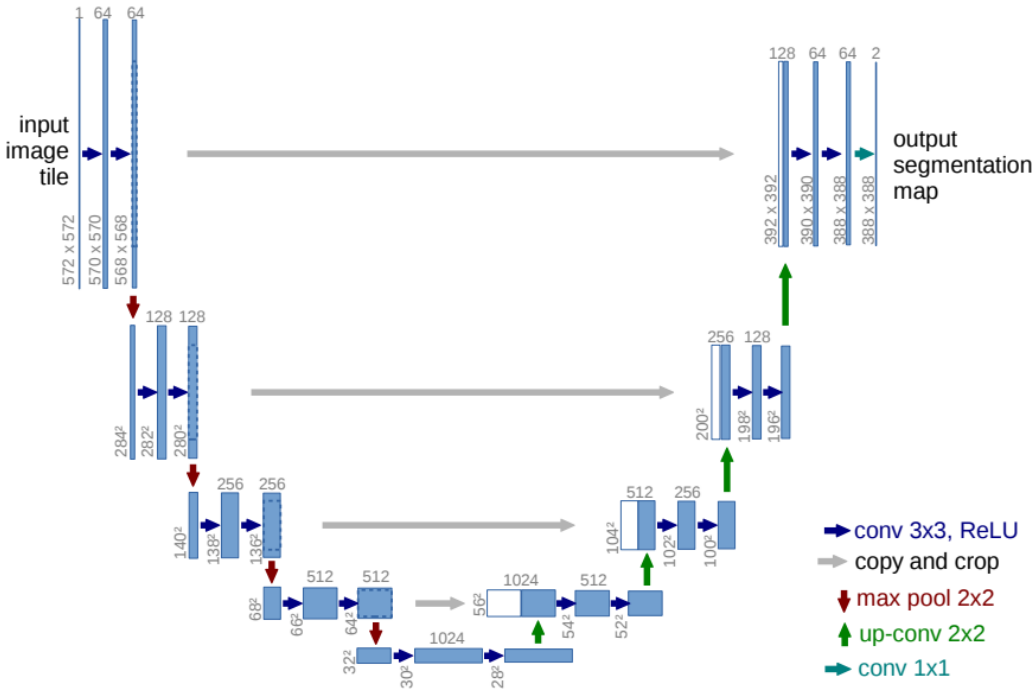


Figure 19: u-net architecture [17].

Without going too deep and risking going off-topic, suffice it to say that due to the complexity of the network and thus the quantity of parameters, it is impossible to find a quantum counterpart to date. So the aim of this work is not to find a model capable of competing with the classical state of the art, but to create a model that can offer interesting insights for future work when the NISQ era reaches a more prolific phase of quantum computation.

4.2 Quantum diffusion models: theoretical outlines

The advantages of quantum computation over classical computation have already been discussed in the literature. Obviously, first and foremost, quantum speedup is a big advantage, and secondly, probability distributions can be generated that are impossible to generate with classical computation.

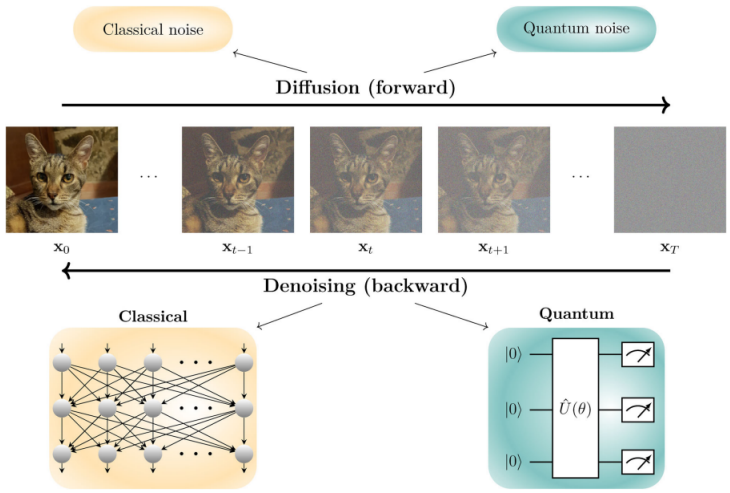


Figure 20: quantum diffusion model implementations: it is possible to use both classical and quantum noise and both classical and quantum denoisng models [18].

As already explained, diffusion models to date represent the state of the art in generative tasks, however they suffer from a problem of computational slowness that makes them disadvantageous in this respect to GANs. This is why one might consider using quantum noise as an advantage here.

The primary challenge in developing quantum processors that outperform classical computers is managing noise. Recent studies suggest that excessive noise can actually render quantum computations classically simulable, potentially negating their advantages. However, an intriguing question arises: could

this noise, typically seen as detrimental, actually benefit certain machine learning tasks?

Despite the generally harmful effects of quantum noise, both theoretical and practical research has shown that it can enhance information transport efficiency in some cases. Surprisingly, noisy quantum systems can sometimes evolve more rapidly than their noise-free counterparts. This phenomenon opens up interesting possibilities.

For instance, quantum noise might enable the creation of more intricate probability distributions, enriched by quantum entanglement. These distributions could be significantly more complex than those achievable through classical means. As a result, sampling from these distributions using a quantum processor might be far more efficient than attempts using even the most powerful classical supercomputers.

In essence, while quantum noise presents significant challenges, it may also offer unique opportunities for certain computational tasks, particularly in the realm of machine learning and probabilistic modeling.

On a practical level, three different versions of a quantum diffusion model can be implemented:

1. **Classical - Quantum diffusion model**: diffusion model where the noise is classical while denoising is done by a quantum model
2. **Quantum - Classical diffusion model**: diffusion model where the noise is quantum while denoising is done by a classical model
3. **Quantum - Quantum diffusion model**: diffusion model where the noise is quantum while denoising is done by a quantum model

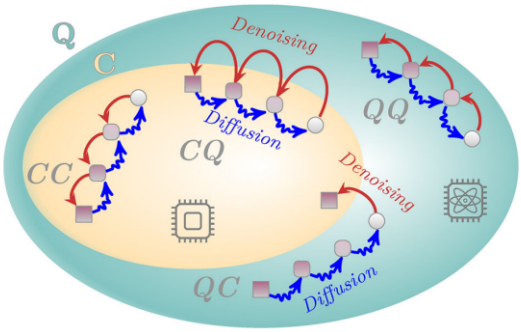


Figure 21: Probability distribution of various types of quantum diffusion models [18].

The figure shows how a quantum model can help explore probability distribution spaces other than classical ones: the two sets C and Q cover the classical space of probability distributions and the quantum space, respectively. As can be guessed, CC stands for classical noising and denoising just as QQ stands for quantum noising and denoising. CQ stands for classical diffusion and quantum denoising and are forced to work only with classical probabilities, but during the denoising phase they can exploit quantum properties within each step, while QC stands for quantum diffusion and classical denoising can manipulate quantum probabilities during the forward, but in that case, it is not possible to train the classical backward to map those probability distributions.

In the following chapters we will discuss the practical implementation of these models.

4.3 Classical Diffusion - Quantum Model

Before talking about my work, I would like to introduce the work of Marco Parigi, Stefano Martina and Filippo Caruso [18] which consists of applying a classical diffusion model and a straight line so as to destroy it and then reconstruct it by applying a quantum model

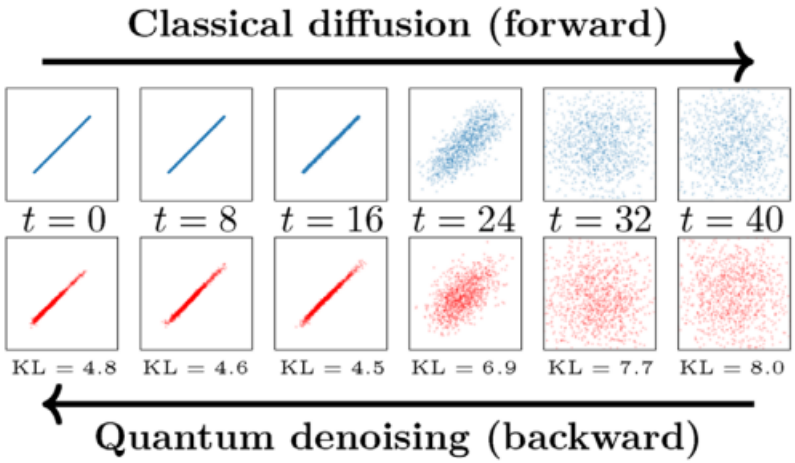


Figure 22: Destruction of a straight line in the forward direction and application of quantum denoising in the backward direction [18].

In this experiment, a set of points on a 2D straight line was generated in an interval [-1, 1]. During the forward phase, the Markovian process already seen in classical diffusion models was applied, i.e. adding Gaussian noise for 40 timesteps, while for the backward phase, a quantum circuit was implemented to reverse the forward process. The circuit consists of two parts, head and tail: the head consists of 4 qubits and uses angle embedding to encode classical information in a quantum state and uses 256 parametric rotations along the 3 axes alternated by control not gates. The ooutput is measured and passed to the tail which is a copy of the head with the only difference being that in the first

circuit the parameters are shared while in this second circuit the parameters are specific to each timestep. The PQC (Parametrized quantum circuit) is used to predict the mean $\mu_\theta(x_t, t)$ and variance $\Sigma_\theta(x_t, t)$ of the desired distribution in addition the model was trained for 40000 epochs on random batches of 1000 points to minimise the Kullback-Leibler divergence between the predicted and desired Gaussian distributions using Adam as an optimiser

The Kullback-Leibler divergence is a measure of distance between two probability distributions and is expressed in this form

$$D_{KL}(P||Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right),$$

where P and Q are two distributions and the divergence D_{KL} measures how far the probability of P(x), of an event x according to a probability, differs from a probability Q(x), of an event x according to a probability Q. The formula $\log(\frac{P(x)}{Q(x)})$ expresses the difference in probability in logarithmic terms, while the summation emphasises that the divergence is calculated for all events $x \in X$

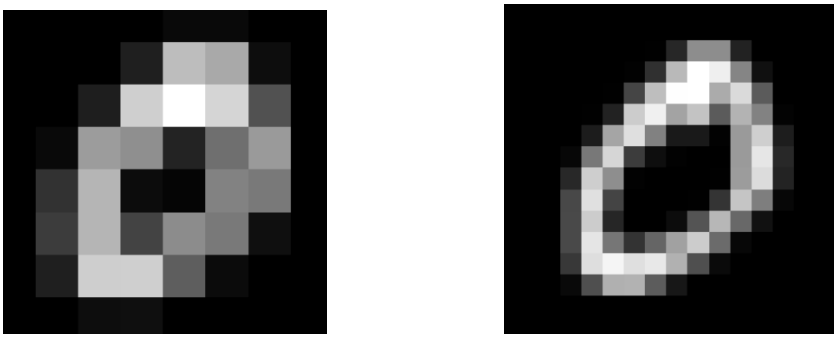
4.4 My implementation

For my implementation I decided to use a quantum diffusion model implemented with different techniques than those used in the paper, in terms of the loss function and the dataset I worked on.

In the paper, an attempt was made to reconstruct a two-dimensional line with a diffusion model composed of a classical forward and quantum backward process. In particular, the Kullback-Leibler divergence between the predicted and desired Gaussian distributions. In this case, however, I decided to work on a more complex problem, such as image generation using the MNIST dataset already used for quantum GANs. As a loss function I used the infidelity loss to minimise the distance between the generated quantum state and the expected quantum state.

I will now go into detail regarding this type of implementation and the advantages I have found in using this method.

Dataset As already mentioned, I used MNIST so that I could make a comparison with the images generated by the quantum GAN I implemented earlier. I tried both working with images scaled to 8x8 and with images scaled to 16x16. Obviously the images have been reduced in resolution due to the limited number of qubits that can be used



(a) (b)

Figure 23: Images downscaled to 8x8 and 16x16.

Forward step As a forward step I decided to add Gaussian noise for t timestep as in the classical implementation. I experimented with different timesteps to see how well the model was able to reconstruct the image with different degrees of noise. Obviously, the higher the noise, the more the model tends towards an isotropic Gaussian, which makes the task of reconstructing the image much more difficult.

Backward step For the backward process, I use a different quantum model from the one implemented in the paper. The main differences lie in the embedding method and the number of layers:

- As an embedding method, I used amplitude embedding, which allows information to be encoded as amplitudes of a quantum state. The advantage of this method is that with n qubits I can encode images of size 2^n. Amplitude encoding is optimal for this type of problem since I can encode an image of size 8x8 = 64 using only 6 qubits since $2^6 = 64$, or a 16x16 image with 8 qubits.
- For the circuit, I used 25 layers of parametric rotations along the three axes, known in the literature as strongly entangling layers. In this case, I used two circuits of 25 layers. The first consists of 7 qubits of which one is ancilla and the second of 6 qubits.The two circuits are the same but the former uses an ancilla qubit to add the non-linearity component already seen in the quantum GAN implementation

Loss and optimizer As an optimiser I always used Adam but as a loss I decided to change. As an optimiser I always used Adam but as a loss I decided to change. Specifically I used to use the infidelity loss i.e. 1 - fidelity. The quantum fidelity measures the distance between two quantum states using the formula:

$$F(\rho, \sigma) = \left(\mathrm{Tr} \sqrt{\sqrt{\rho} \sigma \sqrt{\rho}} \right)^2$$

ρ and σ are two quantum states converted into density matrices. In fact, what I do is take the image, apply all the diffusion steps and convert the image into the corrupted version thus obtaining x_t. Next I move to the x_t model obtaining x_{prev}. Then I obtain the image dirtied at the previous timestep and transform it into a quantum state, so I have x_{target} and finally I calculate the loss between x_{prev} and x_{target}. The two quantum states are converted into density matrices, so from an input (batch dimension,2^n) I get an input (batch dimension,2^n, 2^n), which are the density matrices ρ and σ on which I calculate the fidelity. Since fidelity measures how close two quantum states are, the larger this measure is, the more similar the two states are. For this reason what I do is to minimise the infidelity loss which would be 1 - fidelity, so

$$1 - F(\rho, \sigma) = \left(\mathrm{Tr} \sqrt{\sqrt{\rho} \sigma \sqrt{\rho}} \right)^2$$

So to sum up what I do is take the original image and dirty it until I get x_t which would be the completely dirty image. Next we try to clean up the image $\mathrm{PQC}|x_t\rangle = |x_{t-1}\rangle$ where PQC stands for parameterised quantum circuit, which would be the circuit I mentioned earlier. The circuit is thus applied for all timesteps t thus obtaining the original image x_0.

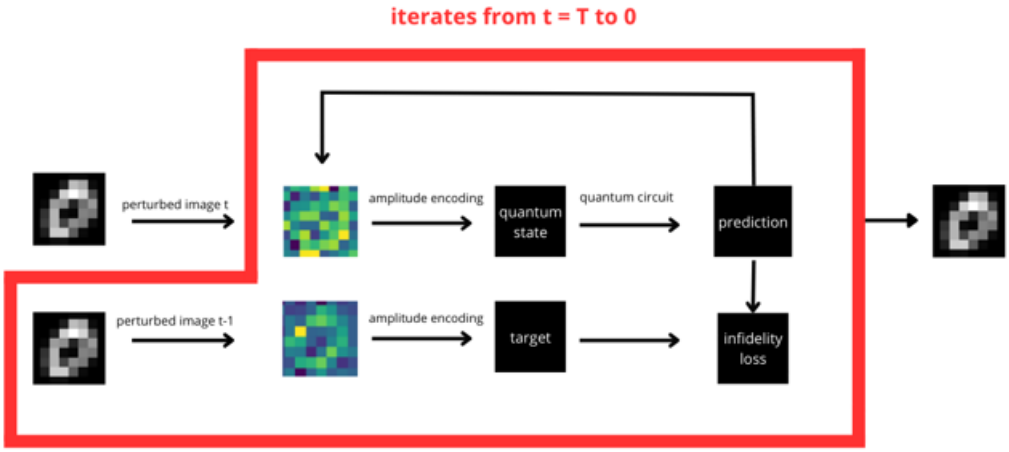


Figure 24: Training process of the quantum circuit: The image is soiled for t steps, then passed to the model to predict the image at timestep t-1. The predicted image is then passed to the model to predict the image at timestep t-2 and so on until trying to reconstruct the image at timestep t_0.

As you can see from the diagram I first take the dirtied image at time x_t and pass it to the model, then the resulting image becomes the one passed to the model in the next timestep and so on, then I apply PQC iteratively on all these generated images

In the next sections I will proceed by listing the results obtained with the different configurations.

4.4.1 Results

I have trained the quantum diffusion model on MNIST images scaled to 8x8 and 16x16 using both 10 and 40 diffusion steps so I will show the results of these configurations

- 8x8 images and 10 timesteps
- 16x16 images and 10 timesteps
- 8x8 images and 40 timesteps
- 16x16 images and 40 timesteps

Afterwards, I also tried integrating classical layers before the quantum model to see if there was any advantage in doing so.

It should also be pointed out that to date, using a quantum model that can compete with the best classical models is not an obvious challenge, which is why I decided to simplify the problem by reducing the number of diffusion timesteps and reducing the size of the images, thus also mitigating the computational problem since using a quantum simulator on a classical computer requires a lot of computational resources.

To evaluate the samples generated I obviously assessed the quality of the samples generated with the naked eye, which is obviously the most reliable metric, while at the quantitative level I used the FID score as in the case of quantum GANs.

8x8 images and 10 timesteps

In this case I trained the model for 5 epochs and as in the other cases I used infidelity loss and Adam as an optimiser with a learning rate of 0.001

The images generated are:

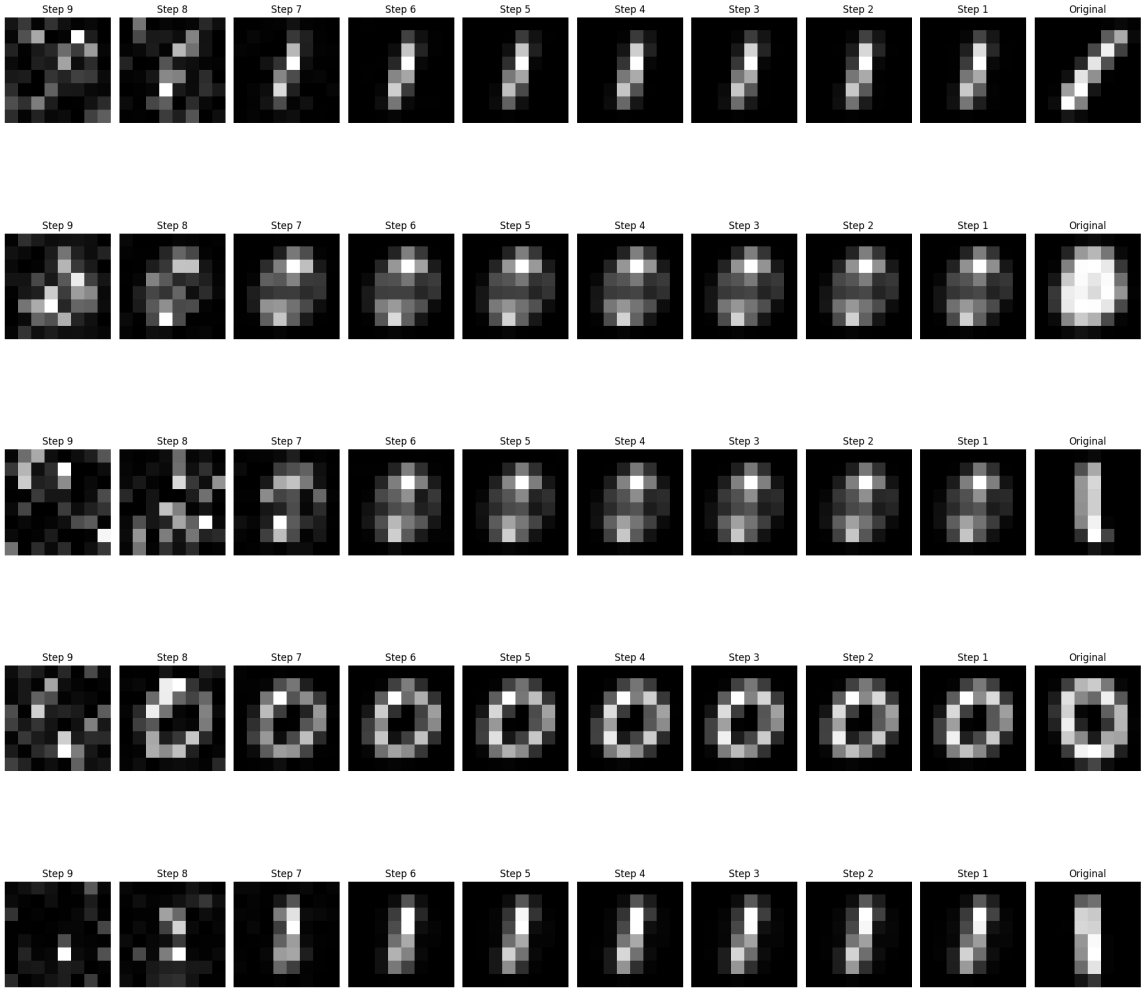


Figure 25: 8x8 images generated after 10 diffusion timesteps

8x8 images and 40 timesteps

As for the 8x8 images to which I applied 40 diffusion timesteps, the results are as expected somewhat worse than before where I applied 10 diffusion timesteps. In particular I notice that the model tends to get more confused when it has to generate the number 1

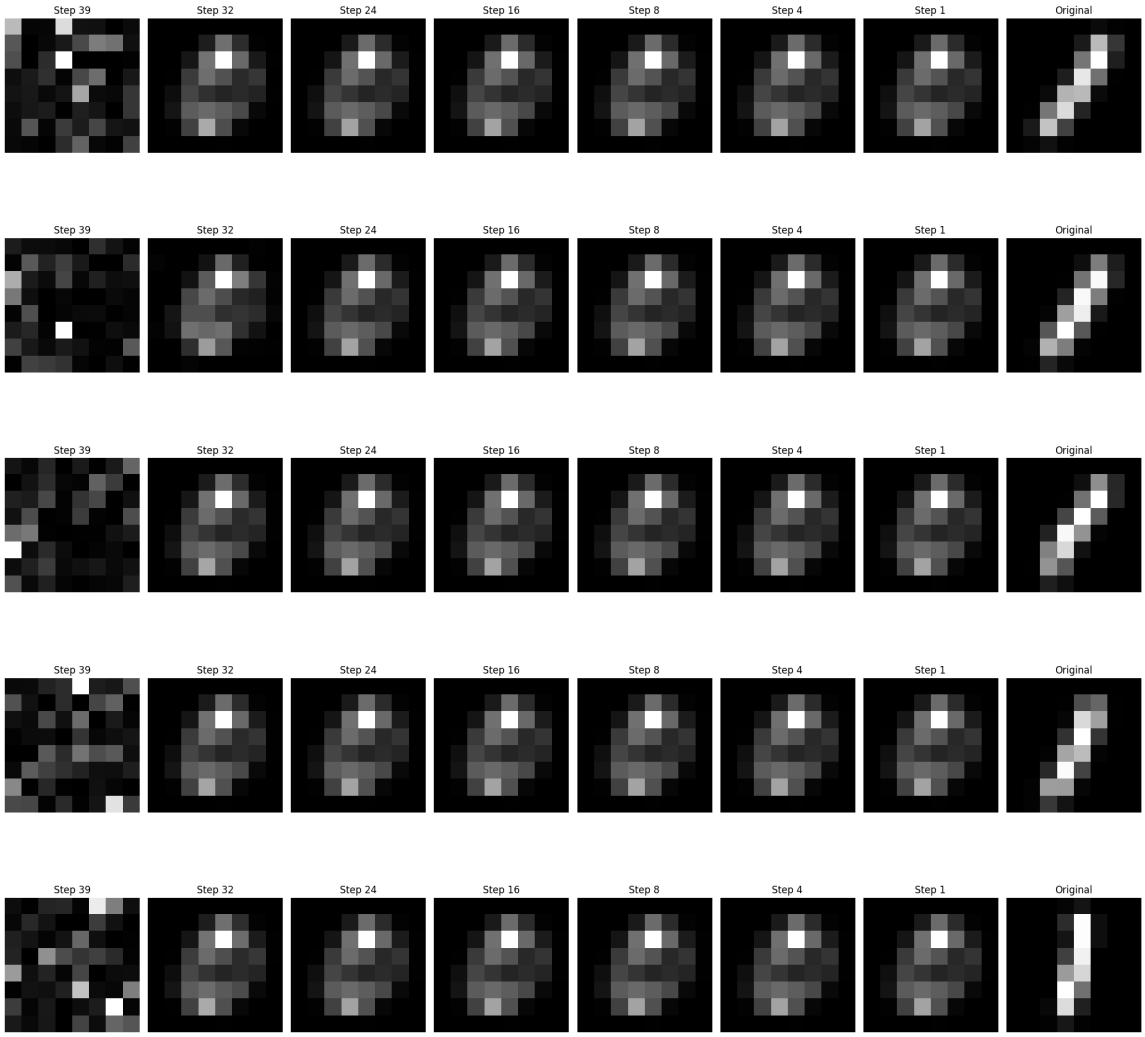


Figure 26: 8x8 images generated after 40 diffusion timesteps

16x16 images and 10 timesteps

I also tried to apply the quantum diffusion model on higher resolution images (16x16 to be precise). I used the same model as in the previous case trained for the same number of epochs, using the same loss and optimiser with the same hyperparameters

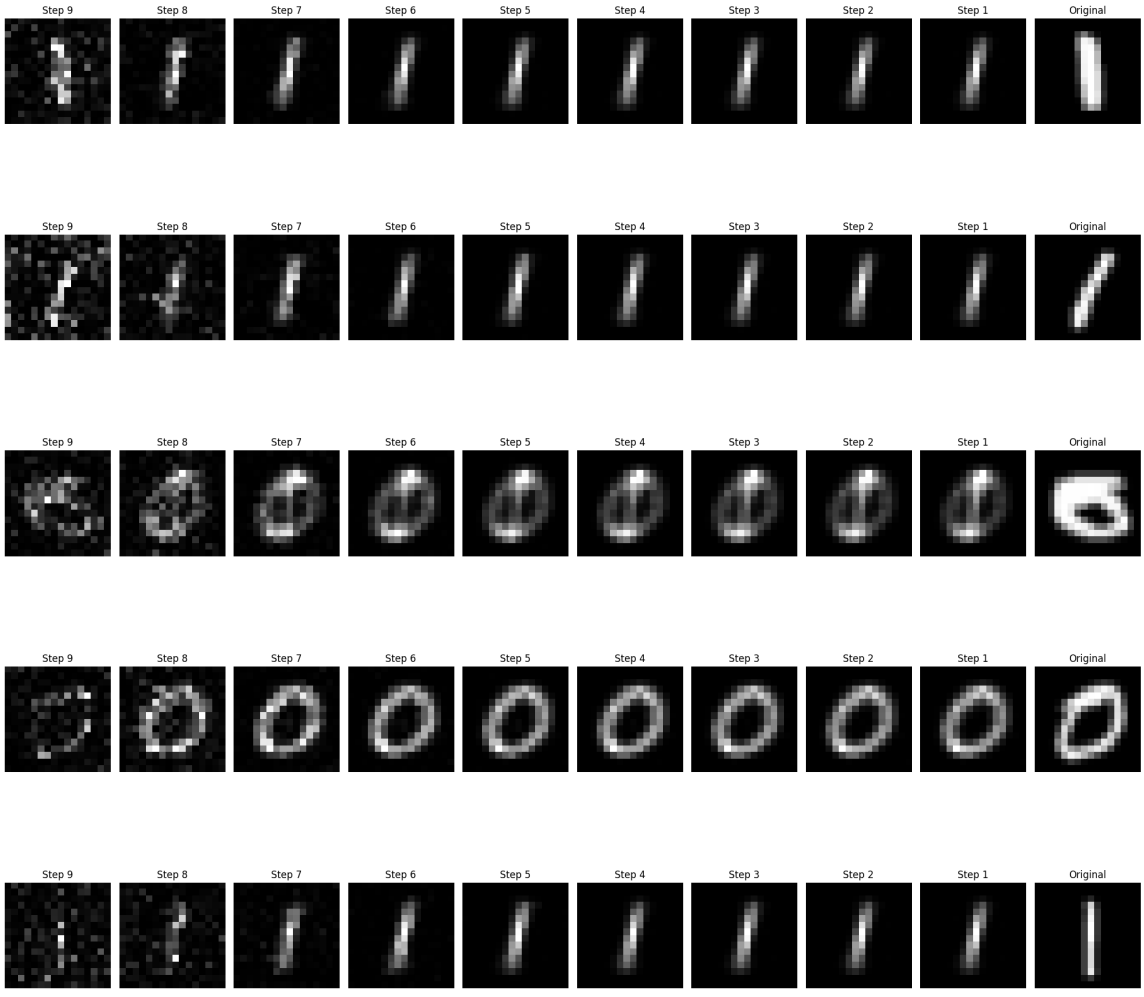


Figure 27: 16x16 images generated after 10 diffusion timesteps

16x16 images and 20 timesteps

I trained the PQC for 4 epochs, this time on 20 timesteps and still observed good results

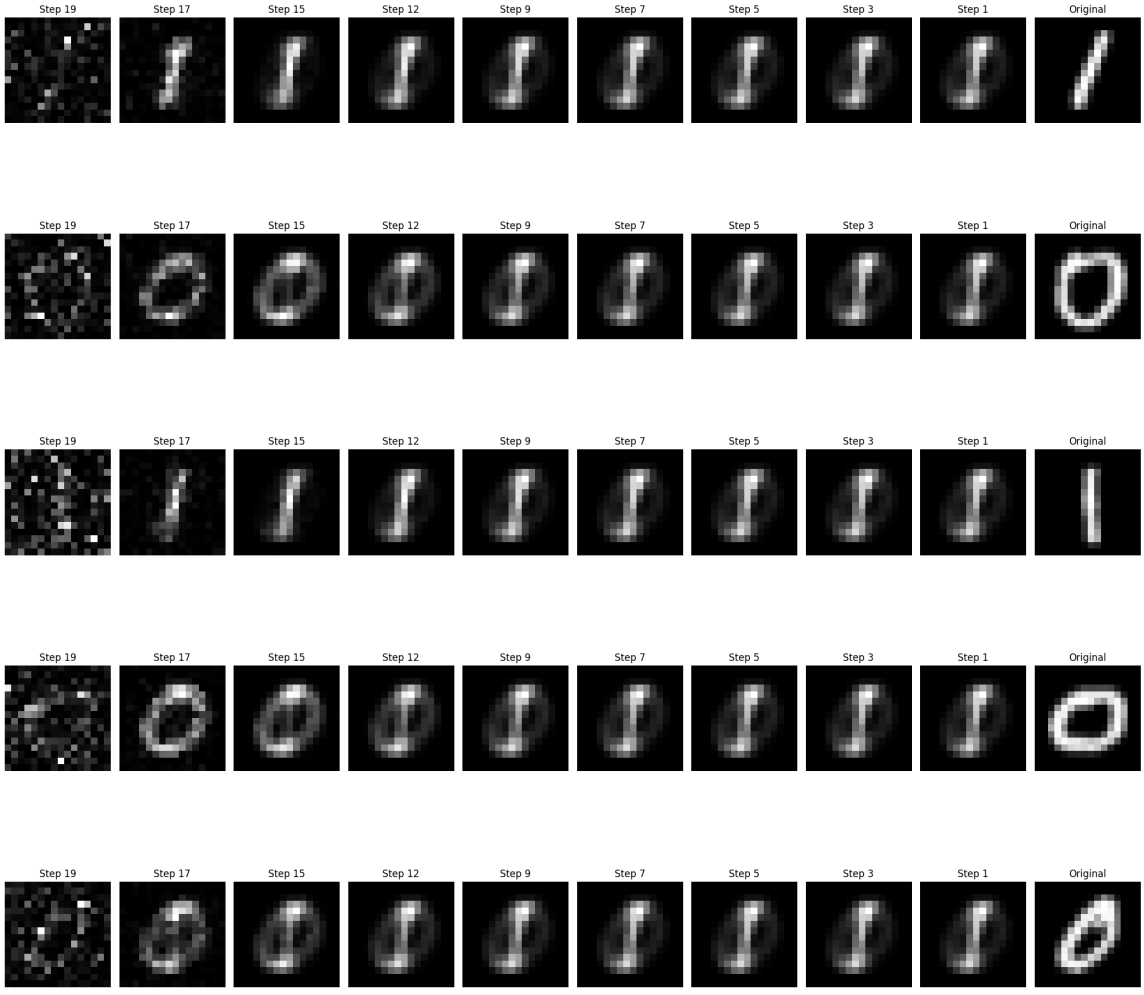


Figure 28: 16x16 images generated after 20 diffusion timesteps

Quantum classical hybrid model

As a final experiment, I tried using a classical, quantum hybrid model to see if there was an improvement. What I did in detail was to implement a simple autoencoder consisting of three linear layers and two relu activation functions that are placed between the first and second layers and between the second and third. Obviously this is a very simple network used only to see if it causes even a small improvement, so that it can be taken as a starting point for the realisation of hybrid circuits in the future.

The test was always done on 8x8 images after 10 timesteps of diffusion while the PQC remains the same as described above. The advantage of using a hybrid approach besides obviously the increase in parameters used is to combine the ability of the PQC to explore unexplored probability spaces combined with the non-linearity component added by a classical circuit with its activation functions

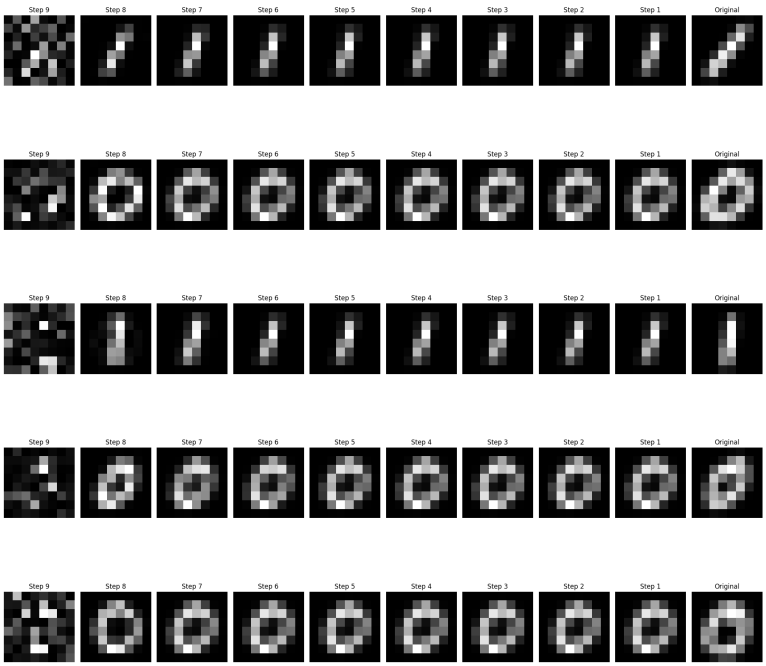


Figure 29: 8x8 images generated after 10 diffusion timesteps using hybrid model

4.4.2 Analysis of results

It's quite obvious and can also be observed how the application of multiple diffusion timesteps affects the quality of the samples, I decided to also use quantitative metrics to assess the quality of the generated images. As quantitative metrics I decided to make a comparison between the loss trend and use the FID score to measure the quality of the generated images I did in the case of the quantum GAN.

Trend of losses

As already emphasised, quantum fidelity measures the distance between two quantum states, the higher this value, the closer the two states will be and thus the images generated will resemble real ones. As a loss I decided to use quantum infidelity which would be 1 - fidelity. This value being the opposite of the previous one must be minimised to obtain images similar to the real ones. In my case, I decided to make comparisons of the losses between different cases including 8x8 images with different timesteps and 16x16 and 8x8 images with the same number of diffusion timesteps to see if the image resolution also complicated the work of the losses. Finally, I also made a comparison with the loss of the hybrid model.

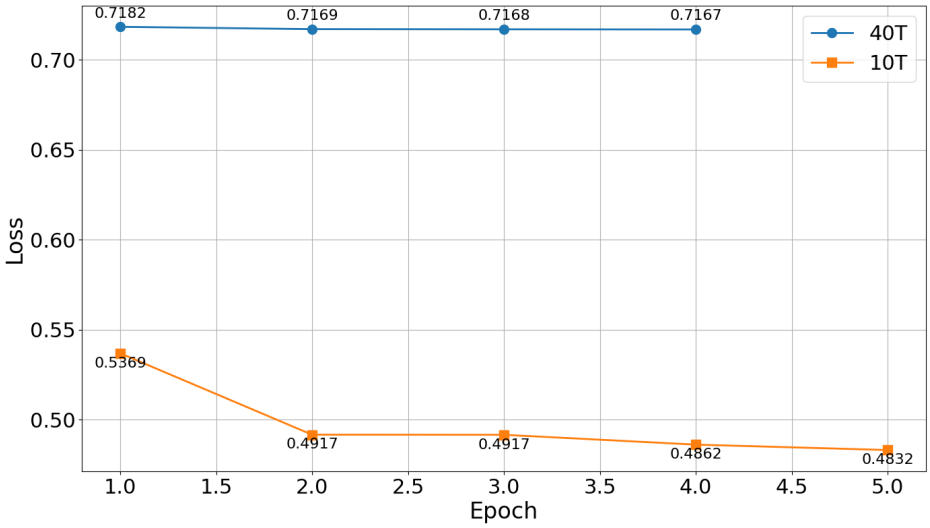


Figure 30: I have compared the losses in the case of 8x8 resolution images after applying 10 timesteps of diffusion and 40 timesteps respectively, and one can see how applying more noise steps can be expected to negatively affect performance, and not a little

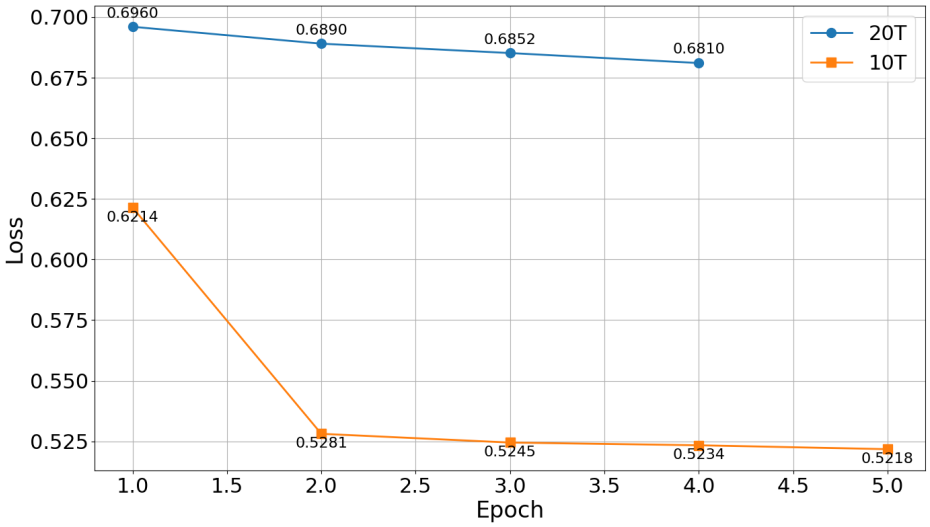


Figure 31: I did the same thing in the case of 16x16 images by comparing the application of 20 diffusion timesteps with 10 and even in this case even obviously less than in the previous case one can see a certain difference

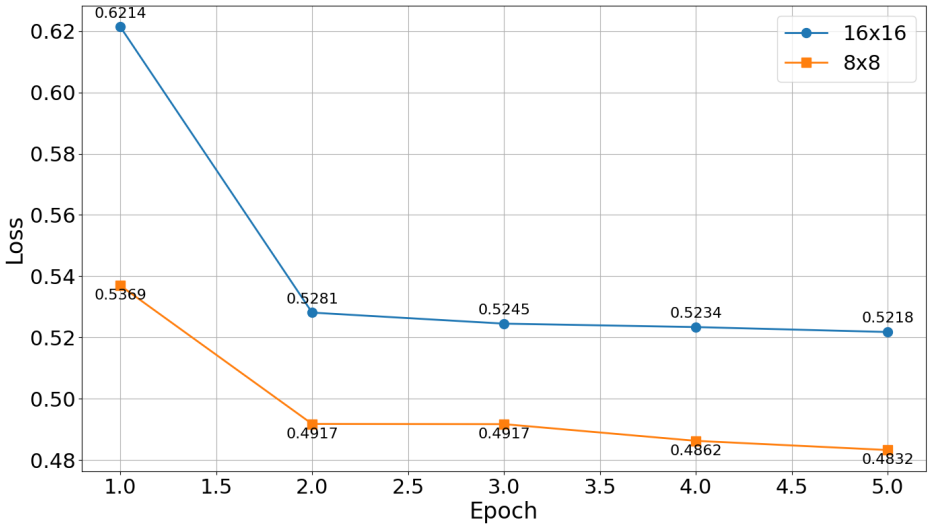


Figure 32: In this case, I compared the application of 8x8 and 16x16 images to see how much the difference in resolution affected performance. As can be seen from the graph, there is some drop in the case of higher resolution images, but not as sharp compared to the use of more diffusion timesteps

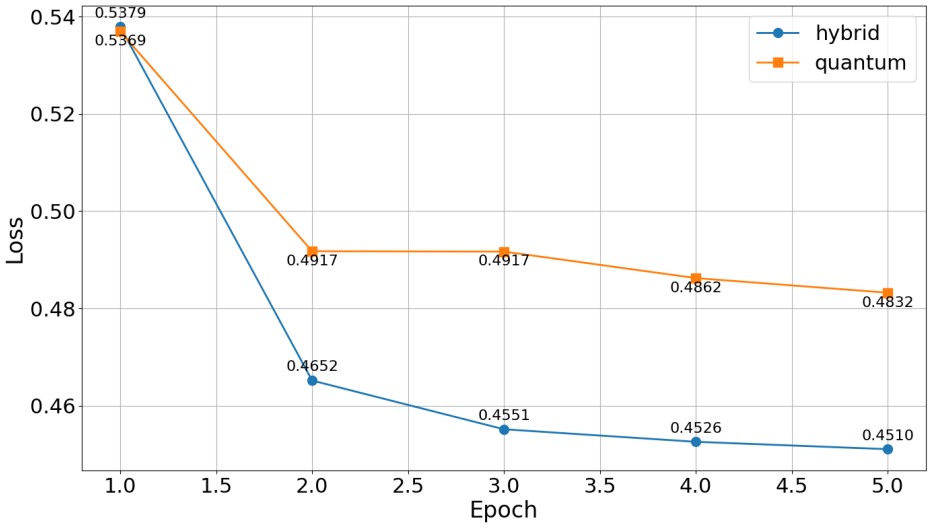


Figure 33: Comparison of the loss from the qauntistic model and the hybrid model both applied on the same 8x8 image dataset for 10 timesteps. The hybrid model performs slightly better than the classic one

For the FID score, I decided to use a sample of 100 images of which 50 are zeros and 50 are ones, then I applied the diffusion model to these images and calculated the score between the generated images and the real ones. Here are the results

Model	**FID score**
8x8 10 timesteps	244.39
8x8 40 timesteps	346.49
16x16 10 timesteps	379.91
16x16 20 timesteps	414.29
Hybrid model	223.64

Table 3: FID scores for different models

These scores are quite consistent with what we have seen above, i.e. the loss trend and the visible quality of the generated images.

4.5 Quantum diffusion Classical denoising

It is also possible to do the reverse of what has been done so far, i.e. to use quantum noise and a classical model to do denoising, and this is useful when dealing with quantum data. To apply quantum noise to data, one can use a quantum Markov chain, i.e. a process in which the state of a system depends only on the immediately preceding state. On a practical level this means that if I apply a noise at each timestep to an image, for example, as we saw in the previous example, the result of the image depends directly only on the image soiled at the previous timestep and the noise t. In essence this is what we have seen so far only instead of Gaussian noise we apply quantum noise.

The question that remains open is how to implement this quantum noise in the forward phase?

A discrete Markov quantum chain can be realised by applying a depolarising quantum channel

$$\rho_{t+1} = (1-p)\rho_t + p\frac{I}{d} \quad \text{with} \quad p \in [0,1],\ t \in [0,T].$$

In this way, the (quantum) information encoded in the initial quantum state ρ_0 is progressively degraded until we reach the maximally mixed state $\rho_T \equiv \frac{I}{d}$. the loss of information on the quantum state can be quantified by the von Neumann entropy $S(\rho) = -\mathrm{tr}(\rho \log_2 \rho)$.

$$S(\rho) = \begin{cases} 0 & \text{for pure states } \rho = |\psi\rangle\langle\psi| \\ > 0 & \text{for mixed states } \rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|, \\ & \text{with } p_i \geq 0 \text{ and } \sum_i p_i = 1 \\ \log_2 d & \text{for maximally mixed state } \rho = \frac{I}{d} \end{cases}$$

Obviously, a classical model such as multilayer percerptron or a convolutional network is used for the denoising phase to predict $\hat{\rho}_t$ from the input at time t + 1 and so on until trying to obtain the state $\hat{\rho}_0$ from the initial maximally mixed state $\frac{I}{d}$.

The authors of Ref. [18] in their paper tried to perform this experiment on a system of 10 QCGDMs (Quantum generative diffusion models) on a single qubit system and set the number of timesteps to 5, T = 5. The network was able to reconstruct ρ_0 and The average reconstruction quantum fidelity on a sample of 100 different random states is equal to 0.997 ± 0.013.

4.6 Quantum Noise Diffusion Models

The same paper cited above also implemented the full quantum version of the diffusion model, i.e. where both the generated noise and the denoising process are quantum and quantum classical diffusion model where the forward part is quantum and the backward part is classic. This process can lead to a purely quantum prior distributions that can be processed during the denoising phase with PQCs obtaining a generation process that is not feasible classically. As in the previous case, a depolarising channel was used to degrade the initial state ρ_0 to the maximally mixed state ρ_T. The implementation of the model, in this case, always uses the trace out technique already seen in quantum gan and the implementation of quantum DM. They consider a single-qubit system coupled via a parameterized circuit to another single-qubit that acts as the environment. During training, at each time step t, the system is initialized with ρ_{t+1} and, after the environment tracing out, the state is $\hat{\rho}_t$. In generation, it is possible to obtain $\hat{\rho}_0$, starting from the maximally mixed state $\frac{I}{d}$, iteratively applying the denoising with the previous prediction as input.

In this section I will present my implementation of diffusion models in which the forward part is quantum, so we would be dealing with quantum noise and no longer Gaussian noise. The quantum noise applied is the one already exposed in the previous section, i.e. the application of a depolarisation channel

$$\rho_{t+1} = (1-p)\rho_t + p\frac{I}{d} \quad \text{with} \quad p \in [0,1],\ t \in [0,T]$$

For the experiment, I initially decided to take a completely pure state and degrade it to a completely mixed state and then try to recast it using both a classical and quantum network. Then I decided to extend the problem to the case of 8x8 and 16x16 images by making a comparison between a classical and a quantum model in the image reconstruction task. As can be seen in figure

[22] from a quantum noise I would expect it to lead the image or at least the generic input into a different probability state and it would be very interesting to see how a classical model deals with this new distribution and whether there are any real advantages to using a quantum model to do the denoising

4.6.1 Reconstruction of a pure state

For the first task, I decided to take a single qubit system, so as to take the simplest possible problem and see if this approach could have any potential. Initially, I generated a pure state, i.e. a state in which the trace of the corresponding density matrix is equal to 1. Mathematically, it can be expressed as follows: taken a state, $|\psi\rangle$ this state is said to be pure if the corresponding density matrix $\rho = \langle\psi\rangle$ has a trace equal to 1 $\mathrm{Tr}(\rho^2) = 1$, i.e. if the sum of the elements of its diagonal equals 1. Explained in simple terms, a pure state is one in which we have exact information about the state the system is in, whereas if the system is in a combination of states we speak of a mixed state and in this case we do not have exact information about the state of the system, which may be in one state with a certain probability or in another with another probability

After generating this pure state, I 'dirtied' it up to a fully mixed state $\frac{I}{d}$ where if d is the system size and n is the number of qubits $d = 2^n$. For depolarisation I used the library implemented by pennylane which performs a total depolarisation with $\rho = 0.75$ by taking the pure state and transforming it into the fully mixed state by destroying the quantum information for 30 timesteps applying $t/T * \rho$ at each step, con t = 1, 2, ..., 30 e T=30, so that the image would gradually lose information.

For denoising, I used both a classical and a quantum model, using in both cases Adam as the optimiser and the difference in absolute value between the generated and the desired state. The classical model uses 356 parameters while

the quantum model uses 150 and both have been trained for 10 epochs, assessing how far the purity of the generated state ultimately deviates from the pure state. To do this, I calculated the trace of the density matrix $\mathrm{Tr}(\rho^2)$ which starts at 0.5 for the maximally mixed state and slowly tries to reconstruct the information to get it to 1

	Classical model	**Quantum model**
Quantum purity	0.631	0.998

Table 4: Quantum purity for classical and quantum model: This table represents the purity generated by the classical and quantum models. Although the classical model has more parameters, it performs much worse than the quantum model in generating a completely pure state. As can be seen, the quantum model performs great. Given a density matrix ρ Quantum purity = $\mathrm{Tr}(\rho^2)$

Given the generated quantum state ρ the quantum purity is calculated via $\mathrm{Tr}(\rho^2)$ so the quantum purity defined above is obtained by trying to reconstruct the initial state and calculating the trace of the corresponding density matrix. As can be seen, the quantum model outperforms the classical one: given the completely pure state that corresponds to 1, the purity of the classical model is about 0.6 while the quantum one is almost 1

4.6.2 Experiments on the MNIST dataset

In this section I discuss the implementation of QCDMs and QQDMs for the MNIST dataset so in the case of images. For the forward part I have used the depolarisation channel explained above while for the backward part I have used first a classical network (QCDM) and then a quantum network (QQDM).

I decided to apply the models on a single image using Adam as the optimiser with lr=0.001 and mean squared error as the loss function

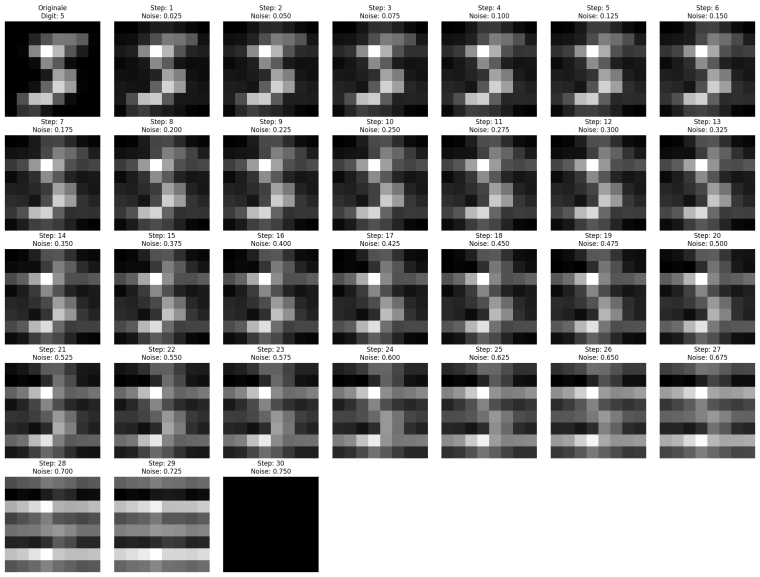


Figure 34: Representation of the quantum noising process, as can be seen, the image loses quantum information until it no longer has any

As a classical denoising model, I used two linear layers interspersed with a Relu activation function. Although it is a very simple model it uses more parameters than the quantum model, so it can be considered a good benchmark to see how promising quantum technology can be in such a task.

In detail, for quantum denoising I used the same architecture as for quantum diffusion models with Gaussian noise: so these circuits consist of a head and a tail whose head (in the case of 8*8 images) is made up of 7 qubits, one of which is ancillary to perform the trace-out, while the second circuit consists of 6 qubits. As an encoding strategy I have always adopted amplitude embedding so that I can encode the state in only 6 qubits, followed by an L number of strongly enrangling layers.Both models have been trained for 50 epochs

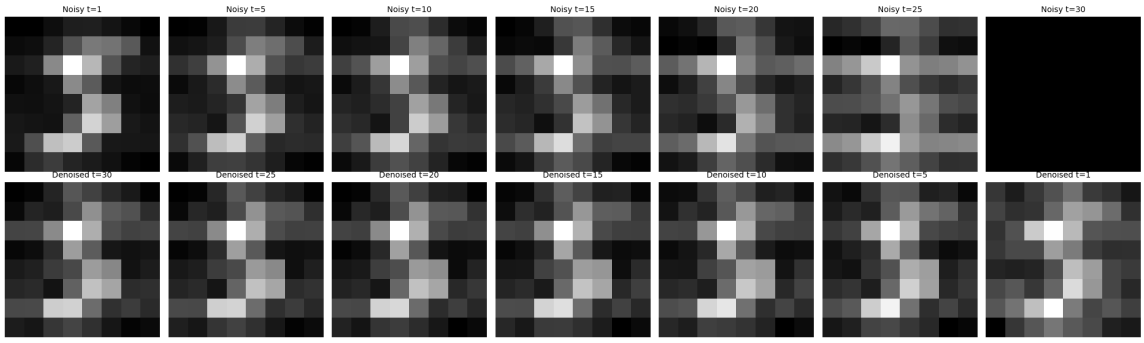


Figure 35: Classical denoising

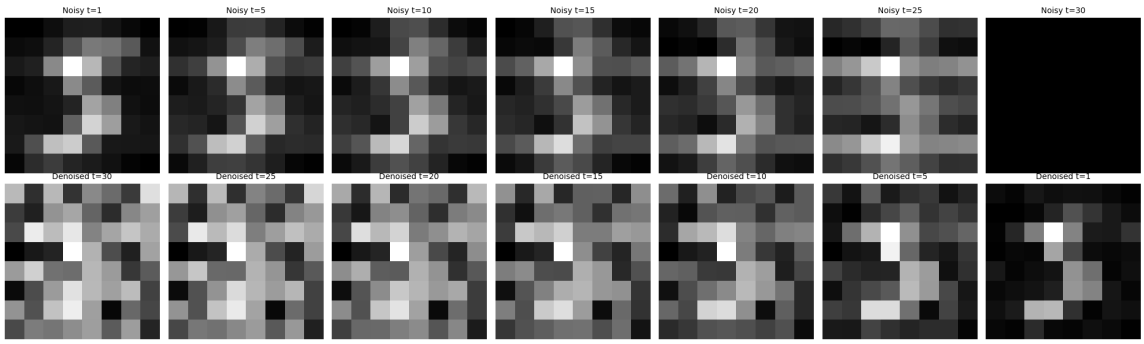


Figure 36: Quantum denoising

As can be seen from the images by eye, the quantum model seems to work better than the classical one despite the use of fewer parameters.

To prove this numerically as well, I used the fidelity score to calculate the closeness between the generated image and the original one:

	Classical model	**Quantum model**
Quantum fidelity	0.554265	0.901033

Table 5: The table shows the fidelity score for both models: as expected, fidelity in the quantum case outperforms fidelity in the classical case

5 Conclusions

In this work, I tried to implement generative models using quantum technology. I started by giving an introduction to classical machine learning and quantum machine learning, explaining the differences and similarities between the two.

Of course, we are still at the beginning of this discipline, i.e. in the NISQ era, and compared to its classical counterpart, we do not have any devices available that can surpass it, but we can already imagine how to implement quantum generative networks using what we have.

The dawn of this era has already seen the implementation of the first quantum GANs and the application of these. I have tried to take existing works, delve into the key points and apply them in my own way to implement my own version of a quantum GAN patch. Subsequently I tried to implement a quantum diffusion model and the bulk of my work is based on the latter.

I have developed several types of quantum diffusion models:

- CQDM: in the forward process the image is dirtied with Gaussian noise while in the backward process I use a quantum circuit
- QCDM: in the forward process the image is dirtied with a depolarizing channel while in the backward process I use a classical circuit
- QQDM: in the forward process the image is dirtied with a depolarizing channel while in the backward process I use a quantum circuit

The first version I applied it to the MNIST images on the digits 0 and 1 and verified that even with few resources it is possible to generate these digits with a good similarity to the real one but of course the capacity cannot be reached today given the computational capabilities we have classically achieved and also given the way they are implemented. Classical models are already optimised to work with classical noise and the probability space generated is optimal to be reconstructed from a classical model.

In order to put the spotlight on the possible quantum advantage, as well as the possible speed up, I decided to try a quantum forward process in order to shift the probability space into the quantum regime and see if this choice could pay off.

To do this I decided to apply a depolarization channel by treating the image as a quantum state and gradually removing quantum information until a maximally mixed state was obtained. I first decided to apply it on a single qubit system and then on a 6 qubit system, which corresponds to the MNIST image.

On both types of systems I observed a clear advantage of the quantum model compared to the classical one proving that when dealing with a quantum probability space a quantum circuit outperforms the classical one

Future works An interesting application of a full quantum model could be to deal with physics problems such as the motion of particles by treating the problem as a time series.

For example, in Ref [1] a similar thing has already been done by predicting the movement of particles in fluid turbulence using a classical diffusion model where the donosing model is a U-net. The model in this case works great in predicting the movement of particles, in the future it would be interesting to use a quantum model to see if it can equal or even surpass the results of the classical one.

References

[1] T. Li, L. Biferale , F. Bonaccorso, M. A. Scarpolini, M. Buzzicotti, "Synthetic Lagrangian turbulence by generative diffusion models"

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

[3] Jonathan Ho, Ajay Jain, Pieter Abbeel, "Denoising Diffusion Probabilistic Models"

[4] Osvaldo Simeone, " An Introduction to Quantum Machine Learning for Engineers"

[5] Maria Schuld, Francesco Petruccione, "Supervised Learning with Quantum Computers"

[6] ttps://learning.quantum.ibm.com/course/variational-algorithm-design/variational-algorithms

[7] John Preskill, "Quantum Computing in the NISQ era and beyond"

[8] Seth Lloyd, Christian Weedbrook, "Quantum Generative Adversarial Learning"

[9] Christa Zoufal, Aurélien Lucchi, Stefan Woerner, "Quantum Generative Adversarial Networks for learning and loading random distributions"

[10] Aston Zhang, Zachary C. Lipton, Mu Li, Alexander J. Smola, "Dive into Deep Learning"

[11] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath, "Generative Adversarial Networks An overview"

[12] Minati Rath, Hema Date, "Quantum Data Encoding: A Comparative Analysis of Classical-to-Quantum Mapping Techniques and Their Impact on Machine Learning Accuracy "

[13] He-Liang Huang, Yuxuan Du, Ming Gong, Youwei Zhao, Yulin Wu, Chaoyue Wang, Shaowei Li, Futian Liang, Jin Lin, Yu Xu, Rui Yang, Tongliang Liu, Min-Hsiu Hsieh, Hui Deng, Hao Rong, Cheng-Zhi Peng, Chao-Yang Lu, Yu-Ao Chen, Dacheng Tao, Xiaobo Zhu, Jian-Wei Pan, "Experimental Quantum Generative Adversarial Networks for Image Generation"

[14] https://pennylane.ai/qml/demos/tutorial_quantum_gans/

[15] https://datasets.activeloop.ai/docs/ml/datasets/mnist/

[16] Yu Yu, Weibin Zhang, Yun Deng, "Frechet Inception Distance (FID) for Evaluating GANs"

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation"

[18] Marco Parigi, Stefano Martina, Filippo Caruso, "Quantum-Noise-Driven Generative Diffusion Models"

[19] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, Surya Ganguli, "Deep Unsupervised Learning using Nonequilibrium Thermodynamics"

[20] Michael A. Nielsen, Isaac L. Chuang, "Quantum Computation and Quantum Information"

[21] Andrea Cacioppo, Lorenzo Colantonio, Simone Bordoni and Stefano Giagu, "Quantum Diffusion Models"

[22] Michael Kolle, Gerhard Stenzel, Jonas Stein, Sebastian Zielinski, Bjorn Ommer, Claudia Linnhoff-Popien, "Quantum Denoising Diffusion Models"

Appendix

A1 Ansatze analysis

In this section, I will present the choices regarding the quantum or ansatz circuits used, along with pictures of the circuits. Starting from the one used for quantum GAN to those used for quantum diffusion models. At the implementation level they are very much the same, the big difference lies in the number of layers used and the data embedding strategy.

In the case of the quantum gan I followed the guidelines of the reference paper and the existing code but modified it to work better with the pytorch library, while in the case of the quantum diffusion models I decided to start from work already done but implement the circuits in my own way and then conclude by implementing a circuit also to add quantum noise.

The following circuits have only one or two layers in order to make visualisation easier,

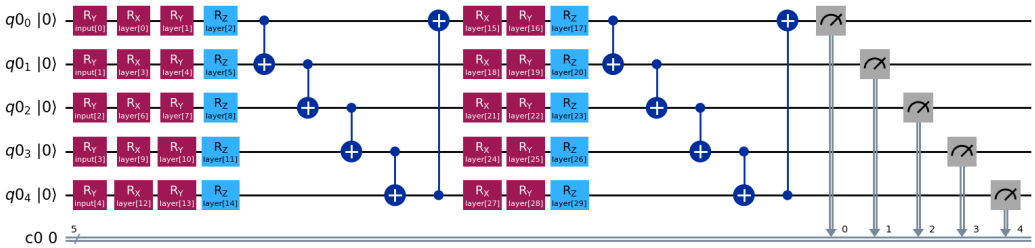


Figure 37: Ansatz used for quantum GAN: this circuit corresponds to a single two-layer generator. Obviously, it is a simplified version since the actual model uses four 4-layer generators.

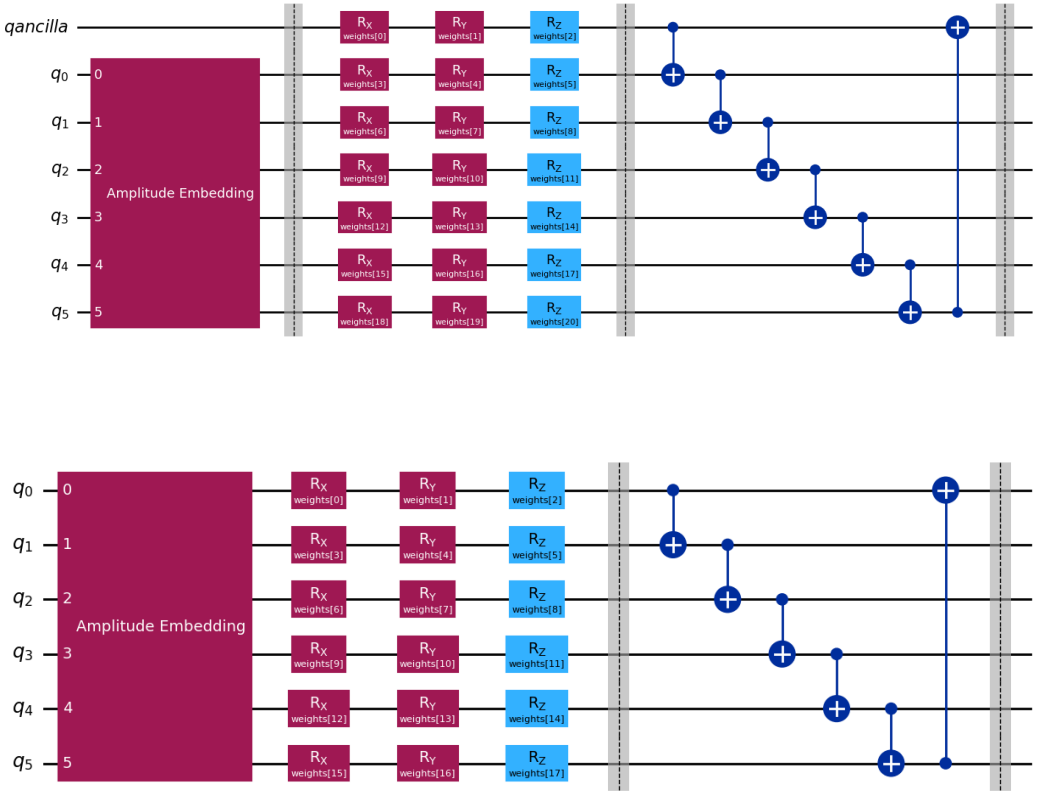


Figure 38: Ansatz used for quantum diffusion modelling on 8x8 images. The first is the circuit with the qbit ancilla, the second the one without.

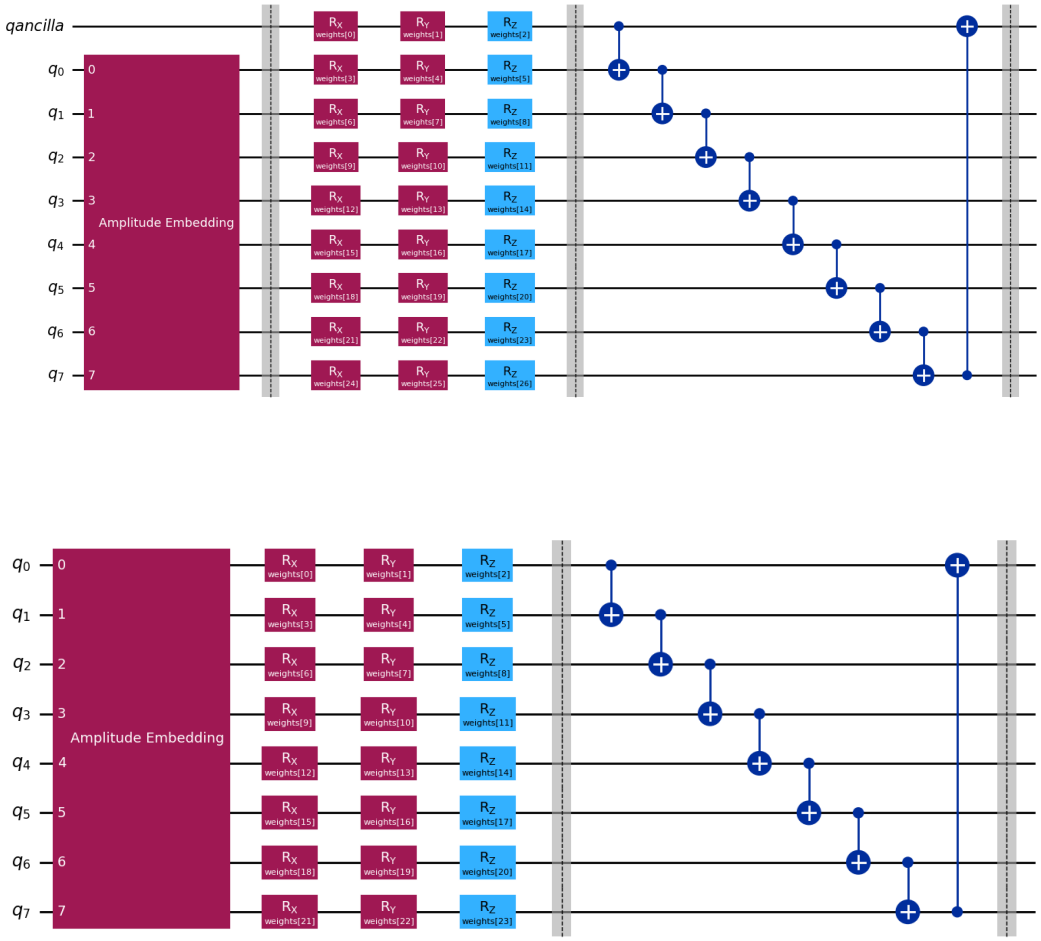


Figure 39: Ansatz used for quantum diffusion modelling on 16x16 images. The first is the circuit with the qbit ancilla, the second the one without.

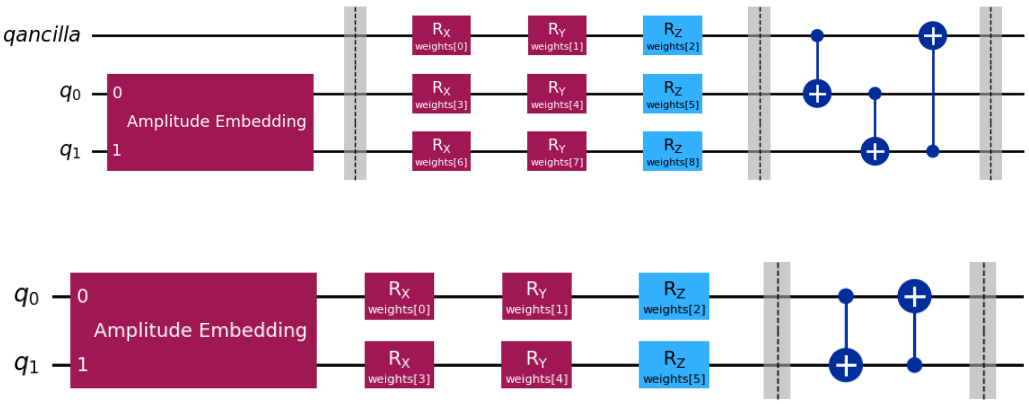


Figure 40: Circuit used for the full quantum denoising experiment on the single qubit system.

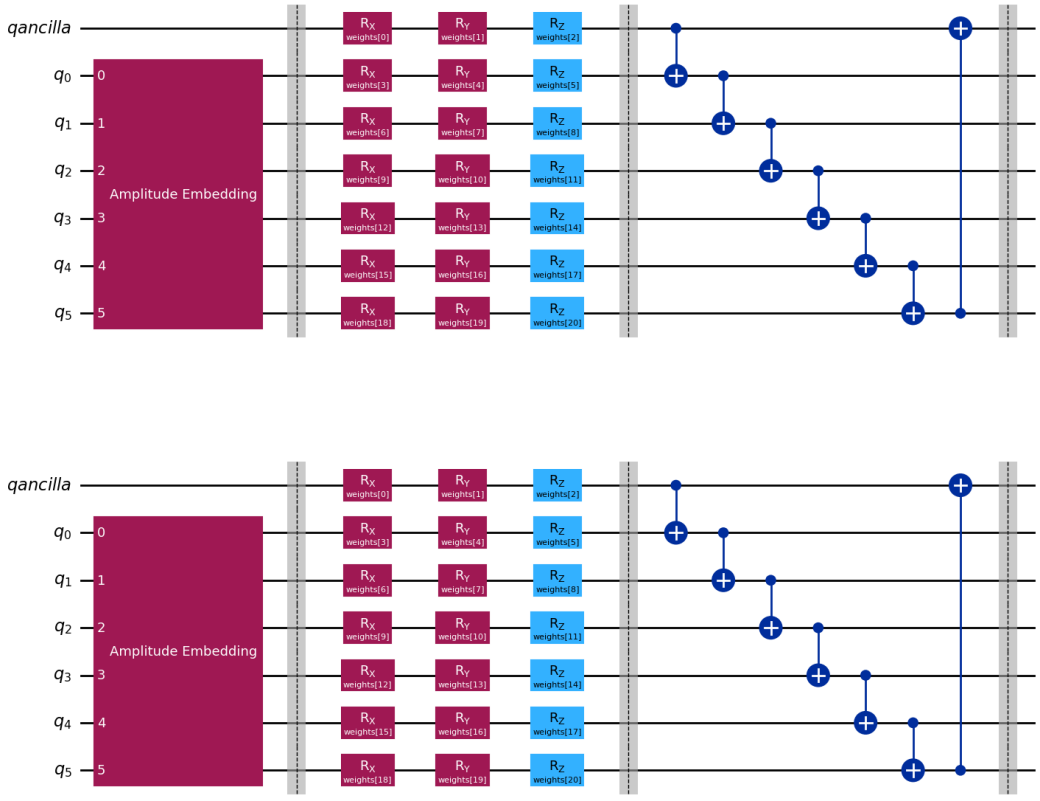


Figure 41: Circuit used for the full quantum denoising experiment on mnist image.

A2 Additional results

In this section, I will present a variety of results that illustrate the effectiveness of our models. To begin with, I will showcase results obtained from 16x16 images, providing a comprehensive overview of our findings. This will allow readers to appreciate the performance and capabilities of our models across different image sizes. Following that, I will delve into the results of training both classical and quantum models on an 8x8 image dataset. It is important to highlight that the forward process of our models is implemented in a quantum framework, leveraging the unique properties of quantum mechanics to enhance the generation and reconstruction of images. By comparing the outcomes of the classical and quantum approaches, we can better understand the advantages and nuances offered by the quantum-based methodology in this specific context. Through these analyses, I aim to provide a thorough understanding of how our models perform and the implications of using quantum processes in image generation.

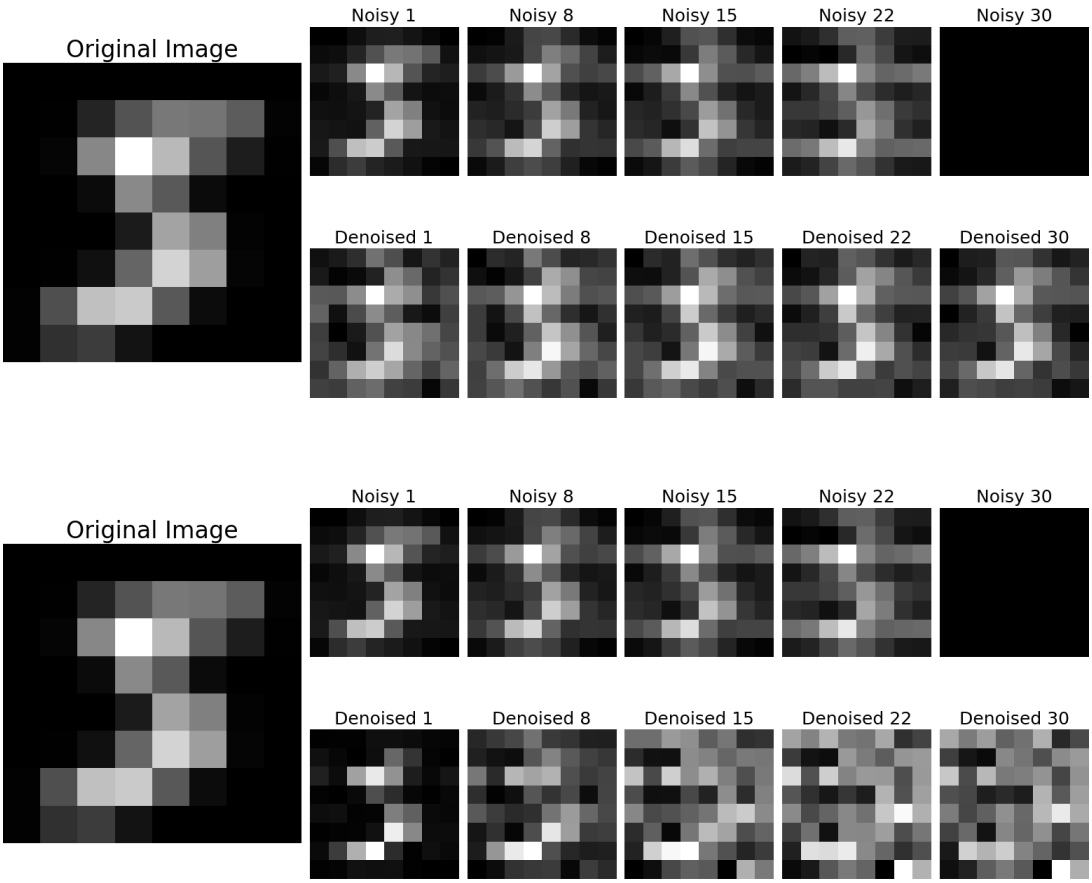


Figure 42: The first image refers to the denoising of number 5 of the classical circuit, while the second corresponds to the denoising of the same number done by a quantum circuit on the 8x8 image format.

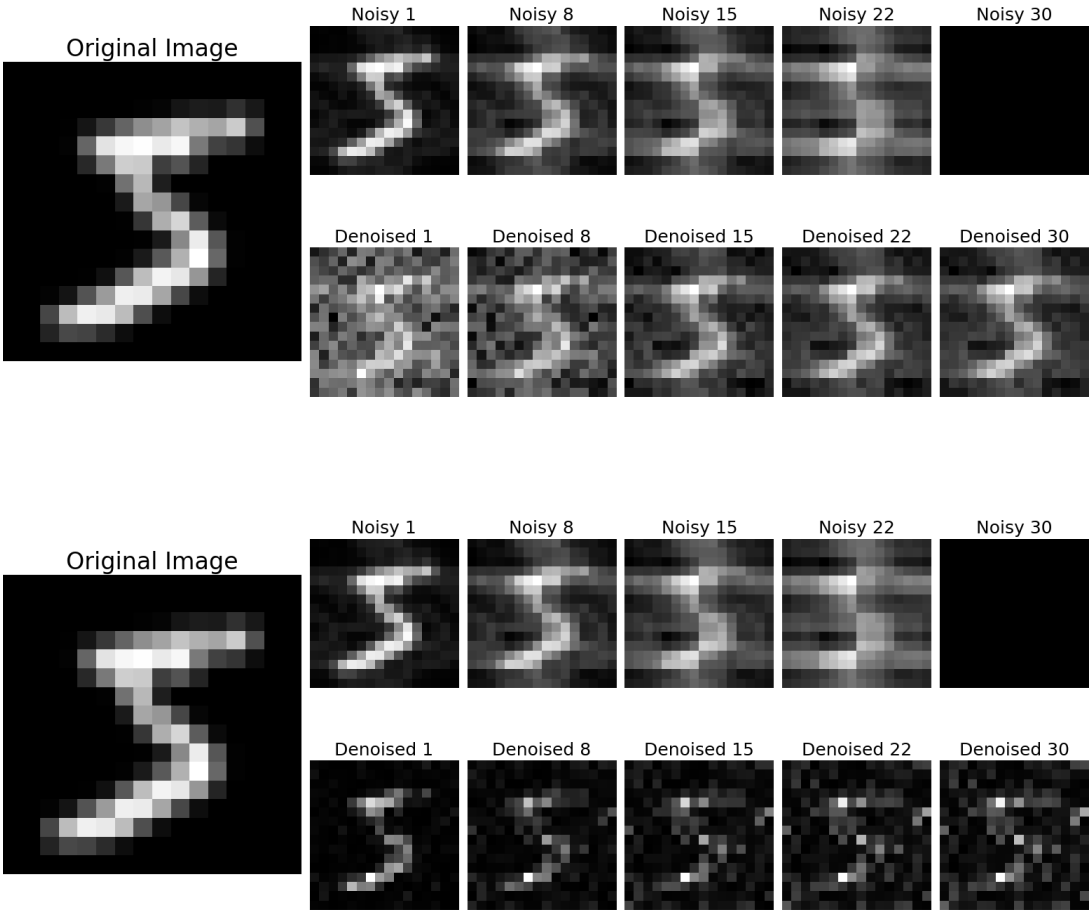


Figure 43: The first image refers to the denoising of number 5 of the classical circuit, while the second corresponds to the denoising of the same number done by a quantum circuit on the 8x8 image format.

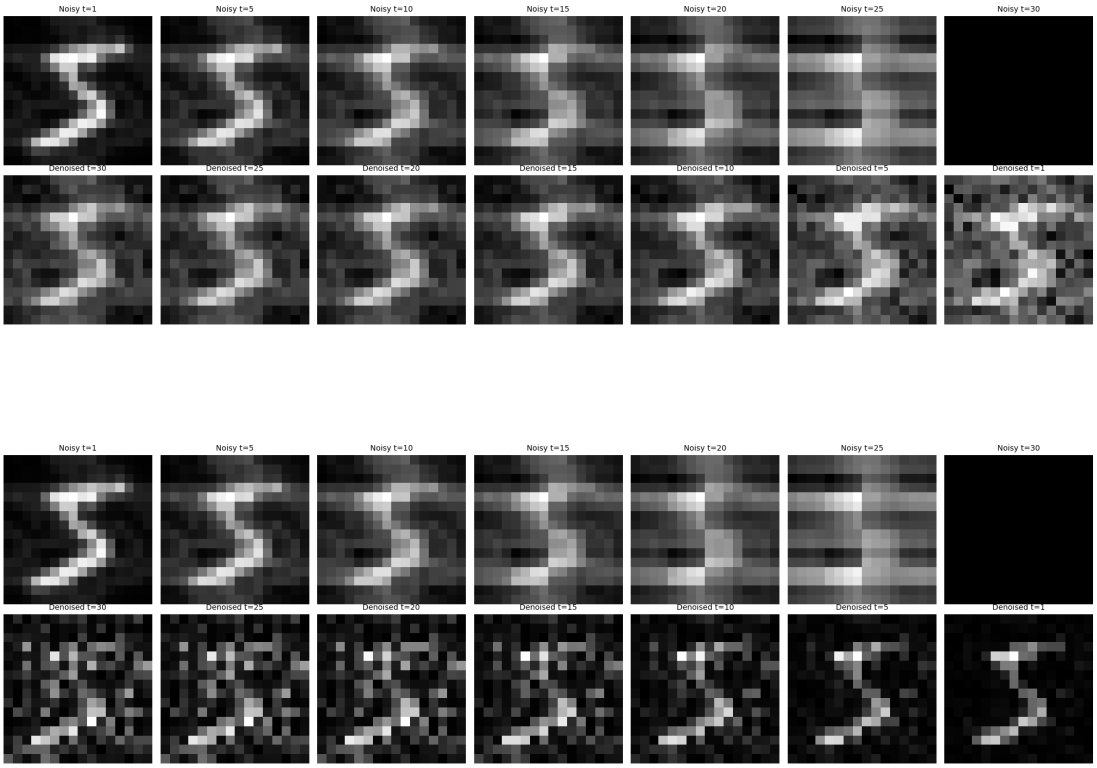


Figure 44: The first image refers to the denoising process of number 5 of the classical circuit while the second to the denoising process of the quantum circuit.

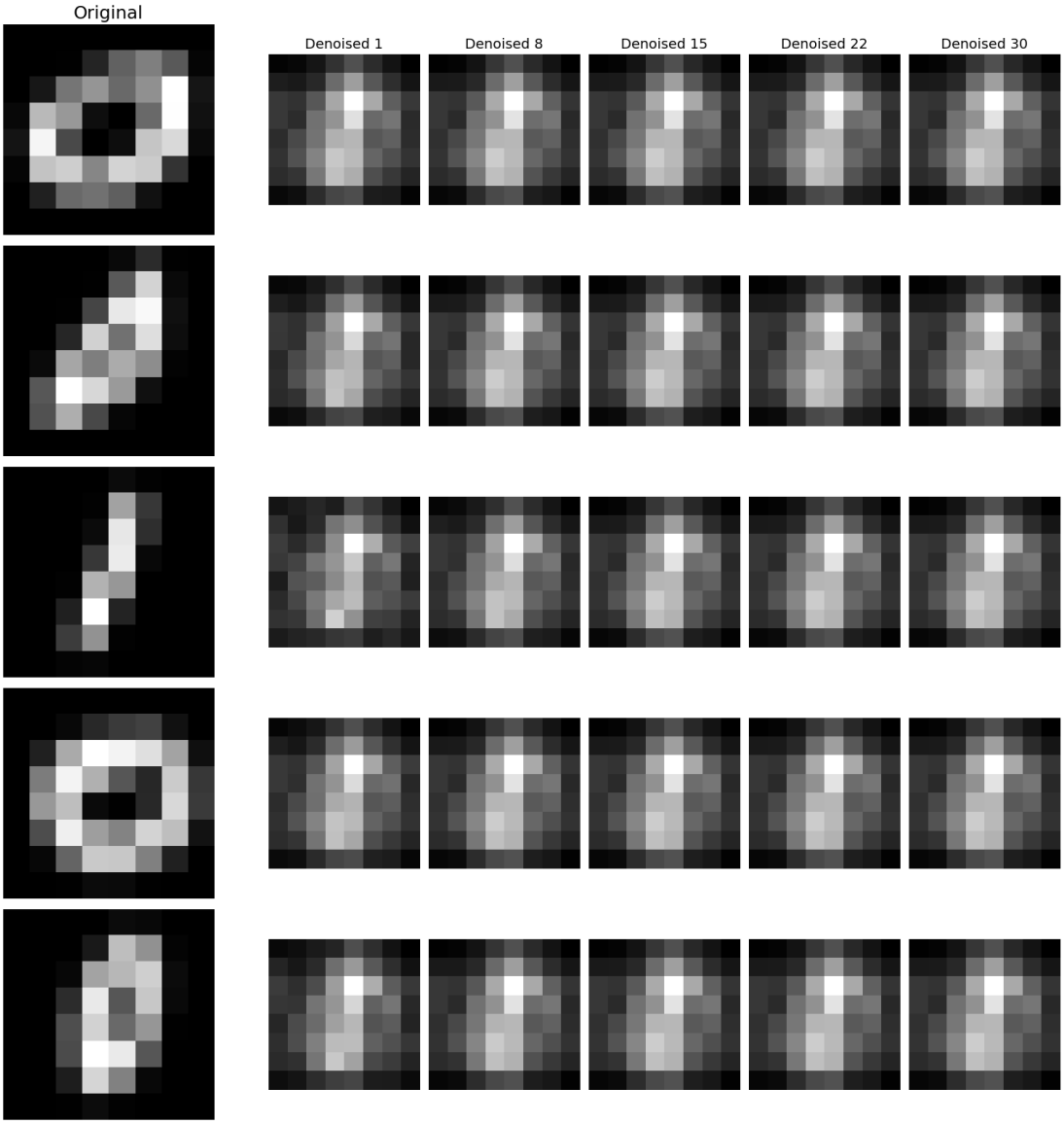


Figure 45: List of original images 8x8 followed by the one reconstructed from the classic model. The model was trained on a dataset of images with p = 0.3.

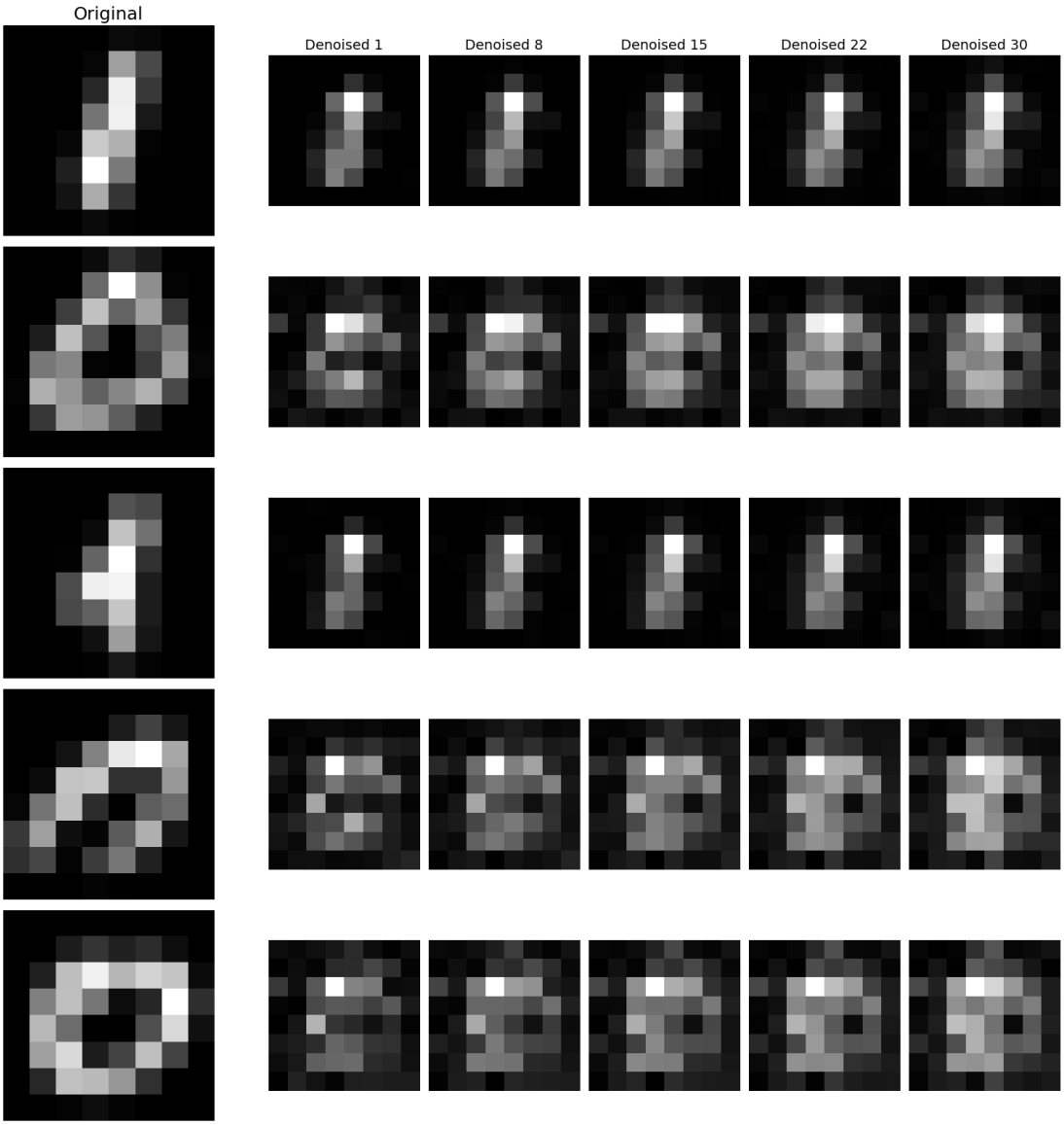


Figure 46: List of original images 8x8 followed by the one reconstructed from the quantum model. The model was trained on a dataset of images with p = 0.3.

A3 List of figures

- Figure 11: Example of a quantum circuit consisting of unitary gates [4].
- Figure 12: Circuit implemented by the paper [9] for the generator of a quantum GAN capable of generating probability distributions[9].
- Figure 13: Figures contained in the MNIST dataset [15].
- Figure 14: Quantum gan patch generator that divides the image into subportions and implements a specific generator for each part
- Implementation of my generator [14].
- Figure 19: U-net architecture used for classical diffusion model generators. Formed by convolution layers that downsample and then upsample by first reducing the feature map and increasing the channels and doing the opposite afterwards [17].
- Figure 20: Diffusion process that can be implemented by applying a forward part that can be classical or quantum and a backward part that can also be classical or quantum [18].
- Figure 21: Different probability spaces generated using different combinations of classical or quantum circuits for the forward and backward part [18].
- Figure 22: Reconstruction of a destroyed line in a point cloud [18].