Quantum Diffusion Models

Andrea Cacioppo*[1], Lorenzo Colantonio[1], Simone Bordoni[1,2,3], and Stefano Giagu[1,3]

[1] *Department of Physics, Sapienza Università di Roma, Roma, Italy*
[2] *QRC, Technology Innovation Institute, Abu Dhabi, UAE*
[3] *INFN Sezione di Roma, Roma, Italy*

Abstract

We propose a quantum version of a generative diffusion model. In this algorithm, artificial neural networks are replaced with parameterized quantum circuits, in order to directly generate quantum states. We present both a full quantum and a latent quantum version of the algorithm; we also present a conditioned version of these models. The models' performances have been evaluated using quantitative metrics complemented by qualitative assessments. An implementation of a simplified version of the algorithm has been executed on real NISQ quantum hardware.

1 Introduction

The ability to reproduce distributions from a dataset and to sample from them is one of the most useful tasks of machine learning (ML) models; these are referred to as generative models. In this context, diffusion models are algorithms inspired by nonequilibrium thermodynamics [1–4] that offer a promising alternative to the established variational autoencoders (VAEs) [5] and generative adversarial networks (GANs) [6], due to their superior sample quality and variability.

Parallel to these advancements in ML, the field of quantum computing has experienced a significant interest. Despite its rapid development in recent years, it is still early in its exploration, with much to learn about its possibilities and limits. Key milestones such as quantum error correction [7–11] have yet to be achieved, underlining the emergent nature of this technology. The current generation of quantum processors (NISQ)[12] is too noisy for the implementation of quantum supremacy algorithms [13]. On the other hand, quantum machine learning (QML) algorithms are less sensitive to noisy devices [14]. Even if these algorithms have not proven to outperform their classical counterparts, many interesting results have been obtained both in simulations and on NISQ devices [15–22]. The first meaningful utilization of quantum computing may likely come from algorithms of this kind, complemented by error mitigation techniques. [23–26].

The quantum versions of some established generative algorithms have already been

*Email: *andrea.cacioppo@uniroma1.it*

developed, like the quantum GAN [27, 28] and the quantum VAE [29], however, initial attempts to develop a quantum version of a diffusion model are limited to basic or simplified scenarios [30, 31]. In this work we present a quantum diffusion model (QDM), which can be employed for the generation of classical data or quantum data directly, in two different versions. We show the application of our approach for generating samples from the MNIST dataset and demonstrate a simplified version of this method, implemented on real quantum hardware.

This work is organized as follows:
in section 2 we provide a high level explanation of classical diffusion models and parameterized quantum circuits.
In section 3 we describe our QDM and the main differences with the classical model.
In section 4 we present the results obtained with the QDM in simulation, for different configurations, along with the metrics we used to evaluate each model.
In section 5 we describe the algorithm adaptations necessary to execute it on a NISQ device and discuss the results obtained.

2 Background

In this section, we review the main aspects of classical diffusion models and parameterized quantum circuits, necessary to understand the proposed quantum diffusion model.

2.1 Parameterized quantum circuits

A parameterized quantum circuit (PQC) [12][32] is a trainable algorithm that implements a parametric transformation on a quantum state, which is the result of the application of quantum gates, generally organized in layers. For this reason it can be considered as the quantum counterpart of an artificial neural network (ANN). In our work we have tested different layer ansatzes, obtaining the best results with the strongly entangling ansatz [33], composed of trainable rotation gates ($\hat{R}_x$, $\hat{R}_y$, $\hat{R}_z$), acting on all qubits, followed by a series of C-NOT gates, coupling neighboring qubits, as represented in figure 1.

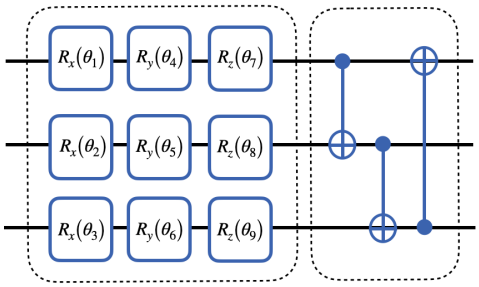

Figure 1: Example of a PQC layer ansatz for three qubits. The rotation gates contain classical, trainable, parameters while the C-NOT gates create entanglement between the qubits.

In principle, it would be possible to train a PQC directly on quantum hardware using techniques like the parameter-shift rule [34, 35]. However, computing gradients on current quantum hardware is unfeasible because of the high level of noise. For this reason, in our work, we trained the PQCs in simulations using the software library *PennyLane* [36]. Both the forward and the optimization steps are performed on a classical computer with techniques based on gradient descent. The trained PQCs can then be employed for sampling both in simulation and on quantum hardware.

When a PQC is applied to process classical data, it is necessary to add a state preparation and a measurement part, to respectively encode and decode classical information into and from quantum states. Throughout this work, we have used amplitude encoding [37]

to encode classical information into quantum states. This consists in writing a classical vector's components as the coefficients of a quantum state.

$$\mathbf{x} \longrightarrow |\mathbf{x}\rangle = \sum_{i=1}^{2^N} x_i |\mathbf{i}\rangle$$

where N is the number of qubits and $|\mathbf{i}\rangle$ represents the ordered vector of the computational basis. This way it is possible to encode 2^N classical features into N qubits. However, to perform this encoding, it is necessary to use a number of C-NOT gates that grows exponentially in the number of qubits [38].

2.2 Classical diffusion model

A diffusion model [2] is a generative model [39], namely an algorithm that is used to learn the probability distribution $p(\mathbf{x})$ associated to a dataset, with the objective of sampling from this distribution.
The main idea of this method is to consider a diffusion process, represented by a Markov chain, which maps an arbitrary distribution $q(\mathbf{x}_0)$ to a treatable distribution $\pi(\mathbf{x}_T)$, e.g. a Gaussian multivariate distribution. This is achieved through a Markov kernel $q(\mathbf{x}_t|\mathbf{x}_{t-1})$, with $t \in \{1, \cdots T\}$. A parametric model is then trained to reproduce the inverse Markov chain (reverse trajectory), estimating the inverse transition probability $p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)$.
Knowing $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, the target distribution $p_{\boldsymbol{\theta}}(\mathbf{x}_0)$ can be computed as:

$$p_{\boldsymbol{\theta}}(\mathbf{x}_0) = \int d\mathbf{x}_{1:T}\, \boldsymbol{\pi}(\mathbf{x}_T) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$$

A common choice for the forward conditional probability is the Gaussian kernel:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}) \quad (1)$$

where $\beta_1, \cdots, \beta_T$ is a variance schedule. This choice allows to sample $\mathbf{x}_t$ at an arbitrary timestep t in closed form:

$$\begin{cases} \mathbf{x}_t(\mathbf{x}_0, \epsilon) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} \\ \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \end{cases} \quad (2)$$

with $\bar{\alpha}_t \equiv \prod_{s=1}^{t}(1-\beta_s)$. When β_t is sufficiently small, the inverse conditional probability has the same functional form [40] of the forward kernel, hence:

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I})$$

where the mean can be further parameterized as:

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\Big) \quad (3)$$

with $\alpha_t \equiv 1 - \beta_t$. In this formula $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ is a function approximator, implemented by an ANN, intended to predict $\boldsymbol{\epsilon}$ from $\mathbf{x}_t$, with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. This is equivalent to minimizing the variational bound on negative log-likelihood:

$$\mathbb{E}\big[-\log p_{\boldsymbol{\theta}}(\mathbf{x}_0)\big] \leq$$
$$\leq \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] \equiv L$$

After training the model, one can sample from $p_{\boldsymbol{\theta}}(\mathbf{x}_0)$ by extracting a data point from the analytically treatable distribution $\mathbf{x}_T \sim \pi(\mathbf{x}_T)$ and applying the inverse Markov chain, as described in Algorithm 1.

Algorithm 1 Sampling algorithm

Sample $\mathbf{x}_T \sim \pi(\mathbf{x}_T)$
for $t = T, ..., 1$ **do**
  Sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = 0$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\big) + \sigma_t \mathbf{z}$
end for

3 Quantum diffusion model

The quantum adaptation of a diffusion model works through direct manipulation of quantum states. This is accomplished by substituting the traditional ANN with a parameterized quantum circuit. Due to the intrinsic differences between these two models, several adjustments are necessary to obtain a working algorithm.

3.1 Training

In a classical diffusion model, as discussed in section 2.2, a forward Markov chain is constructed by adding noise through a Markov kernel (Eq. 1). At the same time, an ANN is trained to approximate the Gaussian kernel mean (Eq. 3). This procedure is not possible in a quantum model, as it would involve decoding and encoding quantum states, at each time step, to perform classical operations on the vectors. This is impractical as current NISQ devices are subject to measurement errors and the encoding of classical data is inefficient [41]. For this reason, we have decided to utilize a hybrid approach, implementing the forward Markov chain classically, while utilizing a PQC in the backward process and the sampling phase. The PQC defines an operator $\hat{P}(\boldsymbol{\theta}, t)$, that performs data denoising directly:

$$\hat{P}(\boldsymbol{\theta}, t)|\mathbf{x}_t\rangle = |\mathbf{x}_{t-1}\rangle$$

Here $|\mathbf{x}_t\rangle$ is the quantum state obtained by encoding the classical vector $\mathbf{x}_t$, which represents the t step in the forward chain. This way, the complete backward process becomes a denoising process, which can be performed by the consecutive application of T different PQCs.

$$\hat{P}(\boldsymbol{\theta}, t=1)\cdots\hat{P}(\boldsymbol{\theta}, t=T)|\mathbf{x}_T\rangle = |\mathbf{x}_0\rangle$$

We observed that training a single circuit for all denoising steps results in only a limited decline in performance. Another important consideration is that, in the iterative application of the PQC, the coefficients of the quantum states can become complex, even if the initial state $\mathbf{x}_0$ had real values. For this reason we have tested an alternative to the classical forward kernel, as described in equation 2, where we used complex noise, instead of real noise, at every time step of the training:

$$\begin{cases} \boldsymbol{\epsilon} = \boldsymbol{\epsilon}_r + i\boldsymbol{\epsilon}_i \\ \boldsymbol{\epsilon}_r, \boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \end{cases}$$

this approach has led to a significant improvement in performance.

The PQC has been trained using a completely classical optimizer. However, during the algorithm's design, we have chosen a loss function that could be easily computed using a full quantum approach, namely the infidelity loss:

$$L(\boldsymbol{\theta}) = 1 - \mathbb{E}\big[F\big(\hat{P}(\boldsymbol{\theta}, t)|\mathbf{x}_{t+1}\rangle, |\mathbf{x}_t\rangle\big)\big]$$

where F indicate the quantum fidelity and the expected value is taken over all possible choices of $\mathbf{x}_t$ at every value of t and $\mathbf{x}_0$. On quantum hardware, the fidelity can be computed efficiently with the use of a multi-qubit swap test [42].

3.2 Quantum circuit ansatz

We have tested several architectures for the PQCs, the two circuits leading to the best performance are represented in figure 2. Both architectures use three unitary blocks operating on different qubits. In the first ansatz (bottleneck PQC), after a first unitary block $\hat{U}_n^{(1)}(\boldsymbol{\theta})$ acting on all qubits, the first m qubits are measured, reducing the number of qubits to $n-m$. After the application of the second unitary block $\hat{U}_{n-m}^{(2)}(\boldsymbol{\theta})$, acting as a bottleneck, m ancillary qubits are introduced in the state $|0\rangle^{\otimes m}$, and a final block $\hat{U}_n^{(3)}(\boldsymbol{\theta})$ is applied to

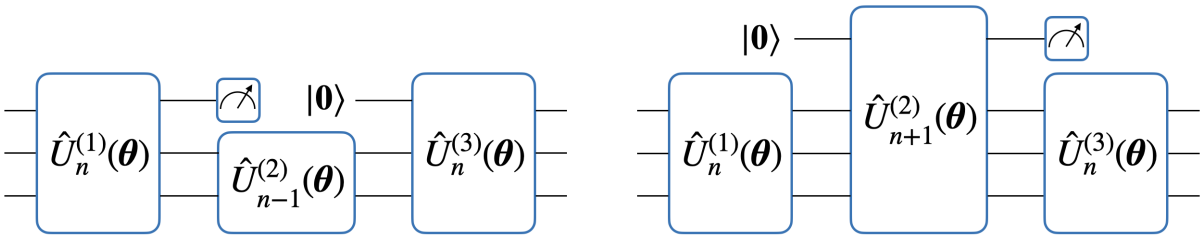

Figure 2: Representation of the bottleneck (left) and reverse-bottleneck architectures (right) using $m = 1$. Each unitary transformation block in the image employs the layered structure reported in Figure 1.

all n qubits. In the second ansatz (reverse-bottleneck PQC), the order of the ancillary qubit introduction and the measurements is reversed. This way, the central block $\hat{U}_{n+m}^{(2)}(\boldsymbol{\theta})$ acts on $n+m$ qubits. We observed little difference in the performance of the two architectures, reverse-bottleneck PQC performing slightly better. We have found that the best value for the number of measured qubits is $m = 1$. For these reasons, every result in the following refers to the reverse-bottleneck architecture, using one measured qubit.

Testing different PQC architectures, we have observed that the number of intermediate measurements of ancillary qubits have a strong impact on the performance of the algorithm. The introduction of intermediate measurements improves significantly the quality of generated samples. This can be explained by the fact that the measurement process acts as a non-linear map over the states amplitudes [37]. Our numerical simulations have confirmed the non-linearity of these maps when using our ansatzes with trained parameters. Another remarkable finding is that measurements reduce the samples variability. In particular, when their overall number exceeds a certain threshold (dependent on the architecture and number of qubits) no sensible variability is observed in the generated samples, resulting in a model collapse. One potential explanation for this phenomenon is that, every time a qubit is measured, it is reset to a fixed state. This process erases part of the information contained in the initial noise [43]. Consequently, by increasing the number of measurements, the initial information is gradually lost, resulting in a loss of variance in the output states.

Measurements of the ancillary qubits can be performed with two methods, namely with or without branch selection [37]. With branch selection, we indicate the process of measuring the ancillary qubits and accepting the remainder of the state only if the outcome coincides with a predetermined choice. If the state is rejected, the circuit is executed again from the beginning. By measurement without branch selection, we indicate the process of accepting the remainder of the quantum state regardless of the measurement outcome. We have empirically observed that utilizing measurements with branch selection leads to better results. However, when performing branch selection on quantum hardware, each measurement has an average success rate 0.5. Hence this process is not suitable for deep circuits, as the probability of accepting the circuit execution diminishes exponentially with the number of measurements. For this reason, throughout this work, we have used measurements without branch selection for the ancillary qubits. Another important remark is that, when per-

forming measurements without branch selection, the output state is nondeterministic. This characteristic aligns our model more closely with the classical diffusion model.

3.3 Model variations

In this section, we present two independent variations on the algorithm described so far: a latent classical-quantum version and a conditioned version.

Latent models To increase the expressive power of the QDM, it is possible to use a hybrid approach, by employing a pre-trained classical autoencoder [44] (latent model). With this approach, the QDM is trained on a low dimensional representation of data. The lower dimensionality of the problem, along with the strong non-linearity introduced by the classical autoencoder, has improved the quality of samples and allowed the use of smaller PQCs. Moreover, the consequent simplification of the PQC, has allowed the implementation of the algorithm on real quantum hardware, as explained in section 5.

Conditioned models Both the full quantum model and the latent model can be easily conditioned by increasing the dimension of the Hilbert space. This is done by taking the tensor product between the input quantum state and an additional state that encodes the label (label qubits):

$$|\mathbf{x}_t\rangle \longrightarrow |k\rangle \otimes |\mathbf{x}_t\rangle \tag{4}$$

where $|k\rangle$ is the k^{th} state of the computational basis. In order to encode N labels, it is necessary to add $\lceil log_2(N) \rceil$ qubits. We emphasize that, as for the ancillary qubits, it is possible to measure label qubits with or without branch selection. We have chosen to use measurements without branch selection in our hardware implementation, and measurements with branch selection in our simulations.

3.4 Model evaluation

To assess the performance of the proposed models, we utilized a combination of qualitative evaluation and quantitative metrics. To visually compare the distributions of the original and generated samples and enhance our evaluation of image quality, we employ dimensionality reduction techniques such as PCA [45] and t-SNE [46].
On the other hand, for the quantitative assesment, we have employed the ROC-AUC [47], to evaluate the conditioning performance, while for the generation performance, we have used both the Fréchet Inception Distance (FID) [48] and the 2-Wasserstein distance for Gaussian mixture models (WaM) [49]. To evaluate conditioned models, the ROC-AUC is computed using 10 binary classifiers, each specialized in distinguishing a single digit within the MNIST dataset. In this context, we assume the classifiers to be perfect, with each achieving an accuracy well above 0.99 on the original dataset. This assumption enables the trained classifiers to serve as a reliable ground truth against which the conditioned samples can be evaluated. To evaluate the generation performance we have employed both the FID and the WaM. We have chosen this second metric as the FID suffers main problems, as pointed out in [48], in particular with small and greyscale images. For the calculation of both metrics for the conditioned and unconditioned latent model, we employed a total of 70,000 samples, 10,000 at a time to evaluate satistics. For the fully quantum model we employed a total of 10,000 samples, 1000 at a time.

Table 1: PQC characteristics of different models. The number of layers are referred to the architectures represented in figure 2.

Feature	Full quantum	Unconditioned latent	Conditioned latent
Layers	30+40+30	15+20+15	15+20+15
Qubits	8	3	3+4
Parameters	2520	510	1110
Denoising steps (T)	15	8	8
Images size	16x16	28x28	28x28
Latent space	N.A.	8	8
Digits	"0"/"1"	All	All

4 Simulation Results

In this section, we present the results obtained in simulation for the full quantum model without conditioning, and the latent model with and without conditioning. We omit the results obtained with the conditioned full quantum model, as adding a conditioning on only two labels did not give relevant results differences. We have chosen to train each model on the MNIST dataset. For the full quantum model, the samples have been downsized from 28x28 pixels to 16x16 pixels to be encoded into 8 qubits using amplitude encoding. Moreover, only zeros and ones have been used to further simplify the problem. For the latent models, we have chosen to use latent vectors of dimension 8, encoded using 3 qubits. In this case, we used every digit of the MNIST dataset. The conditioned version of the latent model required 4 additional qubits to encode the labels (10 digits), for a total of 7 qubits. We have tested different number of layers for the unitary blocks of the reverse-bottleneck circuit ansatz (section 3.2). For the latent models, as the PQC acts on a smaller Hilbert space, it is possible to obtain optimal performances with a total number of 50 layers. The full quantum model requires a total number of 100 layers in order to reach sufficient expressive power. Even if these circuits are composed of a high number of layers, we have not experienced any issue related to barren plateaus [50] during training. A summary of the characteristics of the models are reported in table 1.
To convert quantum states into classical vectors, we have adopted the convention of considering the absolute values of the quantum state's amplitudes. Experimentally, this is equivalent to taking the square roots of the measurement frequencies in the computational basis.

4.1 Full quantum model

Samples generated by the full quantum model are shown in figure 3. The characteristics of the model used for the generation of these samples are reported in the first column of table 1. We highlight how the number of time steps T is significantly smaller with respect to classical diffusion models. Two examples of the denoising process are represented in figure 4.
The value of the FID between the original and generated datasets can be found in the first column of table 2.

4.2 Unconditioned latent model

Samples generated with the unconditioned latent model are shown in figure 5. Each image has been obtained by sampling from the quan-

Table 2: Evaluation metrics for different models

	Full quantum	**Unconditioned latent**	**Conditioned latent**
FID	256.6 ± 2.0	41.4 ± 0.3	38.2 ± 2.7
WaM	240.3 ± 2.4	25.8 ± 0.8	13.5 ± 1.1

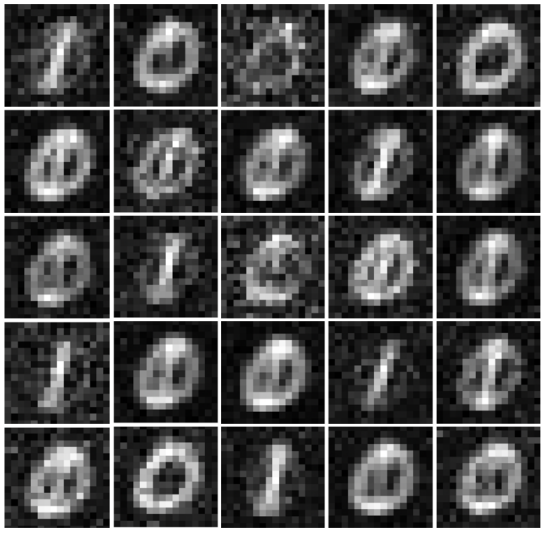

Figure 3: Samples generated from the full quantum model.

tum diffusion model and decoding each sample with a pre-trained decoder. The characteristics of the model used to generate these samples are reported in the second column table 1. The value of the FID and WaM between the original and generated datasets can be found in the second column of table 2.

4.3 Conditioned latent model

Samples generated with the conditioned latent model are shown in figure 6. Conditioning the model improves the samples quality. This improvement can be attributed to the fact that, in the conditioned model, the guiding process produces a clearer differentiation among the digits during sampling. Moreover, adding four qubits increases the dimension of the Hilbert space of the circuit and the number of trainable parameters, thus giving more expressive power to the PQC. The characteristics of the model used to generate these samples are reported in the last column of table 1. A comparison between the original and generated dataset in the latent space is reported in figure 7, where we have used t-SNE to generate a 2D representation. To assess conditioning, we use the ROC-AUC as detailed in section 3.4. The area under the curve (AUC) values for the ROC curves, specific to each digit, are presented in Table 3. The digits "one" and "zero" exhibit the highest AUC values, which can be attributed to their simplicity and minimal overlap with other digits. The value of the FID and WaM between the original and generated datasets can be found in the third column of table 2. This is computed between the overall distributions, hence not considering the separation of digits.

Digit	AUC	Digit	AUC
0	0.951	5	0.899
1	0.977	6	0.975
2	0.983	7	0.928
3	0.989	8	0.908
4	0.857	9	0.933

Table 3: ROC-AUC values for each digit.

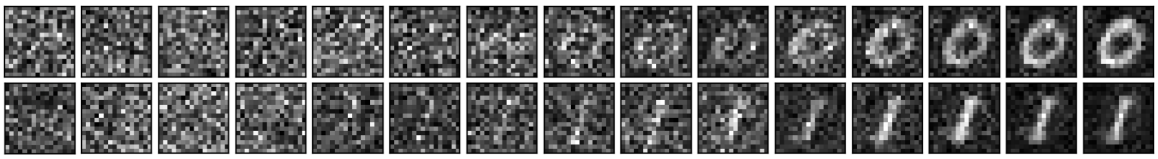

Figure 4: Detail on the denoising process of a "zero" and a "one" digit for the full quantum model.

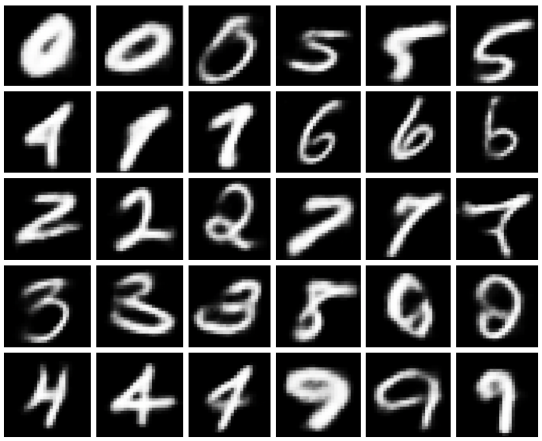

Figure 5: Unconditioned latent model samples (ordered).

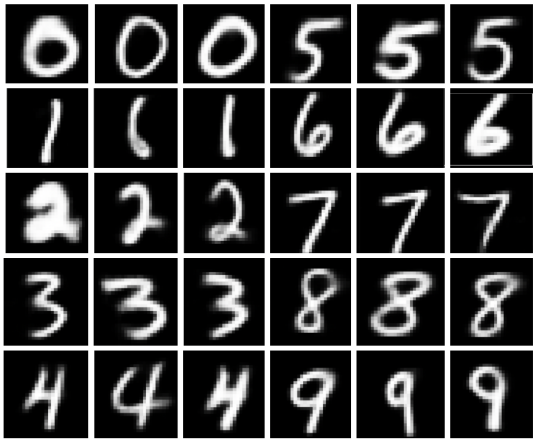

Figure 6: Conditioned latent model samples.

5 Quantum hardware execution

Executing quantum circuits on current quantum devices poses significant challenges even when utilizing cutting-edge quantum hardware. The primary obstacles stem from the substantial error rates associated with quantum gates, particularly the C-NOT gates required for entangling qubits. A second problem comes from the short qubit relaxation time $T1$ [51]. This value, also known as qubit lifetime, provides an estimate of how long a qubit can be used effectively in quantum computations before it becomes too prone to errors caused by decoherence. Another critical issue regards the quantum computer's architectural connectivity, since it's impossible to apply C-NOT gates between all qubit pairs. Contemporary NISQ devices provide connectivity exclusively between neighboring qubits, typically organized in linear or circular configurations. Figure 8 illustrates the architecture of the quantum chip ibm_hanoi employed in this study.

When interactions between non-connected qubits are necessary, SWAP gates must be introduced in the circuit to exchange the quantum states of two qubits. It's worth noting that each SWAP gate requires three C-NOT gates. Given these constraints, achieving significant results using the circuit described in section 3 is not possible. For the implementation of a QDM on quantum hardware, several modifications are essential to reduce the circuit's complexity.

https://quantum-computing.ibm.com/services/resources?system=ibm_hanoi, 2021

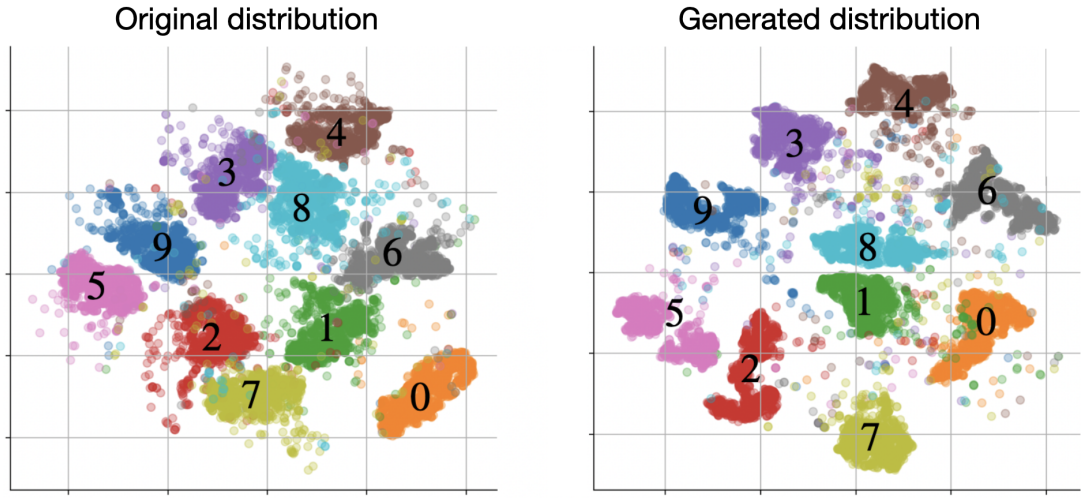

Figure 7: 2D representation (t-SNE) of the original distribution and generated distribution in the latent space (8D). Each digit is generated separately by conditioning the model.

5.1 Quantum circuit adaptation

In this section, we outline the modifications made to the quantum circuit to address the issues introduced before. Regarding the connectivity, it's worth noting that the PQC ansatz proposed in Section 3 necessitates only interactions between neighboring qubits when they are arranged in a circular topology. However, in our quantum hardware configuration (as depicted in Figure 8), arranging six qubits in a circular pattern is not feasible. To mitigate this, we removed the last C-NOT gate from each entangling layer, thereby requiring interactions solely between neighboring qubits arranged linearly. This adjustment eliminated the need for SWAP gates within the circuit, significantly reducing the overall count of C-NOT gates.

To reduce the circuit depth we have applied several changes. The first important change regards the number of denoising step. In fact, the complete quantum circuit is composed by a repetition of the denoising circuit for each denoising step. We found out that it is still possible to generate samples using just three denoising steps.

The second bigger change regards using a smaller denoising circuit, by reducing the number of layers. In particular we have used a reverse-bottleneck ansatz (section 3), reported in figure 9. To restore the label qubit after its measurement, a conditional Pauli X gate is applied.

To reduce the circuit depth further, we have also rearranged the C-NOT gates disposition in the entangling layers. In particular, we first apply all the C-NOT gates with an even control qubit and then the ones with an odd control qubit. This way only two circuit moments are required for each entangling layer.

The expressive power of the quantum circuit is highly reduced by all these adaptations, therefore, we have reduced the complexity of the task. We have restricted the problem to the conditioned generation of only zeros and ones, this way only one qubit is required to encode the label. We have also reduced the latent space dimension from 8 to 4, this way only two qubits are required to encode the state. The final circuit is thus composed of a total of four

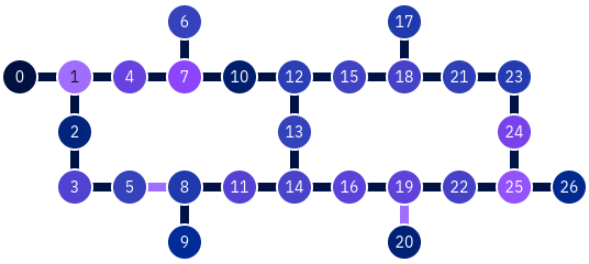

Figure 8: Architecture of IBM_hanoi quantum computer. The C-NOT connectivity is reported. The colors of single qubits and their connections represent respectively the $T1$ and C-NOT error probabilities. Darker colours represent a lower T1, in the range $[47, 265]\mu s$. For the C-NOT errors, darker colors represent a lower error rate, in the range $[3.2 \times 10^{-3}, 1]$. Calibration data are referred to 19/10/2023.

qubits and 37 C-NOT gates (36 for the PQC plus one for the amplitude encoding of the initial noise).

5.2 Results

The hardware adapted PQC has been trained in simulation. Training on hardware devices is unfeasible on NISQ devices because of the high level of noise that makes the computation of gradients too imprecise. The training has been performed in the same way as described in section 3.3 using "zero" and "one" handwritten digits. The trained weights can be used for sampling on the real device. Before the hardware execution we have tested the trained algorithm in noiseless simulations. We have generated 5×10^3 latent vectors of both "zero" and "one" digits in order to compare their distribution with respect to the latent vectors of the train set. Figure 10 shows the two dimensional PCA representation of the latent space for original and generated latent vectors. It is possible to observe that the model is able to correctly generate latent vectors with the distribution of the "one" digits. On the other hand, for the "zero" digits, there is not a full overlap between the two distributions. The model can generate samples that belong exclusively to a specific region within the original distribution, although the generated samples show a good variability. We have observed similar results in different training executions.

In order to study the effects of noise we have generated the latent vector of a "zero" digit in four different ways: state vector simulation, shots simulation, realistic noise simulation and quantum hardware. The realistic noise simulation has been performed with the Qiskit built-in noise model, that uses the calibration data to perform a shots simulation with realistic noise. Apart from the state vector simulation, all other methods require to measure the output states. The need to perform these measurements creates some difficulties, since the final state depends on the results of the measurement of the ancillary qubits. Six measurements are performed before the final state, three for the ancillary qubit and three for the label qubit. This means that, starting from the same initial noise, it is possible to obtain 64 possible final states. In order to sample them all, it is necessary to execute the circuit over a large number of shots. In this study, we have performed 3.2×10^5 shots for every circuit, obtaining, on average, 500 shots for each possible final state. Figure 11 reports the measurements frequency for each of the methods described before. It is possible to observe how the shot noise obtained with the shots simulation is negligible, with no significative difference with respect to the state vector simulation. The difference between the hardware execution and the state vector simulation is more relevant. The noise changes the amplitudes of the states 01 and 11 from about 0.1 to 0.2. This is an expected result, as decoherence, due to unwanted interactions with the environ-

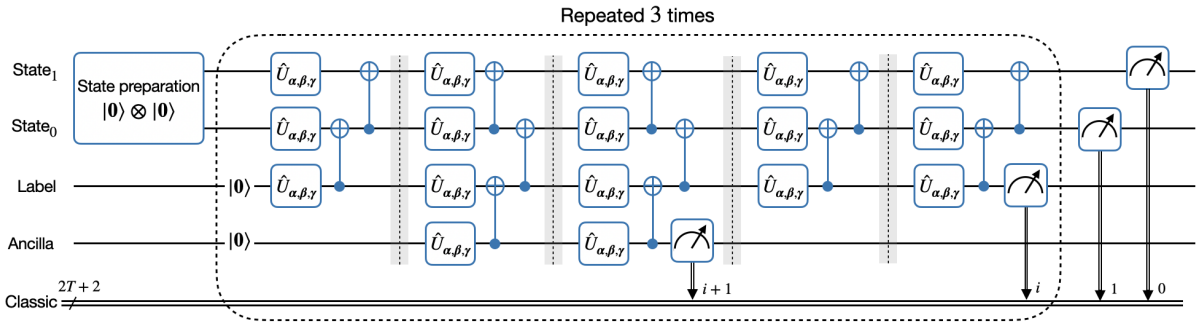

Figure 9: Quantum hardware adapted PQC. The central part of the circuit is repeated three times in total. State preparation is necessary only to encode the initial noise. Similarly, the state qubits are measured only at the end.

ment, tends to bring the output state closer to the maximally mixed state. The realistic noise simulation is in good accordance with the real result, with the differences being in the range $5 - 10\%$.

The effects of noise are relevant but do not influence the quality of the generated samples. Figure 12 shows the comparison of a "zero" and a "one" digit, generated with state vector simulation and on quantum hardware. The latent vectors have been decoded with the classical decoder. It is possible to observe how, in the presence of noise, the generated images change slightly, but still showing all the features of the correct class.

We have used the results obtained on quantum hardware to study the effects of noise on the variability of the generated samples. Figure 13 reports a two dimensional PCA representation of the latent space for the original and generated dataset, both in simulation and on quantum hardware. In both cases, the generated latent vectors have been obtained executing 3.2×10^5 shots starting from the same initial noise. With this process, as previously explained, it is possible to obtain 64 samples for each class.

In figure 13 it is possible to observe that noise reduces the variability of the generated samples. This is an opposite result with respect to the classical case, where introducing noise after each denoising step helps increasing the sample variability [3]. The reduced variability with noise can be attributed to decoherence, which converges all final states towards the maximally mixed state. Moreover, measurement errors tend to mix different samples, thus averaging them. This result highlights how the intrinsic differences between classical and quantum noise lead to differences in the behavior of classical and quantum generative algorithms.

6 Conclusions

In this paper, we introduce and analyze a novel quantum diffusion model, consisting of two distinct variants. The first is a comprehensive quantum model designed to generate classical or quantum data directly. The second is a latent classical-quantum model aimed at generating classical data. Both models can be conditioned using additional qubits to encode labels, thereby enhancing their practical utility. In our experiments we have demonstrated the practical applicability of these models on cur-

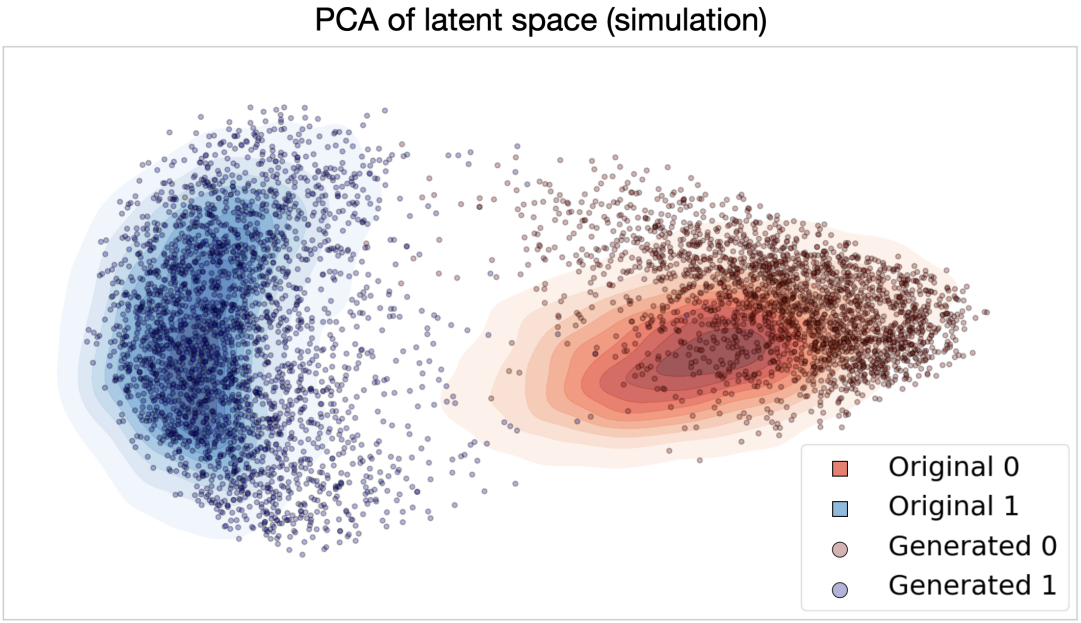

Figure 10: 2D representation (PCA) of the latent vectors of the original distribution (continuous plots) and generated distribution with simulations (circles). The original distribution is composed by all the "zeros" and "ones" handwritten digits of the MNIST dataset. The generated distribution contains 5×10^3 samples of each class.

rent NISQ devices.

The results obtained show that quantum diffusion models require fewer parameters with respect to their classical counterpart. For the latent model, this allows us to generate samples of similar quality to those generated by classical algorithms. Moreover, in the full quantum case, unlike classical models, it is possible to generate distributions whose features scale exponentially with the number of qubits. Although in simulation and on current NISQ devices we can represent a number of features comparable to the classical case, in the future the model might allow us to approximate probability distributions that are not classically tractable, including quantum datasets. At this point, only the sampling part can be performed on quantum hardware. However, with the expectation of noiseless devices becoming available, it will be possible to train our algorithm in a fully quantum fashion, leveraging the parameter-shift rules.

A currently open problem remains to determine if it is possible to define a diffusion process directly in the Hilbert space, in order to implement a fully quantum pipeline that doesn't require classical computations.

Acknowledgements

This work is partially supported by ICSC – Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing, funded by European Union – NextGenerationEU.

Data Availability Statement

Both the datasets and the code used for this study are available on request by contacting the authors.

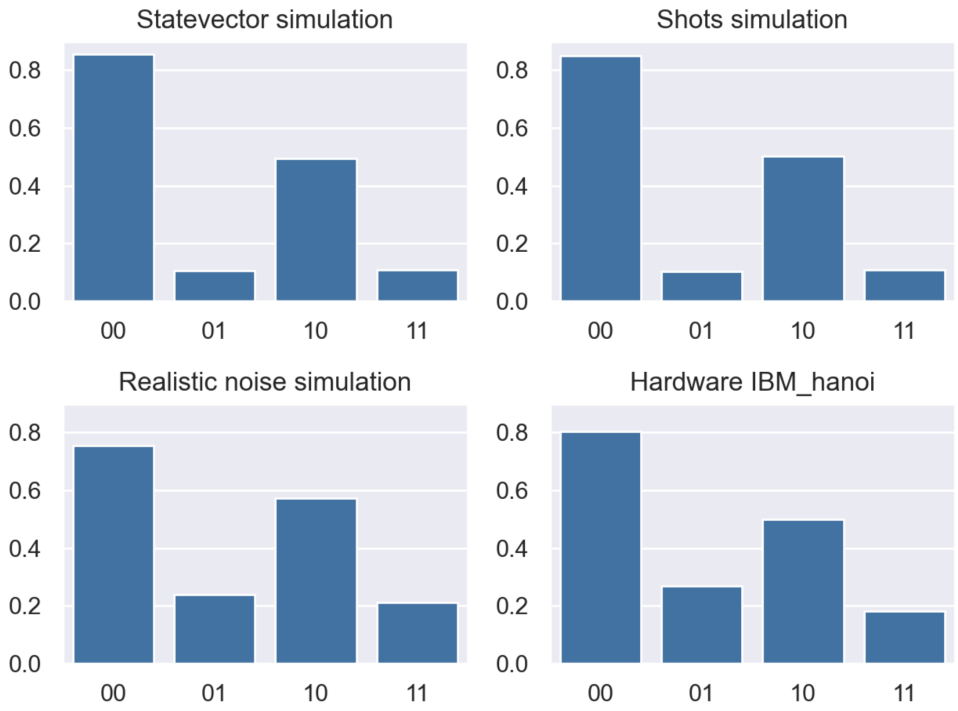

Figure 11: State amplitudes of a generated latent vector obtained under different noise conditions: state vector simulation, shots simulation, realistic noise simulation, quantum hardware.

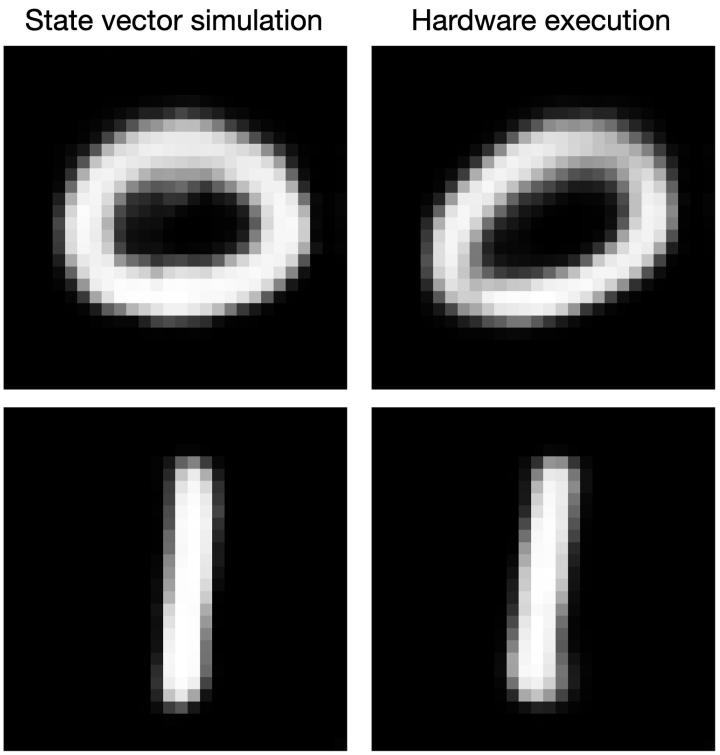

Figure 12: Generated images of a “zero” and a “one” handwritten digits. The latent vector has been generated with both a non noisy state vector simulation and with real quantum hardware in order to study the effects of noise.

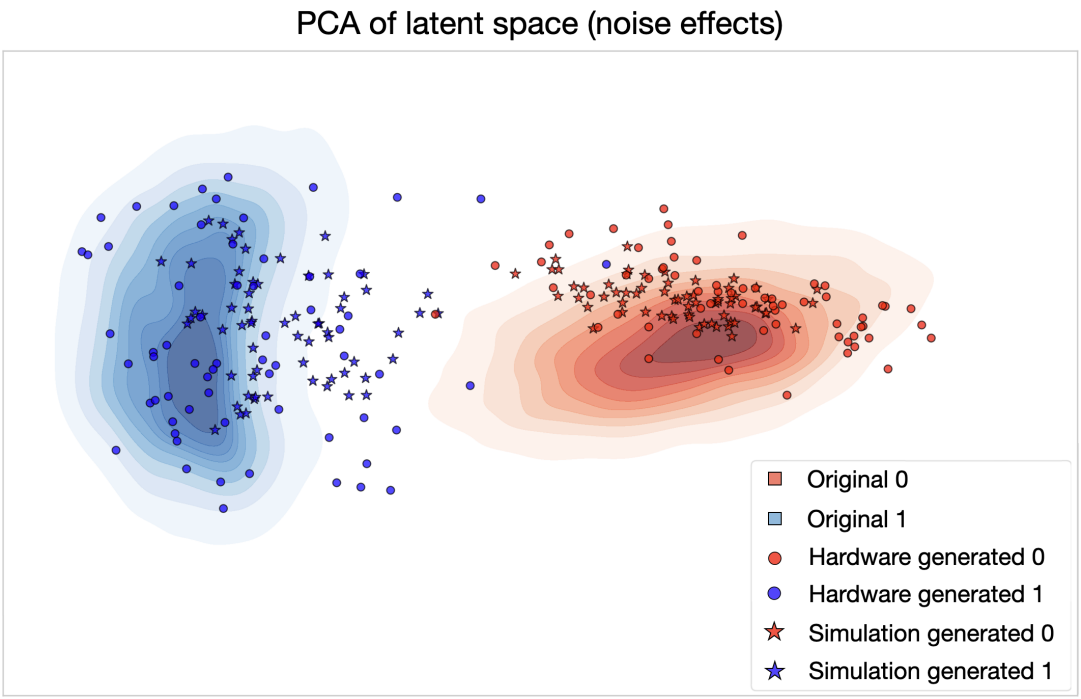

Figure 13: 2D PCA representation of the latent vectors distribution obtained from the original MNIST dataset (continuous plots), generation with quantum hardware (stars) and generation with non-noisy shots simulation (circles). The generated distributions contain 64 samples generated with 3.2×10^5 shots and starting from the same initial noise.

References

[1] Jascha Sohl-Dickstein et al. "Deep unsupervised learning using nonequilibrium thermodynamics". In: *International conference on machine learning*. PMLR. 2015, pp. 2256–2265.

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[3] Florinel-Alin Croitoru et al. "Diffusion Models in Vision: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.9 (2023), pp. 10850–10869. DOI: 10.1109/TPAMI.2023.3261988.

[4] Jiaming Song, Chenlin Meng, and Stefano Ermon. *Denoising Diffusion Implicit Models*. 2022. arXiv: 2010.02502 [cs.LG].

[5] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[6] Ian Goodfellow et al. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[7] Austin G. Fowler et al. "Surface codes: Towards practical large-scale quantum computation". In: *Phys. Rev. A* 86 (3 Sept. 2012), p. 032324. DOI: 10.1103/PhysRevA.86.032324. URL: https://link.aps.org/doi/10.1103/PhysRevA.86.032324.

[8] Savvas Varsamopoulos, Ben Criger, and Koen Bertels. "Decoding small surface codes with feedforward neural networks". In: *Quantum Science and Technology* 3.1 (Nov. 2017), p. 015004. DOI: 10.1088/2058-9565/aa955a. URL: https://doi.org/10.1088%2F2058-9565%2Faa955a.

[9] Bordoni Simone and Giagu Stefano. "Convolutional neural network based decoders for surface codes". In: *Quantum Information Processing* 22.3 (Mar. 2023), p. 151. DOI: 10.1007/s11128-023-03898-2. URL: https://doi.org/10.1007/s11128-023-03898-2.

[10] E Knill. "Quantum computing with realistically noisy devices". In: *Nature* 434 (Apr. 2005), pp. 39–44. DOI: 10.1038/nature03350.

[11] Kosuke Fukui et al. "High-Threshold Fault-Tolerant Quantum Computation with Analog Quantum Error Correction". In: *Phys. Rev. X* 8 (2 May 2018), p. 021054. DOI: 10.1103/PhysRevX.8.021054. URL: https://link.aps.org/doi/10.1103/PhysRevX.8.021054.

[12] John Preskill. "Quantum computing in the NISQ era and beyond". In: *Quantum* 2 (2018), p. 79.

[13] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. DOI: 10.1137/s0097539795293172. URL: https://doi.org/10.1137%2Fs0097539795293172.

[14] Jacob Biamonte et al. "Quantum machine learning". In: *Nature* 549.7671 (Sept. 2017), pp. 195–202. DOI: 10.1038/nature23474. URL: https://doi.org/10.1038%2Fnature23474.

[15] Sau Lan Wu and Shinjae Yoo. "Challenges and opportunities in quantum machine learning for high-energy physics". In: *Nature Reviews Physics* 4.3 (Mar. 2022), pp. 143–144. ISSN: 2522-5820. DOI: 10.1038/s42254-022-00425-7. URL: https://doi.org/10.1038/s42254-022-00425-7.

[16] M. Cerezo et al. "Variational quantum algorithms". In: *Nature Reviews Physics* 3.9 (Aug. 2021), pp. 625–644. DOI: 10.1038/s42254-021-00348-9. URL: https://doi.org/10.1038%2Fs42254-021-00348-9.

[17] Daniel J. Egger et al. *A study of the pulse-based variational quantum eigensolver on cross-resonance based hardware*. 2023. arXiv: 2303.02410 [quant-ph].

[18] Simone Bordoni et al. "Long-Lived Particles Anomaly Detection with Parametrized Quantum Circuits". In: *Particles* 6.1 (2023), pp. 297–311. ISSN: 2571-712X. DOI: 10.3390/particles6010016. URL: https://www.mdpi.com/2571-712X/6/1/16.

[19] Matteo Robbiati, Juan M. Cruz-Martinez, and Stefano Carrazza. *Determining probability density functions with adiabatic quantum computing*. Tech. rep. 7 pages, 3 figures. 2023. arXiv: 2303.11346. URL: http://cds.cern.ch/record/2853183.

[20] Matteo Robbiati et al. *A quantum analytical Adam descent through parameter shift rule using Qibo*. 2022. arXiv: 2210.10787 [quant-ph].

[21] Juan M. Cruz-Martinez, Matteo Robbiati, and Stefano Carrazza. *Multi-variable integration with a variational quantum circuit*. 2023. arXiv: 2308.05657 [quant-ph].

[22] ShiJie Wei et al. *A Quantum Convolutional Neural Network on NISQ Devices*. 2021. arXiv: 2104.06918 [quant-ph].

[23] Alejandro Sopena et al. "Simulating quench dynamics on a digital quantum computer with data-driven error mitigation". In: *Quantum Science and Technology* 6.4 (July 2021), p. 045003. DOI: 10.1088/2058-9565/ac0e7a. URL: https://dx.doi.org/10.1088/2058-9565/ac0e7a.

[24] Armands Strikis et al. "Learning-Based Quantum Error Mitigation". In: *PRX Quantum* 2 (4 Nov. 2021), p. 040330. DOI: 10.1103/PRXQuantum.2.040330. URL: https://link.aps.org/doi/10.1103/PRXQuantum.2.040330.

[25] Paul D. Nation et al. "Scalable Mitigation of Measurement Errors on Quantum Computers". In: *PRX Quantum* 2 (4 Nov. 2021), p. 040326. DOI: 10.1103/PRXQuantum.2.040326. URL: https://link.aps.org/doi/10.1103/PRXQuantum.2.040326.

[26] Lena Funcke et al. "Measurement error mitigation in quantum computers through classical bit-flip correction". In: *Phys. Rev. A* 105 (6 June 2022), p. 062404. DOI: 10.1103/PhysRevA.105.062404. URL: https://link.aps.org/doi/10.1103/PhysRevA.105.062404.

[27] Pierre-Luc Dallaire-Demers and Nathan Killoran. "Quantum generative adversarial networks". In: *Physical Review A* 98.1 (2018), p. 012324.

[28] Carlos Bravo-Prieto et al. "Style-based quantum generative adversarial networks for Monte Carlo events". In: *Quantum* 6 (Aug. 2022), p. 777. DOI: 10.22331/q-2022-08-17-777. URL: https://doi.org/10.22331%2Fq-2022-08-17-777.

[29] Amir Khoshaman et al. "Quantum variational autoencoder". In: *Quantum Science and Technology* 4.1 (2018), p. 014001.

[30] Marco Parigi, Stefano Martina, and Filippo Caruso. "Quantum-Noise-driven Generative Diffusion Models". In: *arXiv preprint arXiv:2308.12013* (2023).

[31] Bingzhi Zhang et al. "Generative quantum machine learning via denoising diffusion probabilistic models". In: *arXiv preprint arXiv:2310.05866* (2023).

[32] Marcello Benedetti et al. "Parameterized quantum circuits as machine learning models". In: *Quantum Science and Technology* 4.4 (2019), p. 043001.

[33] Maria Schuld et al. "Circuit-centric quantum classifiers". In: *Physical Review A* 101.3 (2020), p. 032308.

[34] Kosuke Mitarai et al. "Quantum circuit learning". In: *Physical Review A* 98.3 (2018), p. 032309.

[35] Maria Schuld et al. "Evaluating analytic gradients on quantum hardware". In: *Physical Review A* 99.3 (2019), p. 032331.

[36] Ville Bergholm et al. "Pennylane: Automatic differentiation of hybrid quantum-classical computations". In: *arXiv preprint arXiv:1811.04968* (2018).

[37] Maria Schuld and Francesco Petruccione. *Machine learning with quantum computers*. Springer, 2021.

[38] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. "Effect of data encoding on the expressive power of variational quantum-machine-learning models". In: *Phys. Rev. A* 103 (3 Mar. 2021), p. 032430. DOI: 10.1103/PhysRevA.103.032430. URL: https://link.aps.org/doi/10.1103/PhysRevA.103.032430.

[39] Lars Ruthotto and Eldad Haber. "An introduction to deep generative modeling". In: *GAMM-Mitteilungen* 44.2 (2021), e202100008.

[40] Jascha Sohl-Dickstein et al. *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. 2015. arXiv: 1503.03585 [cs.LG].

[41] John A. Cortese and Timothy M. Braje. *Loading Classical Data into a Quantum Computer*. 2018. arXiv: 1803.01958 [quant-ph].

[42] Adriano Barenco et al. "Stabilization of quantum computations by symmetrization". In: *SIAM Journal on Computing* 26.5 (1997), pp. 1541–1557.

[43] Michael A Nielsen. "The entanglement fidelity and quantum error correction". In: *arXiv preprint quant-ph/9606012* (1996).

[44] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[45] Ian T Jolliffe and Jorge Cadima. "Principal component analysis: a review and recent developments". In: *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences* 374.2065 (2016), p. 20150202.

[46] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008).

[47] Andrew P Bradley. "The use of the area under the ROC curve in the evaluation of machine learning algorithms". In: *Pattern recognition* 30.7 (1997), pp. 1145–1159.

[48] Martin Heusel et al. "Gans trained by a two time-scale update rule converge to a local nash equilibrium". In: *Advances in neural information processing systems* 30 (2017).

[49] Lorenzo Luzi et al. "Evaluating generative networks using Gaussian mixtures of image features". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2023, pp. 279–288.

[50] Jarrod R McClean et al. "Barren plateaus in quantum neural network training landscapes". In: *Nature communications* 9.1 (2018), p. 4812.

[51] Malcolm Carroll et al. *Dynamics of superconducting qubit relaxation times*. 2022. arXiv: 2105.15201 [quant-ph].