UNIVERSITY
OF OSLO

Master's thesis

Quantum Diffusion Model

Markus Leitvoll, Nikolai Theien Kivijervi

Programming and System Architecture
60 ECTS study points

Department of Informatics
Faculty of Mathematics and Natural Sciences

Spring 2024

Markus Leitvoll
Nikolai Theien Kivijervi

Quantum Diffusion Model

Supervisors:
Michael Kirkedal Thomsen
Noah Oldfield
Shaukat Ali

Abstract

Diffusion models excel in image generation and manipulation but suffer from significant computational demands and slow sampling. Quantum computing offers the potential to overcome these challenges. This thesis explores the integration of quantum computing with diffusion models, introducing a novel Quantum Diffusion Model (QDM) with four distinct variants implemented using Parameterized Quantum Circuits. This work investigates both grayscale and, for the first time, full-color image generation using QDMs, and explores the impact of our novel timestep and label embedding techniques. Our results demonstrate the promising capabilities of QDMs, particularly those utilizing ancilla qubit-powered temporal context. This thesis shows that QDMs offer the potential for improved scalability compared to classical diffusion models, enabling the generation of larger images and potentially achieving faster sampling speeds.

Contents

Nomenclature

σ	Standard deviation of the noise distribution
τ	Number of timesteps
ζ	Value used for noise amplification during training
E	Number of epochs
L	Depth of the quantum circuit
t	One timestep
Z	Value used for noise amplification during sampling
DDIM	Denoising Diffusion Implicit Model
DDPM	Denoising Diffusion Probabilistic Model
NISQ	Noisy Intermediate-Scale Quantum
PQC	Parameterized Quantum Circuit
QDM	Quantum Diffusion Model
RGB	Red, Green, Blue (colored images)

Preface

This thesis is a joint effort between Markus Leitvoll and Nikolai Theien Kivijervi, undertaken from January 2023 to May 2024 as part of our computer science studies at the University of Oslo, hosted by the Simula Research Laboratory. We deliberately sought a project that would challenge us and expand our knowledge beyond our familiar programming domain. The intersection of quantum physics and computing presented a compelling challenge for us both.

Quantum physics and quantum computing research is often written and explored through a mathematical lens. Our background lies in classical computer science, and thus our design choices and methodologies are inherently influenced by this. While this approach may differ from traditional methodologies, we believe that our findings offer valuable insights and contribute a fresh viewpoint to the field of quantum machine learning. We are excited to share the results of our exploration and the knowledge we have gained.

We would like to sincerely thank our supervisors, whose guidance and support have been incredibly valuable throughout this thesis. Michael, for your commitment and insights, even in the middle of bustling airports, ensuring we always had the supervision we needed to stay on course. Shaukat, for nudging us in the right direction, and being an incredible academic resource throughout this research. Noah, for sticking with us throughout this whole journey despite your own demanding research, and for your ability to demystify complex quantum concepts in a way even we "less-enlightened" folks could understand. Thank you all for the engaging discussions, laughter, and even the casual conversations about vacations. We are incredibly grateful.

Additionally, we extend a thank you to Michael Kölle and his team for providing access to their quantum denoising diffusion model. It provided some much-needed assistance that helped shape our own model, and by extension this thesis.

Lastly, a special thanks to everyone at the Simula Research Laboratory – your support and friendly atmosphere were invaluable to us. We will always cherish the memories made during those much-needed breaks around the pool table. Thankfully, both our pool shots and our model's output started to resemble something intentional over this one-and-a-half year journey.

The code for the model proposed in this thesis is available at https://github.com/markusleitvoll/quantum-diffusion-model.

Chapter 1

Introduction

Recently, diffusion models have emerged as significant contributors to advancements in artificial intelligence and digital media generation. Their ability to generate diverse and high-quality synthetic media has led to widespread adoption, and platforms like DALL-E 2 [40] and Stable Diffusion [45] have collectively produced billions of images [24]. The realism and adaptability of diffusion models have made them valuable tools in fields including image restoration [31], medical anomaly detection [58], and AI-generated video content [2].

Traditional diffusion models rely heavily on computational resources, and face limitations in scaling and speed. Training leading models requires significant computational and financial resources, which can escalate to hundreds of thousands GPU hours and dollars [37]. These substantial resource requirements is a critical bottleneck in deploying diffusion models widely, as emphasized in recent studies which document the high computational cost and elaborate on the difficulties in training these models efficiently [61]. Moreover, diffusion models uses an iterative sampling process involving complex differential equations, resulting in inherently long sampling times due to the extensive number of function evaluations required [1].

In response to these challenges, this thesis explores the integration of quantum computing with diffusion models to form a Quantum Diffusion Model (QDM). Quantum computing offers the potential to process information much more efficiently than classical computers [15, 23], which could dramatically reduce the computational load and time required by traditional diffusion models. This research presents one of the first implementations of a QDM, introducing a novel model with four variants. These include a base model establishing the core architecture, a temporal model integrating temporal information through timestep embedding, a conditional model enabling targeted image generation using label embedding, and a hybrid model combining the strengths of both embedding techniques. Moreover, This thesis introduces a model capable of generating full-color images which, to the best of our knowledge, is the first of its kind.

This thesis will include our rationale behind the design, a detailed walk-through of the implementation, and the subsequent evaluation of our QDM. Beyond the technical contributions, this thesis is committed to fostering an understanding of quantum diffusion processes among a broader audience. In that light, and in an effort to contribute to the growth and accessibility of this research field, we will release the source code for our QDM.

1.1 Research Questions

This thesis explores the intersection of quantum computing and diffusion models, focusing on the development and evaluation of a novel Quantum Diffusion Model (QDM). Our overarching research questions investigate the potential performance gains, hyperparameter optimization, and the applicability of quantum timestep embedding techniques for enhancing the QDMs image generation capabilities. Furthermore, we aim to break new ground in QDM research by exploring their potential to generate color (RGB) images, venturing into a territory yet to be explored in this field.

RQ 1: What are the generation capabilities of our QDM, and how do factors such as computational resources, hyperparameters, and training data influence its performance?

This research question investigates the core capabilities of our QDM. It investigates how computational resources, hyperparameter choices, and the nature of training data shape the QDMs performance. Additionally, it probes the broader potential of quantum techniques to bring theoretical advantages to diffusion models, such as improved speed, efficiency, and reduced resource needs.

RQ 2: How does timestep embedding influence the quality and coherence of images generated by quantum diffusion models?

This investigates the adaptation of timestep embedding, a critical technique in classical diffusion models, to the quantum domain. The focus is on whether this adaptation can improve the quality and coherence of generated images, potentially setting new standards for temporal modeling within QDMs.

RQ 3: Is the model suited for generating RGB images, and what are the potential limitations?

This research question explores the potential for extending our QDM to generate RGB (color) images. In doing so, it aims to identify the computational challenges and requirements associated with this advancement in the field of QDMs.

1.2 Scope and Limitations

This thesis approaches quantum machine learning from a background in classical computer science. While this provides a unique and practical perspective, it is important to acknowledge that this background may mean insights differ from those derived through a traditional mathematical lens.

Our work takes place within the current Noisy Intermediate-Scale Quantum (NISQ) era of quantum computing [44]. NISQ devices, with 50 to several hundred qubits, surpass the ability of classical supercomputers to directly simulate quantum systems. However, NISQ devices are highly susceptible to errors due to quantum noise and decoherence. This is the primary challenge in this era: the difficulty in managing errors that occur during quantum computations. These errors are due to the extremely sensitive nature of qubits, which can be easily disturbed by environmental interactions, such as temperature fluctuations, electromagnetic fields, and even imperfections within

the quantum hardware itself. Due to these NISQ limitations, this thesis focuses on developing and evaluating QDMs using quantum simulators.

A positive externality of using a simulator is that it offers immediate access to the output probabilities from the Parameterized Quantum Circuit (PQC). The PQC provides 2^n probabilities, where n denotes the number of qubits, illustrating the likelihood of the quantum system being in each possible state. This set of probabilities is crucial to our current Quantum Diffusion Model (QDM). Conversely, on real quantum computer hardware, you would not have directly access to these probabilities. We will discuss potential solutions for non-simulator QDM implementations in later chapters.

The computational resources available for this thesis introduce constraints on the complexity of the QDMs that can be effectively trained and evaluated, as well as the size and complexity of datasets that can be used. Since home computer setups are used, training complex models and working with large datasets presents challenges in feasibility. This requires an inherent focus on QDMs designed with efficiency and resource constraints in mind.

Chapter 2

Background

This chapter introduces the fundamental aspects of quantum computing and diffusion models that are needed to understand how the QDM works. We begin by exploring quantum fundamentals, addressing the three key concepts of superposition, entanglement, and interference. Following that, we introduce the concepts behind diffusion models, breaking down the three phases this model is build upon: forward, reverse, and sampling. The chapter also includes a review of contemporary literature in the field of QDMs. This review aims to situate the discussed concepts within the wider academic field, offering insights into the practical challenges and potential advancements in quantum computing and diffusion models.

2.1 Quantum Fundamentals

The concept of computers was first envisioned by the English mathematician Charles Babbage, and the world's first functional program-controlled Turing-complete computer, the Z3, was made by the German engineer Konrad Zuse in 1941 [43]. At its core, a computer uses transistors, which are tiny electronic switches, to perform binary operations. Transistors can be in an "on" (representing a binary "1") or "off" (representing a binary "0") state. By arranging billions of these transistors in complex circuits within a chip, a computer can process information, execute logical operations, and store and retrieve data, thereby allowing it to run software and perform tasks. Moore's Law, coined by Intel co-founder Gordon Moore in 1965, describes the observation that the number of transistors on a microchip doubles roughly every two years, driving exponential growth in computing power while reducing the cost per transistor [36]. This prediction has largely held true for several decades, driving rapid advancements in technology and the digital revolution. However, as physical limitations are approached at the atomic scale, maintaining this pace has become increasingly challenging. To keep up with the ever-growing demand for computational power, a fundamental shift in computing paradigms is necessary. This is where quantum computing enters the picture.

The origin of quantum computing dates back to the 1980s, propelled by the pioneering work of figures like Yuri Manin and Richard Feynman [35]. Over the ensuing decades, what began as theoretical exploration has blossomed into tangible experimentation. The implementation of Grover's search algorithm in the late 1990s marked a pivotal moment, showcasing the first practical demonstration of quantum advantage—where quantum computers outperform classical counterparts in specific calculations [9, 18]. This breakthrough ignited a global race among corporations and research institutions to advance quantum computing technologies, recognizing their

potential ramifications across diverse domains, from materials science to cryptography. Perhaps surprisingly, these companies and institutions have made quantum computers publicly available. Anyone is able to create an account and access one of these supermachines through the cloud, something that amusingly has been referred to as being the equivalent of "*NASA letting you control a Mars rover from home*" [48]. The incredible accessibility of these quantum computers has led to a massive increase in interest in the quantum world. The potential applications of quantum computing hint at the profound secrets held within the quantum realm. To understand and harness these secrets, we must venture into the fundamentals of quantum mechanics.

Quantum mechanics delves into the behaviors of matter and energy at their smallest scales, where classical physics proves inadequate. As we push towards even smaller computer components, reliance on quantum technology becomes essential, deviating from traditional computing norms. Quantum computing operates on a paradigm vastly different from conventional logic gates. Rather than only increasing bit capacity or processor speeds, quantum tech unlocks a computational realm grounded in quantum principles [43].

A quantum computer possesses a memory capacity that vastly exceeds its physical size, and can process an exponential number of inputs concurrently. This stems from quantum superposition, where a qubit can exist in multiple states at once. This unique mechanism leads to a groundbreaking form of parallelism, which we will explore further in the following chapter. Its computations occur within the realm of Hilbert spaces [43], which is the vector space where quantum states reside. The inherent physical constraints of conventional computers and the potential of quantum computers to execute specific tasks, in theory, exponentially faster than their conventional counterparts [33] is what motivates the study of quantum computing.

Quantum physics, distinct from classical physics, introduces a set of unique principles that defy our everyday intuition. These principles—superposition, entanglement, and interference—offer capabilities unparalleled by the conventional laws of physics. When harnessed in the realm of computation, they pave the way for quantum computers to tackle problems that are insurmountably complex for classical computers. By leveraging these quantum properties, we stand at the precipice of a computational revolution, one that has the potential to redefine the boundaries of what is computationally achievable.

2.1.1 Superposition

In the domain of quantum physics, the principle of superposition stands as a cornerstone, challenging our classical intuitions about the nature of reality. The principle of superposition posits that a particle is capable of existing in multiple states simultaneously, a concept that defies the binary existence witnessed in the macroscopic world.

This concept is crucial for understanding quantum bits (qubits), the basic units of quantum information. Unlike classical bits in conventional computers, which are constrained to being either 0 or 1, qubits operate in a more fluid manner. They can exist in a superimposed state that spans any value between 0 and 1, with each possible state having a distinct probability of being realized upon observation. It is only upon the act of measurement that the particles superposition collapses, and it settles into a single, definitive state.

Bra-ket notation, also known as Dirac notation, is a standard representation used in quantum mechanics to describe quantum states. It provides a concise way to represent vectors in the Hilbert space. A "bra" is represented as $\langle\Psi|$ and corresponds to the dual

vector or adjoint of the state vector. A "ket" is represented as $|\Psi\rangle$ and corresponds to the state vector itself. For instance, a qubit's state is not limited to $|0\rangle$ or $|1\rangle$; it can be represented as a superposition of the two basic states

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{2.1}$$

where α and β are complex coefficients that encode the state amplitude probabilities of the qubit being in the $|0\rangle$ or $|1\rangle$ state respectively. The magnitudes of these amplitudes squared ($|\alpha|^2$ and $|\beta|^2$) provide the probabilities of measuring the qubit in either of these basis states, embodying the probabilistic nature of quantum mechanics. Moreover, the sum of these probabilities must adhere to the normalization condition, $|\alpha|^2 + |\beta|^2 = 1$, ensuring the total probability encompasses all potential outcomes.

Intriguingly, while an n-bit classical register can hold one of 2^n possible values, a quantum register possesses 2^n distinct quantum states, enabling it to concurrently represent a superposition of all 2^n states [32]. For example when we have two qbits, there are four possibilities: 00, 01, 10, 11. This can be described as

$$|\Psi\rangle = a\,|00\rangle + b\,|10\rangle + c\,|01\rangle + d\,|11\rangle\,. \tag{2.2}$$

2.1.2 Entanglement

Quantum entanglement, often described as the "*spooky action at a distance*" by Einstein [55], is one of the most profound and counterintuitive phenomena in quantum mechanics. It is a phenomenon where particles become intricately linked, such that the state of one particle influences the state of another, regardless of the distance separating them. This interconnectedness is a drastic shift from classical computing where the state of one bit remains independent of others.

In quantum computing, when qubits become entangled, they no longer exist in isolation but meld into a collective quantum state. Consider two entangled qubits. Their individual probabilities become intertwined, necessitating the calculation of a combined probability distribution. Instead of viewing them as separate entities with individual probabilities, we must consider the combined probabilities of them existing in states 00, 01, 10, or 11. This combined probability distribution reflects the likelihood of the qubits collapsing into each of these states.

A key insight into entanglement is its dynamic nature. Modifying the state of one qubit in an entangled pair impacts the entire probability distribution. Essentially, with n entangled qubits, there exists a probability distribution across 2^n states. A mere change to one qubit can influence the states of all entangled qubits. In contrast, a classical computer can exist in any one state at a given moment, while a quantum computer can exist in a superposition of multiple states simultaneously, showcasing the immense potential of quantum computation. Entanglement is a powerful resource, enabling qubits to exhibit correlations that are fundamentally beyond the capabilities of classical bits. This unique property is harnessed to perform complex computations more efficiently and to achieve levels of communication security impossible by classical means. Importantly, for quantum computers to achieve their most powerful advantage – exponential speedup – research by Jozsa and Linden demonstrates that entanglement must increase along with the size of the computation [25].

2.1.3 Interference

Building on the principles of superposition and entanglement, interference stands as the third key concept in quantum computation, weaving together the fabric of quantum

algorithms. This phenomenon is not just a byproduct of quantum mechanics but a fundamental mechanism that quantum algorithms exploit for their unparalleled computational capabilities. Interference occurs when the wavefunctions of two or more qubits interact, leading to a combined outcome. A quantum wavefunction is a mathematical tool describing a quantum particle's probabilistic existence as a spread of possibilities.

In quantum computing, interference is harnessed through quantum gates — the building blocks of quantum circuits. Each gate, defined by precise quantum operations, is akin to a mathematical matrix that manipulates qubit states, guiding them through a landscape of constructive and destructive interference. This interplay is not random but meticulously controlled, ensuring that quantum computations steer towards desired outcomes. This interference mechanism is pivotal; it is the key process enabling computations on quantum information. The true power of interference becomes evident when designing quantum algorithms. The objective is to strategically use interference to amplify the probabilities of correct outcomes while simultaneously diminishing the probabilities of erroneous ones.

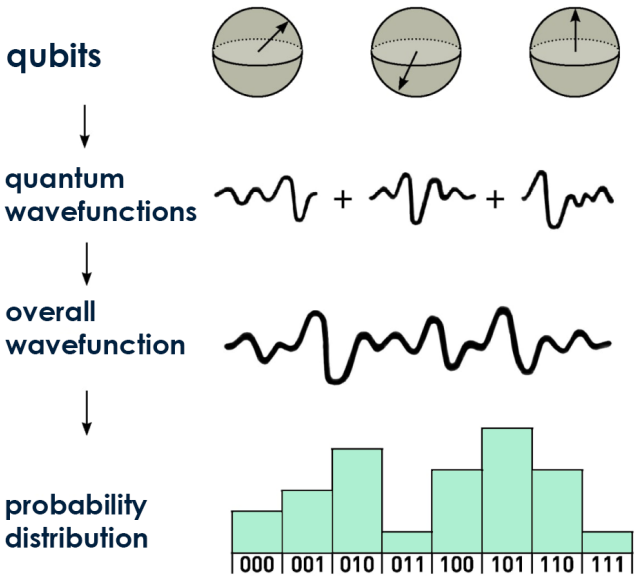


Figure 2.1: Illustration on how entangled qubits produces an overall wavefunction that changes the probability distribution of measurement outcomes. Inspired by [47].

2.1.4 Parameterized Quantum Circuits

A parameterized quantum circuit (PQC) is a concept from quantum computing that combines classical and quantum computation [3]. It is a key component in many quantum algorithms, especially in the field of quantum machine learning and quantum optimization. A PQC is composed of a fixed structure of both parametric and non-parametric unitary quantum gates that manipulate qubits. The word unitary comes from the fact that the gates are reversible, meaning that there is a way to "undo" the operation, which is crucial for certain quantum algorithms and error correction methods.

This reversibility is a fundamental characteristic of quantum operations, as it guarantees that quantum computations preserve the total probability of possible outcomes [56]. The parametric gates are also adjustable, allowing for the manipulation of quantum states based on specific parameters. These parameters can intuitively be compared to the weights in a classical neural network. Consequently, a PQC can be viewed as the quantum counterpart of a classical neural network.

An example of parametric gates is the rotation gates, a qubit rotation gate is a quantum gate that rotates the state of a qubit around a specific axis on the Bloch sphere by a certain angle [56]. The bloch sphere is a visual and mathematical representation of the state of a qubit in quantum mechanics [17]. The sphere's surface is representing all possible pure states of a qubit. On the Bloch sphere, three axes are used to define the position of the qubit state: the x-axis, y-axis, and z-axis. Therefore, we have three different types of rotation gates Rx, Ry, and Rz gates. The Rx and Ry gate can modify the probabilities of a qubit being measured in the $|0\rangle$ or $|1\rangle$ state, by rotating it around the x and y axis. The Rz gate is a bit more nuanced, as rotations around the z-axis change the relative phase between the $|0\rangle$ and $|1\rangle$ states without altering their direct measurement probabilities. This phase difference becomes crucial when the qubit interacts with others, especially in quantum algorithms involving interference and entanglement, where the relative phases between qubits can determine the outcome of the computation.

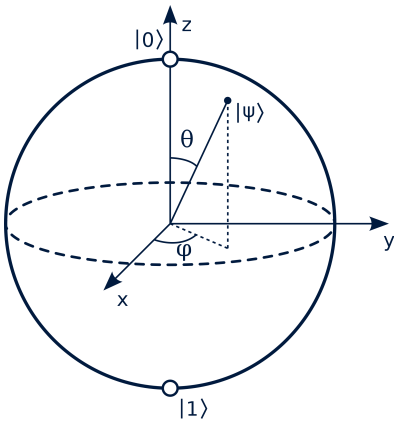


Figure 2.2: The bloch sphere. Source: [54]

On the other hand, the Hadamard gate and Controlled-NOT (CNOT) gate serves as examples of non-parametric gates. The hadamard gate sets the qubits in a an equal superposition of $|0\rangle$ and $|1\rangle$. The CNOT gate is a two-qubit quantum gate that flips the state of the second (target) qubit if the first (control) qubit is in the $|1\rangle$ state, effectively introducing entanglement between qubits in quantum circuits [56]. Collectively, rotation gates, the Hadamard gate, and the CNOT gate are the foundational elements of quantum algorithms, as these form a complete set that can be used to build any quantum circuit.

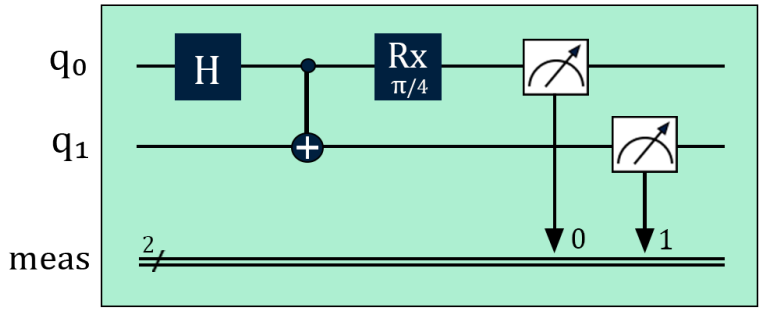


Figure 2.3: Two-qubit quantum circuit with measurement.

In Figure 2.3 a two-qubit quantum circuit schematic is displayed, showcasing a sequence of the three foundational quantum gates. First, a Hadamard gate puts qubit q0 into superposition. Next, a CNOT gate with q0 as the control and q1 as the target entangles the two qubits. If q0 is measured to be in state $|1\rangle$, the CNOT gate will flip q1. Due to this entanglement, measuring the system without any further operations would yield either 00 or 11 with equal probability. This can be mathematically expressed as

$$\frac{1}{\sqrt{2}}\left(|0\rangle_c |0\rangle_t + |1\rangle_c |1\rangle_t\right), \tag{2.3}$$

where c denotes the control qubit, and t denotes the target qubit. This is a maximally entangled state, known as a Bell State [42]. Then a Rx rotation gate with $\pi/4$ radians is applied to q0. This means that the qubit is being rotated around the x-axis of the Bloch sphere by an angle of $\pi/4$, that is 1/8 of the way around the axis. This rotation alters the probabilities of q0 being in the $|0\rangle$ or $|1\rangle$ state. Specifically, it introduces a non-zero probability for q0 to be found in the opposite state from its initial superposition components due to the rotation. This means that even though the CNOT gate would have resulted in q0 being measured as $|1\rangle$ with certainty, the subsequent Rx gate introduces a probability of measuring q0 as $|0\rangle$, and vice versa. Finally, both qubits are measured and then collapsed to a final state.

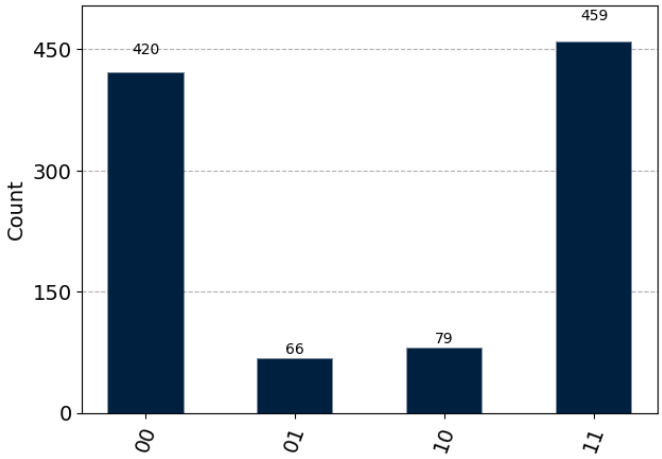


Figure 2.4: Circuit in Figure 2.3 simulated 1024 times.

In Figure 2.4 you can see the circuit simulated 1024 times. Notice how the rotation gates manipulated the circuit's outcome. This demonstrates the power of parametric

gates — a small change in a qubit's rotation can significantly alter the circuit's overall output. By fine-tuning the rotation angles, one can explore a vast landscape of quantum states, even minor adjustments to the parameters of rotation gates can lead to significant variations in the circuit's output probability distribution.

Consider the quantum circuit $U(w)$ in a PQC. It executes a unitary transformation on an initial quantum state $|x\rangle$, resulting in the final quantum state $|y\rangle$. Mathematically, this transformation is expressed as:

$$|y\rangle = U(w)\,|x\rangle\,. \tag{2.4}$$

Here, w represents the adjustable parameters in the circuit, such as the angles in qubit rotation gates. The operation of a PQC involves applying a unitary transformation to an input quantum state, followed by the measurement of the resultant quantum state. The outcomes of this measurement are analyzed by a classical computer to determine a loss value, which evaluates the quantum operation's performance or accuracy. Subsequently, in a process of iterative refinement, the classical computer adjusts the circuit parameters w, aiming to optimize them to meet a desired expectation value or until a maximum iteration limit is reached.

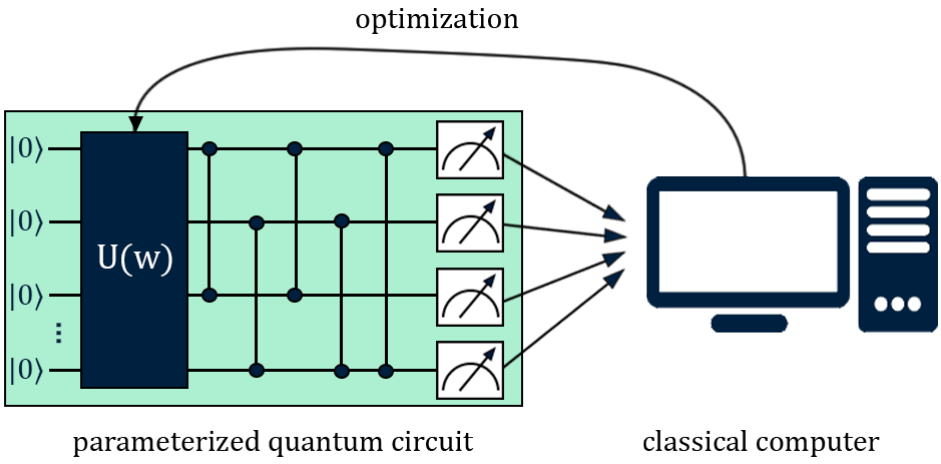


Figure 2.5: The quantum circuit $U(w)$ and its classical optimization loop. Inspired by [30].

PQCs are often designed with the intent of being robust against certain types of errors and noise, making them particularly suitable for implementation on NISQ devices [3, 44]. The robustness of PQCs to noise is attributed to their ability to be efficiently trained with a limited number of gates which reduces error accumulation, and their adaptability through training on noisy simulations or actual quantum hardware. This training process enables PQCs to potentially discover configurations that mitigate the effects of system noise.

2.1.5 Amplitude Embedding

Amplitude embedding is the process of converting classical data into the amplitudes of qubits. The key idea is to take advantage of the superposition principle in quantum mechanics to encode multiple classical data points into the amplitudes of a single quantum state, allowing for a potentially exponential compression of information compared to classical data representation. In other words, this technique can encode 2^n

classical data points using just n qubits. For successful embedding, the classical data vector $\mathbf{x} = (x_1, x_2, \ldots, x_{2^n})$ must be normalized so that the squares of the amplitudes sum to one: $\sum_{i=1}^{2^n} |x_i|^2 = 1$. This is what we referred to as the normalization condition in 2.1.1. This normalization is crucial due to the quantum mechanics principle that the total probability of all possible outcomes in a quantum system must equal one. This ensures that, upon measurement, the sum of the outcome probabilities — determined by the squares of the amplitudes — also equals one, confirming the occurrence of an outcome. Therefore, before embedding, classical data often requires preprocessing to meet this normalization condition, making the quantum state valid and normalized for quantum computations. The transition form an initial quantum state to the desired state, necessitates a carefully orchestrated sequence of quantum gates. This sequence is tailored to manipulate the amplitudes and phase of the quantum state in accordance with the encoded classical data. Figure 2.6 visualizes that 2^n data points gets encoded into n qubits.

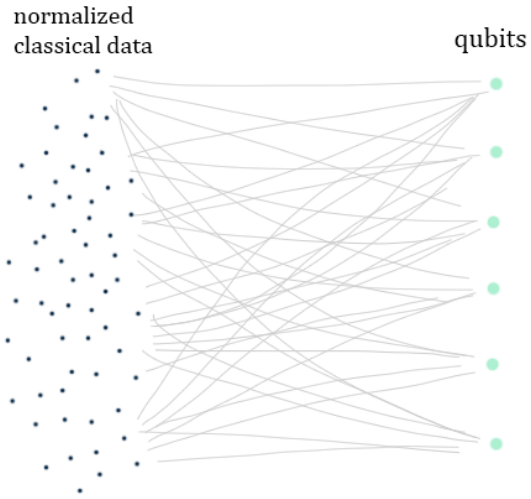


Figure 2.6: Amplitude embedding visualised.

2.2 Diffusion Models

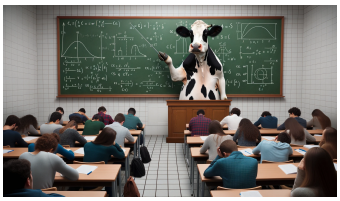
Diffusion models, first developed by Sohl-Dickstein et al. in 2015 [49], represent a breakthrough in generative machine learning. Their core principle is deceptively simple: systematically corrupting data and then learning to reverse the corruption process. This approach tackles the long-standing challenge of balancing tractability and flexibility within machine learning models.

Diffusion models have become a powerful tool within computer vision tasks, noted for their ability to generate diverse and high-quality images (as seen in Figure 2.7). Their straightforward training process, architectural flexibility, and capacity to produce varied results contribute to their increasing adoption within the field. Their recognition by both the academic community and the broader public indicates the significant role diffusion models are likely to play in advancing digital creativity and technological innovation [12]. Additionally, they offer practical advantages in their ease of training and adaptable architecture [12, 13].

The generated images in Figure 2.7, while almost impossible to tell apart from a real photograph (given that we ignore the absurdity of a standing cow giving a lesson in physics in), does exhibit some slight imperfections. These imperfections are commonly

(a) Photo of a Poodle and a Rottweiler, both wearing tiny glasses, deeply engrossed in a game of chess. The chess board is on a rug in a cozy room, with a fireplace glowing in the background.

(b) Photo of a rabbit wearing a space suit inspired by modern space exploration designs, standing against the backdrop of a starry space. The suit is sleek, white, and has minimalist design elements.



(c) Photo of a cow standing at a podium with a chalkboard behind, explaining complex physics equations. The classroom is filled with students, but most of them are slumped in their chairs, asleep.

Figure 2.7: Images generated by DALL-E 3 [40] based on text prompts.

referred to as *artifacts*. For instance, several artifacts can be found in Figure 2.7a: looking at the chess board, the white player has three rows of pieces, some of the pieces looks like they are the product of a bishop and a knight (a *knishop*, perhaps?), and the board itself is imperfectly colored. It definitely conveys the concept of a chess board, but the distorted details and unusual pieces betray its artificial origins.

Diffusion models fall within the machine learning domain, categorized as a *self-supervised*, *latent variable*, and *generative* model. Self-supervision means that the model itself generates labels for the input data. Generative means that the model tries to understand the data it is being trained on, with the goal of generating new synthetic data that is similar to the training data. Latent variable models are designed to capture hidden or unobservable factors within the data, enabling the model to learn complex data distributions and underlying structures that are not directly observable. In summary, a diffusion model is being fed a number of images as its input data, learns the patterns and underlying distributions of those images, and is then able to generate similar images.

Diffusion models are split into three main components: forward, reverse, and sampling.

Forward In the forward step of diffusion models, input data undergoes a series of transformations where noise is added incrementally at each stage, progressively degrading the original data. This process introduces a concept known as entropy, a measure of randomness or disorder within a system. As noise is added, the entropy of the data increases, moving the system further from its original state and closer to a state of maximum disorder. At the end of the forward step, the original data is unrecognizable.

Reverse In the reverse phase the primary objective is to eliminate the noise introduced during the forward phase, thereby restoring the data to its original distribution. This entails a reduction in the entropy, signifying a systematic reduction in randomness and disorder. The process demands that the model discerns the underlying structure of the data from its highly entropic states.

Sampling During the sampling phase, diffusion models initiate with a purely noise-generated distribution, bypassing the forward phase entirely. This noise distribution is then subjected to the reverse phase, where the model employs a series of learned transformations to progressively remove the noise. This step-by-step refinement facilitates the gradual emergence of coherent data structures.

> Unlike in the reverse phase, where the aim is to recover pre-existing data, the sampling phase is characterized by the generation of entirely new data instances. These instances, though novel, are influenced by the statistical properties of the data on which the model was trained.

The concept behind the model was inspired by thermodynamic principles - especially the process of diffusion, from which it got its name [49]. In thermodynamics, diffusion is characterized by the stochastic movement of particles from regions of high concentration to regions of lower concentration. Over time, the system will have been diffused to the point where all the particles are equally concentrated. This is called the equilibrium state, and the entropy of the system will be at its peak during this state.

This approach offers a unique perspective on data generation, relying on the gradual destruction and reconstruction of data. Although adding noise to input data is not a novel technique in machine learning - it has been used in data augmentation for neural network training [22] and as a regularization technique [6] for three decades - diffusion models are unique in their utilization of noise. The aforementioned techniques have been introduced in order to increase model robustness and their susceptibility to noise. In other words, they are trained to *ignore* the noise, and the noise itself is limited to auxiliary purposes. Diffusion models, on the other hand, are trained to *learn* the noise, in order to have the capability to remove it.

2.2.1 Forward

The forward step of a diffusion model is a fixed, stochastic process that transforms an initial data distribution into a sequence of intermediate states and ultimately into a final sample through a series of τ discrete time steps. This sequence can be denoted as $x_0, x_1, ..., x_\tau$, where x_0 is the original data and x_τ is the most noisy state. In each time step t, indexed by $t(0 \leq t \leq \tau)$, it adds a controlled amount of noise ϵ_t to x, resulting in a new state x_{t+1}.

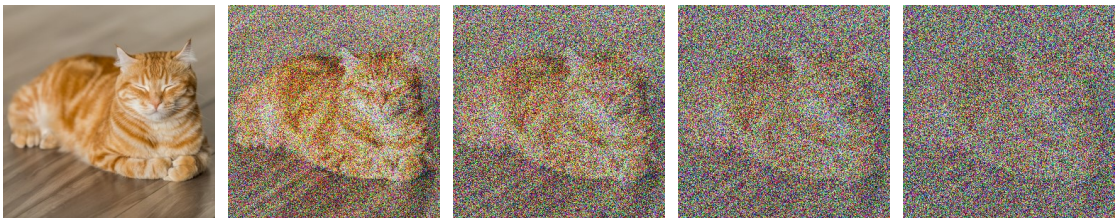
The added noise ϵ_t is typically *normal*, also referred to as *Gaussian*, noise. Gaussian noise is simply noise that follow a Gaussian distribution; its probability function is equal that of a Gaussian distribution.

There are variations to this process, or more specifically in regards to how much noise is added at each time step. This is nicknamed the noise or β_t schedule. The schedule details how much noise is to be added at every time step t, and can be both linear and non-linear functions. In the original diffusion model paper, for instance, the authors used a linear β_t schedule [49], but further search by Nichol et al. has proved other schedules could be more optimal [39]. For both linear and non-linear β_t schedules, the next phase in the chain can be expounded as:

$$x_{t+1} = x_t + \epsilon_t \tag{2.5}$$

Consequently, although dependant on the β_t scheduler and the intensity of the added noise, the final sample x_τ would be almost pure noise. An example of this forward process can be found in Figure 2.8.

It is important to note that any arbitrary state x_t is derived solely from its predecessor x_{t-1}. Thus, the sequence of $x_0, x_1, ..., x_\tau$ forms a kind of chain. Additionally, the sequence is a memoryless process that conditions on the previous state to generate the next state - the next state is independent of the previous history. This means that the states adhere to the Markovian property, and altogether form a Markov chain.



(a) $t = 0$ (b) $t = 10$ (c) $t = 20$ (d) $t = 30$ (e) $t = 40$

Figure 2.8: A visual example of the forward process.

2.2.2 Reverse

The objective of the reverse phase in a diffusion model is to iteratively undo all of the data degradation that happened in the forward process. At each iteration of the reverse process, the model focuses on a single noisy data x_t with the aim of reconstructing the less noisy version x_{t-1}, moving step-wise towards the original image x_0. Typically, this process involves predicting either the previous state x_{t-1} or the noise ϵ_t that was added to x_{t-1} in order to obtain x_t during the forward process.

We can extend this with a more mathematical exposition. Given a sequence of data $x_0, x_1, ..., x_\tau$ obtained during the forward diffusion process, the reverse process seeks to iteratively recover the sequence $x_{\tau-1}, x_{\tau-2}, ..., x_0$ where x_0 represents the original, noiseless data. Mathematically, this can be represented as:

$$x_{t-1} = f_\theta(x_t, t, \epsilon_t) \tag{2.6}$$

where f_θ is a parameterized function, typically a neural network with parameters θ, aimed at predicting the noise ϵ_t added at step t or directly predicting x_{t-1} from x_t. The function f_θ is trained to minimize the difference between its output and the actual previous step image x_{t-1}, effectively learning to reverse the diffusion process.

The learning is achieved by training the model to minimize a specific loss function that quantifies the discrepancy between the predicted and the actual x_{t-1} at each reverse step. A commonly used loss function in this context is the mean squared error (MSE) between the predicted clean image and the actual clean image at each time step. More specifically, the model takes x_t and t as inputs and outputs an estimate of x_{t-1}. The model is then trained to minimize the discrepancy between its estimate and the actual x_{t-1}. Alternatively, the model can aim to predict the noise ϵ_t that was added to to x_{t-1} during the forward process, like explained in Equation (2.5).

2.2.3 Sampling

The sampling phase enables the model to generate new, novel images. Independent of any forward process, it instead generates a random noise distribution on the same forward that the model has trained on. The crux of the sampling phase lies in the ability to refine a random noise input into a coherent image that aligns with the learned data distribution.

The process initiates with x_τ, a sample drawn from a standard Gaussian distribution, $N(\mu, \sigma^2)$, where μ denotes the mean of the distribution and σ is the standard deviation. Squaring the standard deviation μ^2 gives us the variance of the distribution. This x_τ sample represents the most entropic state within the model's learned distribution - essentially pure randomness.

Based on this sample, the model feeds this through the reverse process. Since this sample is not based on some data x_0, but is rather a pure Gaussian, the model uses the learned reverse mappings to transform pure noise into diverse and realistic images.

There are two notable variants of diffusion models in the field of generative modeling: the Denoising Diffusion Probabilistic Models (DDPMs) [20] and the Deterministic Denoising Diffusion Implicit Models (DDIMs) [50]. Their difference primarily lies in the reverse process and the underlying mechanisms through which they transition from a noise distribution to a structured data distribution. While DDPMs employ a stochastic approach, introducing randomness at each step to ensure a broad exploration of the data, DDIMs has a deterministic strategy, achieving a more direct and computationally efficient path through the latent space.

DDPM

The defining feature of DDPMs is the stochastic nature of their reverse generation process. Proposed in 2020 by Jonathan Ho et al. [20], the DDPM reverse process involves reintroducing noise at each step and then applying a learned denoising function to remove this noise, progressively moving from a state of pure noise towards structured data. This stochasticity gives the model the ability to explore the data distribution and generate a diverse array of outputs. The randomness introduced at each step ensures that even when starting from the same noisy input, the reverse process can yield different results, contributing to the model's generative diversity. The training objective, alike other diffusion models, involves minimizing the discrepancy between the noisy distribution at each reverse step and the distribution of the denoised outputs produced by the model. This training approach, focused on iteratively denoising and introducing stochasticity, contributes to the robustness of DDPMs in generating diverse samples. The model learns not just to invert the diffusion process but to handle a range of noise levels, enhancing its generative flexibility. The stochastic nature of DDPMs, while beneficial for diversity, necessitates a careful balance in the step size and total number of steps to avoid diverging from the target distribution or erasing critical details. Increasing the number of steps can lead to higher quality and more diverse outputs but at the cost of increased computational time and complexity.

DDIM

DDIMs, proposed by Jiaming Song et al. in 2022 [50], on the other hand, offer a deterministic approach to the reverse diffusion process, allowing for a more predictable and often faster path from noise to data. This deterministic nature makes DDIMs particularly useful for applications where consistency and control over the generated samples are desired, such as in image editing or text-to-image generation tasks, while maintaining decent-level quality outputs and a much faster sampling process. DDIMs utilize a specific formulation that allows for an explicit calculation of intermediate latent variables without needing to simulate the entire Markov chain. This is possible because, in a deterministic setting, the state at any point in the reverse process can be calculated directly from the initial noisy image and the original noise. This is thanks to the *normal sum theorem*, which states that the sum of two statistically independent normal variables yields another normal variable [28]. Consequently, any sum of independent Gaussian processes throughout the steps remains Gaussian. This property enables the model to sample any arbitrary x_t directly in closed form, mitigating the need to simulate the entire Markov chain. The ability to directly compute x_t without sequentially processing each

diffusion step offers a considerable computational advantage. It not only accelerates the reverse process but also ensures that the deterministic path from the noise to the data distribution is both consistent and predictable.

2.2.4 Number of Timesteps

In the preceding sections, we outlined the mechanism by which diffusion models iteratively apply and then reverse noise over a series of τ steps, effectively transitioning from a pure noise distribution back to data resembling the target distribution. This process, governed by the hyperparameter τ, is central to the operation and performance of diffusion models.

The role of the τ variable in diffusion models cannot be overstated, as it directly influences both the quality of the generated images and the computational resources required for model operation. A higher τ value allows for a more gradual transition from noise to coherent images. This incremental denoising process can potentially lead to higher-quality and more complex outputs by providing the model with more opportunities to refine and correct the generated image at each step [34]. However, the benefits of increased image quality with higher τ values come at the cost of computational efficiency. Each step in the diffusion process requires a forward pass through the model, meaning that a higher τ significantly increases the computational load and time required to sample an image.

Conversely, a lower τ value implies fewer steps in the noise addition and removal process, leading to faster image generation. While this increase in efficiency is desirable, especially for applications requiring rapid generation, it may compromise the nuanced details and overall coherence of the generated images. The reduction in steps affords the model fewer opportunities to refine the image, which can result in outputs that are less detailed or exhibit artifacts.

It is also relevant to mention that τ does not increase the "amount" of noise that is added to the data. For instance, any arbitrary data x will be similarly degraded at x_τ no matter if $\tau = 5$ or $\tau = 50$. An increase in τ instead increases the number of increments and lowers the increment sizes. A visual representation of this can be found in Figure 2.9.

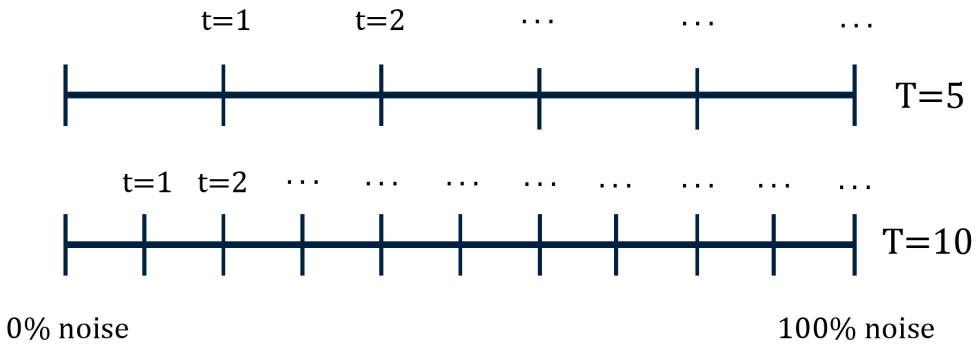


Figure 2.9: Visualization of τ increments.

In practice, the choice of τ is often a result of empirical experimentation and optimization, guided by the specific requirements of the application and the computational resources available. Recent research in diffusion models has explored various strategies to optimize τ, including dynamic scheduling methods that adapt the

number of timesteps based on the characteristics of the input data or the desired quality of the outputs [29].

2.2.5 Embedding

Data embeddings in the context of diffusion models serve as compact, high-dimensional representations of data, capturing underlying patterns and semantics. These embedding vectors allow diffusion models to efficiently process and generate complex data structures by providing a structured way to incorporate diverse information, such as temporal dynamics and categorical attributes. The advantage of using embeddings is their ability to enhance model performance by enabling a more nuanced understanding and manipulation of data.

Embedding vectors are essentially numerical representations of specific data features. For example, in text-to-image diffusion models, words or phrases can be transformed into embedding vectors. The model embeds data by appending embedding vectors directly to the existing representation. This combined representation provides the diffusion model with richer contextual information, improving its ability to generate outputs that align with the desired characteristics. The size of these embedding vectors is an important parameter, as larger embeddings can potentially encode more detail but also increase computational costs.

Timestep Embedding

In Section 2.2.1 we described how a diffusion model iteratively applies noise to the data over a series of *time steps*. In Section 2.2.2 we explained how the model learns to remove this noise. However, the model does not known which timestep it is trying to remove the noise from. Having access to this information is crucial because the characteristics of the noise being removed vary depending on the stage of the reverse process. For instance, while early stages involve removing high-level noise, later stages may focus more on fine-tuning and subtle details. To solve this issue, we will introduce the concept of timestep embedding.

Time embedding adds a "time"-dimension to these models, giving the model context at each step of the reverse diffusion process. Technically, timestep embedding is often implemented using a positional encoding mechanism similar to those used in transformer models [52]. These embeddings allow diffusion models to understand the sequence order and the relative positioning of data points, enhancing their ability to reverse the diffusion process effectively.

This involves mapping discrete time steps to a continuous, high-dimensional space, which are able to capture complex and nonlinear relationships between the various stages of the diffusion process. This enables the model to adapt its denoising strategy according to the specific stage of the process. This is particularly important in models that operate over a large number of diffusion steps, where the precise adjustment of the denoising operation at each step can significantly impact the quality of the final output.

In summary, time embedding in diffusion models serves as a fundamental mechanism for encoding temporal information, allowing the model to effectively tailor its denoising operations to the specific stage of the reverse diffusion process.

Label embedding

In Section 2.2 we categorized diffusion models as self-supervised. Although that still holds true, diffusion models can also be supervised through a process called label

embedding.

Label embedding is a technique used to condition the generation process, such as class labels or attributes, to guide the model towards generating data that conforms to desired characteristics. In other words, it allows for controlled generation of data. These models are often referred to as conditional models. The most common use of label embedding in diffusion models is text-based prompting, where textual descriptions guide the image generation process. Label embedding is crucial in diffusion models; without it, generated images would become indiscriminate blends of the entire training dataset. Adding label embeddings inherently changes the problem the model is trying to solve, shifting the focus from learning the general data distribution to learning how to generate image variations that align with given classes.

The first step involves embedding the labels or attributes into a continuous vector space. This is typically done using an embedding layer. The resulting label embeddings capture the semantic information of the labels in a form that the diffusion model can utilize. The key idea is then to incorporate the label information at each step of the generation process, either by concatenating the label embeddings with the input noise or the features, or by modulating the parameters of the diffusion model neural networks based on the label embeddings. [16, 21]. During training, the model learns to reverse the diffusion process not just from noise to data but from noise conditioned on label embeddings to data that corresponds to those labels. When generating new samples, label embeddings are provided to the model along with the initial noise, resulting in output that reflects the characteristics or categories represented by those labels.

By using label embeddings, a single diffusion model can be used to generate a wide variety of data types or styles, simply by changing the labels used during the generation process. Conditioning on label embeddings can help the model learn more robust and discriminative features, as it has to learn to correlate specific features in the data with the information contained in the labels.

2.3 Existing Quantum Diffusion Models

The paper "Quantum-Noise-driven Generative Diffusion Models" by Marco Parigi et al. [41], is a notable reference in the development of quantum diffusion models. It proposes three quantum variations of Generative Diffusion Models (GDM): Classical-Quantum (CQGDM), where the forward process uses classical noise and the reverse process employs a Parametrized Quantum Circuit (PQC) for denoising; Quantum-Classical (QCGDM), where the forward process uses quantum noise and the reverse process utilizes a classical neural network; and Quantum-Quantum (QQGDM), where both the forward and reverse processes are quantized. While the paper introduces these three models, it mentions that only two are theoretically feasible, highlighting the impossibility of training a classical neural network to perform the denoising on an entangled quantum noise distribution in the QCGDM. Specifically relevant to this thesis is the CQGDM model, which initially sparked our interest and guided our early development of our own quantum diffusion model. For those new to quantum machine learning, especially in quantum diffusion models, this paper is recommended as a great starting point.

In the work titled "Quantum Diffusion Models" by Andrea Cacioppo et al. [7], the authors present a quantum version of generative diffusion models that incorporate parameterized quantum circuits (PQCs) in place of traditional artificial neural networks. This approach directly generates quantum states and offers variants including a full

quantum model and a latent quantum model, with each model conditioned for specific tasks. A key aspect of their methodology is the use of amplitude embedding to encode classical information into quantum states. This encoding is integral to their process, involving the representation of a classical vector components as coefficients of a quantum state. This technique, amplitude embedding, was particularly influential in shaping our research, and we adopted a similar approach for encoding data in our quantum diffusion models. Our subsequent research confirmed that this method is highly effective in preparing data for the parameterized quantum circuits, making it a cornerstone of our model development.

In their paper "Quantum Denoising Diffusion Models," Michael Kölle et al. [26] explore the integration of quantum machine learning with denoising diffusion models to enhance image generation. They address significant challenges faced by classical diffusion models, such as slow sampling speeds and extensive parameter requirements. By introducing quantum diffusion models, specifically the Q-Dense and QU-Net architectures, they demonstrate enhanced image generation capabilities with fewer parameters compared to classical models. They also highlight their unitary single-sample consistency model architecture, which optimizes the sampling time by condensing the diffusion process into a single step. They validate through empirical testing on datasets like MNIST, Fashion MNIST, and CIFAR-10.

The research presented in "Quantum Generative Diffusion Model" by Chuangtao Chen and Qinglin Zhao proposes a novel quantum generative model, the Quantum Generative Diffusion Model (QGDM) [8]. A central aspect of their work is the introduction of timestep embedding, a technique for incorporating information about the diffusion process's current stage into the quantum state itself. This allows the model to effectively track the noise-induced corruption and guide the subsequent denoising process. In their model, depolarizing channels are utilized to introduce noise, mimicking realistic physical noise processes within the quantum system. The authors showcase the effectiveness of their approach through simulations generating quantum states up to 4 qubits. Additionally, they propose a resource-efficient version, RE-QGDM, which tackles problems involving up to 8 qubits while maintaining impressive generative capabilities.

Chapter 3

Designing a QDM

This chapter provides an overview of our QDM, setting the stage for the detailed implementation discussed in Chapter 4. Here, we aim to share the thought process that guided us from the initial concept to the development of a functional model. We will outline the key steps and considerations involved in designing the QMD, providing insight into the techniques and strategies adopted during its creation.

3.1 Model Selection

Our model design decisions were guided by an analysis of the literature on quantum diffusion models outlined in Section 2.3. Building upon the classical-quantum (CQGDM) model proposed by Marco Parigi et al. [41], we recognized the potential of a Parameterized Quantum Circuit to effectively model the reverse diffusion process. This insight, coupled with insights from Andrea Cacioppo et al.'s work on "Quantum Diffusion Models" [7], led us to adopt a classical-quantum architecture as shown in Figure 3.1.

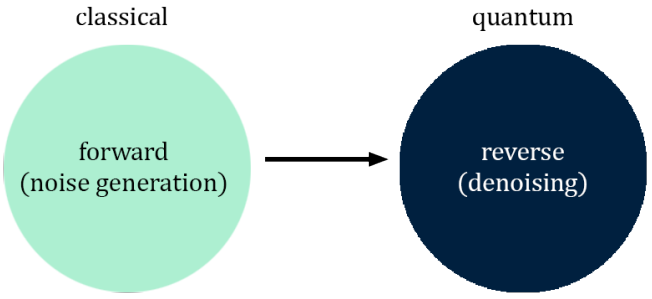


Figure 3.1: A simplistic overview of the model.

Specifically, we use a PQC in place of traditional neural networks in the reverse step. Based on Section 2.1.4, we concluded that these circuits are intently designed to be robust against errors and noise, making them suitable in the current NISQ-era. While our experiments utilize quantum simulators (as clarified in Section 1.2), exploring the theoretical feasibility of PQCs in this context lays the groundwork for potential future implementation on real NISQ devices.

For the sampling phase, we decided on a Denoising Diffusion Implicit Model (DDIM) [50]. This choice was made in light of the unique constraints and requirements associated with quantum computing simulations. As we explored in Sections 2.2.3 and 2.2.3, DDIMs

offer several advantages in this context. Firstly, they provide significant computational efficiency gains compared to DDPMs by bypassing the need to model noise at each timestep, as we discussed in Section 2.2.3. This efficiency is crucial for quantum simulations, which are inherently computationally demanding. Secondly, Nichol & Dhariwal demonstrated that DDIMs outperform DDPMs in generating high-quality samples using fewer than 50 sampling steps [39]. This characteristic aligns particularly well with our model's usage of a limited number of timesteps, allowing for efficient generation without sacrificing sample quality. Finally, due to the technical limitations outlined in Section 1.2, we knew we were going to use simple, low-dimensionality data for training. This may possess less inherent complexity than those with a wider range of image types. DDIMs align well with such datasets, allowing for efficient image generation without the extensive exploration that DDPMs offer.

3.2 Initial Approach

Our initial objective was to evaluate whether a PQC could learn to reconstruct a known label image when presented with a randomly generated noisy image. This process mirrors a single T step within a diffusion model and aimed to assess the PQC's ability to learn image distributions sufficiently for denoising tasks. Successful reconstruction would demonstrate the PQC's potential for denoising applications within diffusion models, extending its capabilities beyond simple image recreation.

We designed a "smiley" training label, composed of black and white pixels on an 8x8 grid. Since quantum circuits generate outputs of 2^n qubit probabilities (64 in this case) with a sum of 1 (as explained in section 2.1.5), we normalized the label to align with these specifications. This normalization enabled direct comparison between the label and the quantum prediction. Our goal was to optimize the PQC's weights such that it would consistently output the "smiley" pattern when given any noisy input image. Figure 3.2 shows how the training label looks when plotted into a grid.

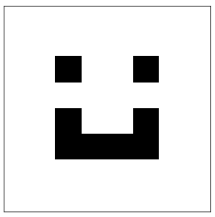


Figure 3.2: The smiley.

In this initial approach, we compressed input data from n pixels to $log_2(n)$ data points using 1 to 3 fully-connected sequential layers. Experiments demonstrated the PQC's ability to learn the "noise to smiley" task, evidenced by a consistent decrease in the loss function over training epochs. This proved the PQC a suitable neural network replacement.

While successful, our research suggested the potential for more effective data compression using quantum-specific techniques. In Section 2.1.5, we detailed amplitude embedding, a technique for encoding classical data into the amplitudes of a quantum state. This technique encodes n features into the amplitude vector of $log_2(n)$ qubits, using superposition to minimize information loss. Here, each quantum state amplitude corresponds to an input data feature, theoretically allowing a full quantum state to

encode up to 2^n features. Moreover, Andrea Cacioppo et al.'s successful use of this technique suggests its strong suitability for image-based tasks, similar to our own research [7].

Figure 3.3 compares the results of using fully-connected sequential layers and amplitude embedding after 10 training epochs.

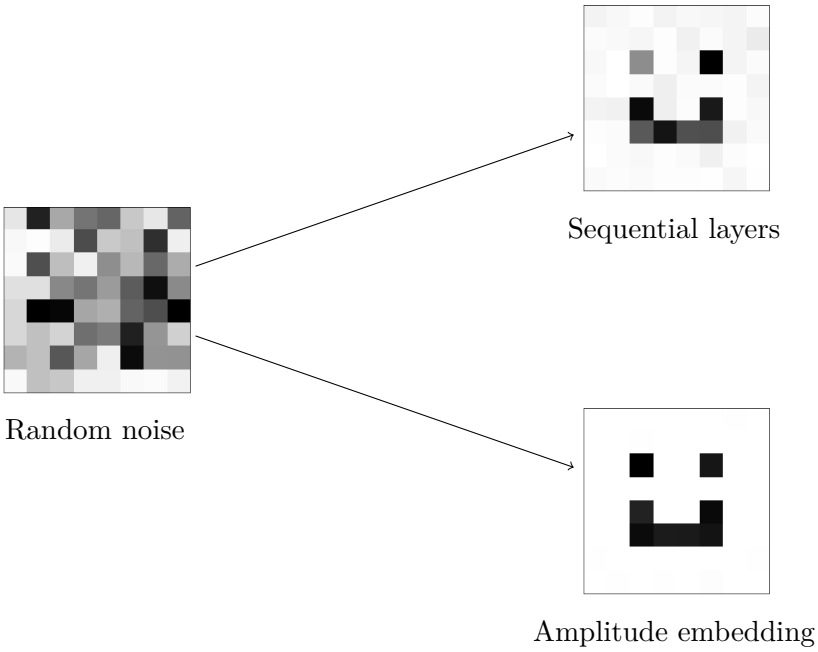


Figure 3.3: A comparison between the prediction of sequential layers and amplitude embedding.

As the figure clearly demonstrates, amplitude embedding yields superior results compared to the sequential layer approach. The prediction is nearly indistinguishable from the label, validating its effectiveness for our model due to its strong performance and efficient data compression.

3.3 Incorporating Timestep Embedding

We have adapted a well-established technique from classical diffusion models, known as timestep embedding, detailed in Section 2.2.5. This method informs the model of its current position within the diffusion timeline, enabling more precise learning and denoising at each specific timestep.

To implement timestep embedding in a QDM, we employ an ancillary qubit. This qubit undergoes rotation to a predefined orientation corresponding to the particular timestep the model is addressing. During the sampling phase, we apply this precise rotation to the ancilla qubit at each step, ensuring the model accurately interprets the stage of the diffusion process it is engaged in.

This is effectively acting as an embedding vector similar to those used in classical diffusion models, as explained in Section 2.2.5. However, a crucial difference lies in the integration method. In classical models, the embedding vector is simply concatenated with data. In contrast, within the quantum realm, the ancilla qubit is directly integrated with the main qubits, influencing the underlying quantum state and its evolution throughout the diffusion process.

Interestingly, our approach to timestep embedding differs from the technique proposed in "Quantum Generative Diffusion Model" by Chuangtao Chen & Qinglin Zhao [8]. Their work also implements timestep embedding for QDMs, but utilizes a more complex circuit implementation. We will explore their implementation in greater detail in Section 6.2.5.

3.4 Guiding Generation with Conditional Labels

To enable our model to train on a diverse set of images, such as different numerals, we extend the embedding technique to incorporate conditional labels, as detailed in Section 2.2.5. This is essential to prevent the model from producing ambiguous mixtures of the training data. For instance, if trained exclusively on images of zeros and ones, the model might generate results that are neither zero nor one without a guiding mechanism.

To implement label embedding, we again use an ancillary qubit. This qubit is rotated in a specific direction corresponding to the label of the training image. This rotation is integrated with the main data-carrying qubits, altering the quantum state. During sampling, we similarly adjust the ancilla qubit based on the desired output, effectively "steering" the sampling process.

3.5 RGB Encoding

Our initial design for the QDM focused on grayscale images. However, a transformative discussion among our research team identified the potential to extend the model's architecture to accommodate RGB image processing. This realization stemmed from the understanding that adding merely two more qubits could quadruple the probabilistic resolution of the model, aligning with the ternary structure of RGB color data.

We proceeded to integrate this concept into our framework and successfully adapted our model to handle color images. This development represents an advance, as we believe it to be the first implementation of an RGB-capable quantum diffusion model. It highlights the potential of quantum computation for pushing the boundaries of image processing even in this rudimentary NISQ-era.

3.6 Model Variations

Our research explores four key variations of our Quantum Diffusion Model (QDM). These variations build upon the core QDM architecture, extending its capabilities through selective embedding strategies:

The Base Model This foundational model serves as the starting point. It is designed for learning from a designated dataset and generating new, synthetic data that reflects its training.

The Temporal Model This variation incorporates timestep embedding by employing an ancilla qubit. Embedding the current timestep within the diffusion process allows the temporal model to gain a precise understanding of noise patterns, leading to more accurate denoising and improved image generation.

The Conditional Model This variation introduces label embedding, also using an ancilla qubit. By encoding label information corresponding to categories or

identifiers, the conditional model can generate diverse images tailored to specific input instructions.

The Hybrid Model This variation combines the strengths of both the temporal and conditional models. Using two ancilla qubits, it integrates timestep and label embedding, allowing for the generation of precise, targeted images based on both temporal context and specific labels.

Chapter 4

Implementation

In Chapter 3, we outlined the conceptual framework and design of our QDM. This chapter explores its technical realization, demonstrating how we implemented the QDM in practice. Here, we will discuss the integration of classical and quantum computing techniques as they underpin the model's functionality. We will cover noise generation, PQC optimization, the use of ancilla qubits, quantum post-processing, and both training and sampling cycles. In this setup, classical algorithms introduce noise (the forward step), while a PQC is used for the denoising process (reverse step), making a classical-quantum diffusion model. The PQC is refined through classical optimization methods. The PQC itself is refined through classical optimization methods. Once trained, the quantum circuit generates synthetic images after τ iterations, requiring only minimal post-processing.

4.1 Tools and Technologies

In developing our QDM, we have chosen the PennyLane framework [59]. PennyLane is a cross-platform Python library specifically made for quantum machine learning and quantum computing. It stands out for its impressive integration with machine learning techniques and its versatile compatibility across a wide range of quantum computing backends. This gives us access to the strengths of both quantum computing and classical machine learning within a single programming ecosystem. PennyLane connects quantum computing with machine learning frameworks like NumPy's autograd, JAX, PyTorch, and TensorFlow, making them quantum-aware. Moreover, PennyLane's differentiable programming paradigm ensures that our model is not confined to a single quantum hardware platform but can be executed on various quantum backends, providing us with the flexibility to adapt to the rapidly evolving quantum computing landscape [4].

Despite being in the NISQ era, as outlined in Section 1.2, with its inherent hardware limitations, we have opted to train and deploy our models using noiseless simulators. This decision reflects the early stage of QDM and quantum machine learning, allowing us to prioritize establishing a strong theoretical groundwork. Using noiseless simulators allows us to isolate and address the theoretical and algorithmic challenges without the additional variable of hardware-induced noise.

4.2 Dataset Overview

In this thesis, we will mainly be working with the iconic MNIST dataset. This dataset has a long history in machine learning, making it a foundational resource in the fields

of image recognition and generative modeling. While seemingly simple, its collection of handwritten digits (0 to 9) offers a surprisingly powerful way to train and evaluate machine learning models specialized for image-related tasks. This makes it a great starting point to understand core concepts.

The dataset consists of 70 000 images, each with a standard resolution of 28x28 pixels. This provides a substantial volume of data for robust training and testing of machine learning models. The diverse handwriting styles present within these images offer a wide array of digit representations. This diversity is crucial for training the diffusion model to generalize its understanding of handwritten digits, enabling it to effectively handle the natural variability found in real-world examples.

When conducting experiments in Chapter 5, we reduced the resolution from 28x28 to 16x16 pixels to conserve qubits and accelerate training times. Figure 4.1 showcases examples of the downscaled dataset.

Figure 4.1: The MNIST dataset: ten examples with labels 0-9 respectively, downscaled to size 16x16.

To explore the capabilities of our model with RGB images, we created two custom datasets: "rainbows" and "flags". Each dataset comprises 16x16 color images. The "rainbows" dataset features 10 variations of rainbow-colored images, as illustrated in Figure 4.2. The "flags" dataset features the Norwegian and Swedish flags, and serves to demonstrate the model's capability to differentiate training labels within colored images. The images are displayed in Figure 4.3.

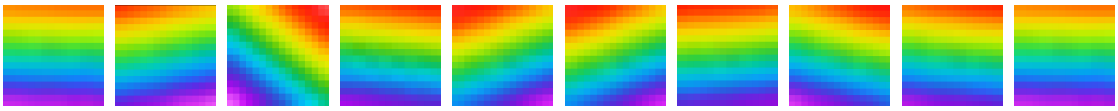


Figure 4.2: The custom "rainbows" dataset: 10 distinct rainbow-colored images used to train and evaluate the model's ability to handle color data.

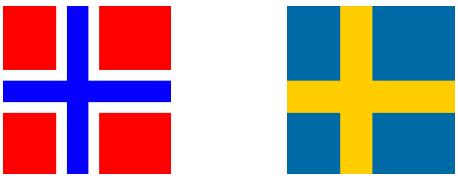


Figure 4.3: The "flags" dataset.

4.3 Noise

We are training our model with a Gaussian (or normal) noise distribution. Since the image pixels are normalized between 0 and 1, we have set the mean of the noise to be 0.5, so we get an even distribution of noise, where values close to 0 will indicate a light pixel, and values close to 1 indicates a dark pixel. The same applies for RGB images where the color channel values range from 0 (absence of color) to 1 (maximum color intensity).

The σ (standard deviation) of the distribution on the other hand is a parameter that can be tweaked based on how noisy images the model trains on.

In the forward process of our diffusion model, we introduce noise to the data through a structured approach that resembles the formation of a Markov chain. This process begins by generating Gaussian noise. In Section 2.2.1 we mentioned that there are various techniques to apply noise to the data - the β_t schedule. In our model, we simply apply a linear β_t schedule. This serves the dual purpose of noise weighting and data weighting. This linear framework allows us to incrementally blend the original data with the generated noise in a controlled manner.

To efficiently apply noise across multiple iterations, we utilize a technique inspired by the normal sum theorem (as detailed in Section 2.2.3). This theorem enables us to bypass the sequential simulation of each step in the Markov chain. Instead, we can directly compute all noisy versions of the data by appropriately scaling the Gaussian noise with the calculated weights. This scaling involves a linear combination of the noise and data weights, resulting in a spectrum of progressively noised data instances.

To efficiently apply noise across multiple iterations, we utilize a technique characteristic of DDIMs, inspired by the normal sum theorem as discussed in Section 2.2.3. This theorem allows us to directly compute all noisy versions of the data, eliminating the need for sequential simulation of each step in the Markov chain. We achieve this by appropriately scaling Gaussian noise with calculated weights – a linear combination of the noise and data weights – resulting in a spectrum of progressively noised data instances.

By using this method, we not only retain the original, unaltered image but also effectively generate and store τ distinct noisy versions of it. Each version represents a specific point along the trajectory of the forward diffusion process, offering a comprehensive view of the noise integration over time. This detailed temporal progression of all τ versions is crucial for subsequent stages of the model's application. Figure 4.4 and Figure 4.5 illustrates the τ noisy versions generated during this process. Here we use the MNIST and the rainbows dataset, and $\tau = 10$.

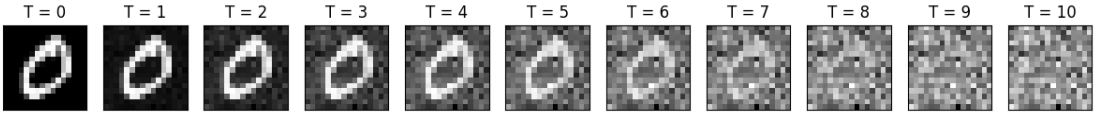


Figure 4.4: The forward process using the first element in the MNIST dataset.

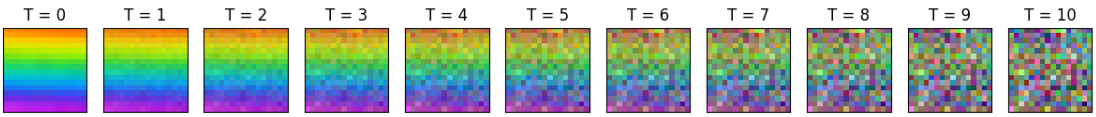


Figure 4.5: The forward process using the first element in the rainbow dataset.

4.4 The PQC

The PQC forms the core of our diffusion model, acting as the generative component within the reverse diffusion process. PennyLane's framework makes the circuits implementation straight-forward and highly readable. As outlined in Section 3.6, we

explore four distinct model variants. However, we will begin by examining the structure of the base model without any embedding techniques.

Let us preface our implementation with importing the PennyLane package:

```
import pennylane as qml
```

We now proceed to a step-by-step analysis of the key operations within the PQC class initializer. The initial step involves calculating how many qubits (named "wires" in PennyLane) the model will need:

```
def num_wires(self):
    return math.ceil(math.log2(self.w * self.h * self.c))
```

The calculation of the necessary qubits for our model begins with the product of the total number of pixels and the number of color channels in the image. For grayscale images, the number of channels 1; for RGB images, the number of channels increases to 3 to accommodate the red, green, and blue components that combine to produce the full spectrum of colors. Moreover, we feed this aggregated number into the binary logarithm. This operation reflects the exponential relationship between the number of qubits and their capacity for state representation. Since the total number of states that can be represented by qubits doubles with the addition of each qubit, the binary logarithm provides a direct measure of the qubits required for any given amount of data.

However, since the result of this logarithmic calculation may not always be a whole number, and the number of qubits must be a whole integer, we round up to the nearest whole number. This ensures that we have a sufficient number of qubits to represent the image data without any loss of information, effectively capturing the full detail of the image in either grayscale or RGB mode.

Subsequently, we select the quantum backend we want to use for computations. We briefly touched on PennyLane's interchangeable backends in Section 4.1, which provide built-in support for multiple quantum devices [60]. In this instance, PennyLane's default simulator with PyTorch integration is employed:

```
self.device = qml.device("default.qubit.torch", wires=self.wires)
```

The initialization of the weights for the entangling layers within the PQC follows. These weights are initialized to random values from a normal distribution, with a mean of 0 and a standard deviation of 1. This choice of initialization aligns the weight magnitudes with the dynamic range of the quantum-friendly hyperbolic tangent activation function we will be using later:

```
weights_shape = qml.StronglyEntanglingLayers.shape(
    n_layers=self.hyperparams.n_layers,
    n_wires=self.wires,
)
self.weights = torch.nn.Parameter(
    torch.randn(weights_shape, requires_grad=True)
)
```

The parameter `self.n_layers` determines the depth of the quantum circuit, which refers to the number of layers L of quantum gates the circuit is composed of. More

specifically, it is the maximum number of quantum gates applied sequentially on any path through the circuit.

The quantum circuit must be defined. The implementation of the base quantum circuit is outlined below. For the present walkthrough, you may disregard the parameters "labels" and "timesteps" as our initial emphasis centers on the model without any embedded information.

```
def circuit(self, inp, labels=None, timesteps=None):
    qml.AmplitudeEmbedding(
        features=inp,
        wires=range(self.dataset.num_wires()),
        normalize=True,
        pad_with=0.0,
    )
    qml.StronglyEntanglingLayers(
        weights=qw_map.tanh(self.weights),
        wires=range(self.wires),
    )
    return qml.probs(wires=range(self.wires))
```

The circuit receives an input vector `inp` representing the noise-corrupted image (inp_τ), which is amplitude embedded across all qubits. The `pad_width` parameter allows us to pad the input feature vector with a constant value to reach the required dimension of 2^n, where n is the number of qubits. This parameter is useful when the input vector has a dimension less than 2^n and needs to be padded to fit the quantum system. As detailed in Section 2.1.5, the `normalize=True` setting ensures the input data is normalized, transforming it into a valid quantum state.

Following this, the circuit uses the `StronglyEntanglingLayers` template, a feature in PennyLane designed to construct layers consisting of parameterized rotational gates and two-qubit entangling gates [4]. This component is fed the previously initialized weights, modulated by the tanh activation function through `qw_map.tanh()`. This activation function uses a technique called weight re-mapping, especially designed for quantum circuits, and is a method for accelerating convergence for quantum variational algorithms [26].

Lastly, the `qml.probs` function calculates the probabilities of obtaining specific measurement outcomes in the computational basis when measuring the specified wires.

After defining and initializing our quantum circuit, we pass it into a `QNode` for execution. The `QNode` (short for Quantum Node) in PennyLane is the fundamental building block that connects the quantum circuit to a quantum device and enables the powerful concept of differentiable quantum programming. The QNode serves as a construct that encapsulates the quantum circuit (`self.circuit`) and designates the quantum device (`self.device`) on which the circuit will be executed. It also establishes a bridge between quantum and classical machine learning. The `interface` parameter (`"torch"`) integrates PennyLane with libraries like PyTorch. This integration enables the optimization of quantum circuit parameters using well-established gradient-based techniques. The `diff_method` parameter (`"backprop"`) instructs PennyLane to use the efficient backpropagation algorithm for calculating gradients. This algorithm, widely used in training neural networks, allows seamless optimization of hybrid models that combine quantum circuits with classical machine learning elements.

```
self.qnode = qml.QNode(
    func=self.circuit,      # Quantum circuit function
    device=self.device,     # The device to run it on
    interface="torch",      # Integrate with PyTorch
    diff_method="backprop",# Use backpropagation for gradients
)
```

The `QNode` is now ready for use within the forward function. This function accepts all τ noisy image versions generated as shown in Section 4.3. It processes each image version through the quantum circuit and returns the results. The `*args` keyword argument represents the image data, while `**kwargs` allows for optional parameters (timestep or label data).

```
def forward(self, *args, **kwargs):
        x = self.qnode(*args, **kwargs)
        return x[:, : self.dataset.num_features()]
```

The final line of code, `return x[:, :NUM_PIXELS*NUM_CHANNELS]`, plays a crucial role in aligning the output with the expected format of the image data. Quantum circuits can generate a vast number of probability values (2^n, where n is the number of qubits). To ensure compatibility, this line slices the output array, selecting only the first `NUM_PIXELS*NUM_CHANNELS` elements along the second dimension. This ensures that the returned data is precisely aligned with the dimensions of the image being processed.

Figure 4.6 shows a visualization of the base models circuit. This one has 6 qubits, which is what the base model trained on MINIST 16x16 uses.

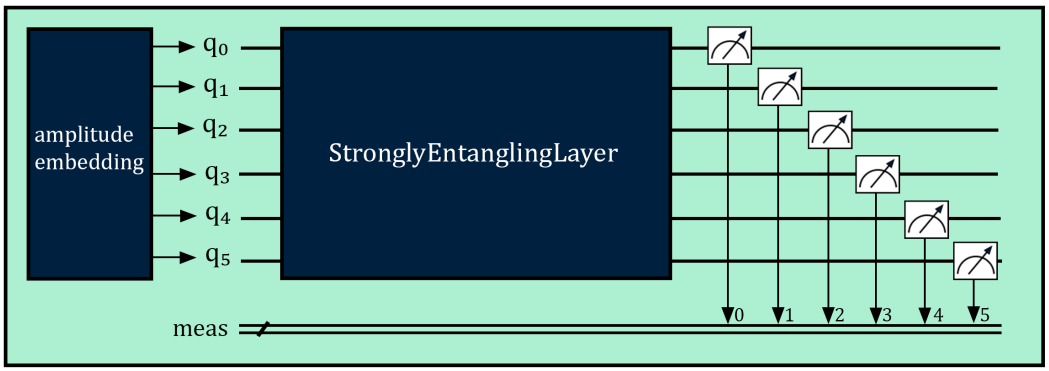


Figure 4.6: The circuit for the base model with 6 qubits.

4.5 Prediction with Quantum Values

A crucial aspect of generating accurate predictions involves the ability to model both positive and negative value fluctuations. This capability is essential because the model must introduce variability, or "noise," in both directions to function effectively.

If the noise output by the model is added in only one direction, it would mean that the model can only perform one type of adjustment to the pixels. In the context of generating black and white images that would be either darkening or lightening them, depending on how the model interprets and applies this positive noise. The key to a more versatile and effective noise application is the ability to introduce both positive and negative

adjustments, allowing the model to both increase and decrease the pixel intensity as needed. Without the capability to apply noise in both directions, the model's ability to add depth, contrast, and detail to the images would be significantly constrained. This principle extends to the output of our quantum circuit, which inherently produces probabilities that are non-negative. Given this constraint, our model cannot directly output negative values, which are necessary for a comprehensive noise prediction.

To bridge this gap, the data undergoes a critical post-processing step immediately after we obtain the probability predictions from the quantum circuit. The initial phase of this post-processing involves the following operation:

```
sums = predicted_noise.sum(dim=1)
predicted_noise -= (sums / self.dataset.num_features()).view(-1, 1)
```

Essentially, what this line does is subtract all predicted probabilities for each image by the mean probability of that image. By subtracting this value from each probability, it effectively shift the entire distribution of predicted noise values, centering the mean of the distribution around 0. The mean of 0 is significant because it establishes a balanced baseline from which noise can be added in both positive and negative directions. In essence, some noise values will become negative as a result of this subtraction, enabling the model to reduce pixel intensity where needed.

After adjusting the predicted noise to include both positive and negative values, the next challenge involves ensuring that the scale of the noise matches the scale of the actual noise distribution we aim to model. This is where the next line comes into play, and it is a crucial step in fine-tuning the model's output to more accurately reflect the characteristics of the target noise:

```
predicted_noise *= self.hyperparams.zeta
```

This operation functions to scale the amplitude of noise generated by our model. To achieve this, we define a custom constant: ζ (zeta) - a predefined constant used as a scaling factor. The selection of ζ is crucial, as it directly influences how closely the range of the model's generated noise approximates the range of the actual noise present in the target data. This alignment is important for ensuring the model's ability to accurately predict noise characteristics.

Why we need ζ:

Adjusting Scale The primary function of ζ is to calibrate the scale of the noise generated by the model. While a mean-centered distribution ensures that the noise predictions are balanced around zero, scale of these predictions might not initially match the characteristics of the actual noise distribution. ζ directly influences the scale of the noise values, allowing for expansion or contraction to achieve a closer alignment with the scale of the noise being modeled.

Enhancing Model Accuracy: By carefully calibrating ζ, we ensure that the noise generated by the model not only has the correct directionality (positive and negative adjustments) but also the appropriate magnitude. This alignment is critical for the model's ability to accurately simulate or predict noise, as it ensures that the model's noise effects are neither too subtle nor too exaggerated compared to the actual noise characteristics.

Customization and Flexibility: ζ provides a lever for fine-tuning model performance. Depending on the specific characteristics of the noise in the used dataset or the particular requirements of your application, ζ can be adjusted to optimize the fidelity of the noise prediction.

Different datasets would need a different ζ value for optimal results, and larger images would need a higher ζ value than small images as the probabilities predicted by the circuit will be inherently smaller. To address this, we define a dataset-dependent initial value for ζ and then multiply it by the number of features to determine the appropriate scaling.

```
self.zeta *= dataset.num_features()
```

4.6 Implementing Embeddings

When adding label and timestep embedding in the quantum diffusion model, a similar technique is utilized for both. An ancilla qubit is employed, and its state is modified using an RX-gate. The angle of rotation applied by the RX-gate is determined directly by a normalized value of the label or timestep being encoded. This approach allows the model to differentiate between distinct labels and timestep values.

4.6.1 Data Normalization

Encoding data directly as rotation angles in qubits can lead to issues due to the periodicity of rotations. A rotation of 2π radians in the quantum state space corresponds to a full rotation, bringing a qubit back to its original state. For instance, using raw data values like the labels 0 and 6 for rotations could be problematic. Given that 6 is very close to 2π radians (approximately 6.28), rotations corresponding to these values would be nearly indistinguishable, potentially leading to confusion in differentiating between the intended states. Figure 4.7 shows how rotating using zero and six as raw values makes the rotation almost indistinguishable.

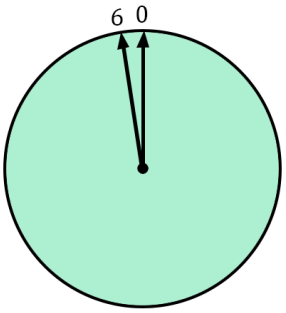
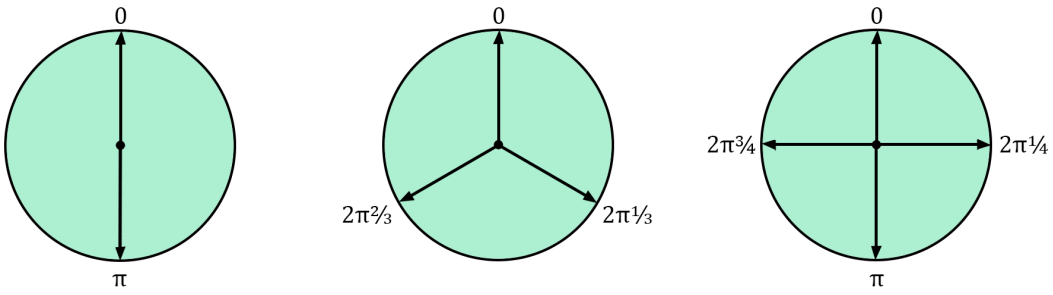


Figure 4.7: Rotations without normalization.

To circumvent this, we have developed a normalization technique that scales the data values to fit within the 0 to 2π range, ensuring that each value corresponds to a unique rotation angle, thereby enhancing state differentiation. Figure 4.8 shows how the rotations would look with two, three, and four different values.



(a) Two values. (b) Three values. (c) Four values.

Figure 4.8: Rotations with normalization.

This technique does not completely solve the aforementioned problem of indistinguishable embeddings when the number of embeddings is high. Rather, it reduces the similarly of the rotation angles, making sure to space them out as evenly as possible. This limitation is something we will discuss in depth in the discussion of this thesis (specifically; Sections 6.2.3, 6.2.5 and 6.5.2).

4.6.2 Implementing Label and Timestep Embedding

For flexible embedding within the model, the class `Embeddings` has been created. To specify the desired embedding, select one of the following options (using timestep embedding as an example):

```
embeds = Embeddings(
    #embeddings=[],
    embeddings=["timestep"],
    #embeddings=["label"],
    #embeddings=["timestep", "label"],
)
```

When calculating number of wires, the ancilla qubit(s) is being taken into account, which is shown under.

```
def num_ancilla(self):
    return len(self.embeddings)

self.wires = dataset.num_wires() + embeds.num_ancilla()
```

This ensures that the number of qubits is sufficient to represent all pixels in the input data (via amplitude encoding) and includes one additional qubit per embedding.

Two `if` statements are added to the quantum circuit `self.circuit`, after the input data is amplitude embedded.

```
if self.embeds.label_embedding() and labels is not None:
    qml.RX(phi=labels, wires=self.wires + self.embeds.label_wire())

if self.embeds.timestep_embedding() and timesteps is not None:
    qml.RX(phi=timesteps, wires=self.wires + self.embeds.timestep_wire())
```

Based on if label embedding and/or timestep embedding is present, the ancilla qubits are rotated in a specific angle using an Rx-rotation. The rotation angle is determined by the parameters `labels` and `timesteps` which is sent into the circuit. This is the normalized rotation technique explained in Section 4.6.1, which means it is a value between 0 and 2π. To know which qubits should be rotated, `self.embeds` has two functions `self.embeds.label_wire()` and `self.embeds.label_wire()`, which returns what qubit should be rotated based on how many embeddings are present.

Finally, the `StronglyEntanglingLayers` are applied across all qubits, including the ancilla qubits. This step ensures the information encoded in the ancilla qubit is entangled with the rest of the quantum system. This allows the encoded timestep and label information to influence the overall quantum state, and consequently, the model output.

Integrating an ancillary qubit into a quantum circuit inherently alters the output probability distribution. Specifically, the inclusion of one or two extra qubit doubles or quadruples the number of possible outcomes, thereby expanding the output space from n to $2n$ or $4n$ probabilities. This expansion reflects the binary nature of quantum measurements, where each qubit adds a factor of two to the number of possible states due to its two possible outcomes (0 or 1).

To reconcile this with our model's expected input dimensionality, we focus on the initial n probabilities, effectively compressing the expanded set back into an n-dimensional vector. The process of discarding the latter half or quarter of the probabilities is to maintain consistency with the model structure, and ensure the final output precisely matches the image dimensionality, defined by its size and color channels. The same forward function as introduced in Section 4.4 is used here, with the addition of timestep and label embeddings as optional keyword arguments.

An important consideration in this process is that the sum of the selected n probabilities will be less than 1, given that the total probability space (including the ignored values) originally summed to 1. This discrepancy necessitates an adjustment in the scaling factor ζ. When using one ancilla qubit the remaining probabilities will, on average, summarize 0.5, so ζ is multiplied by 2. For two ancilla qubits, the probabilities will summarize to around 0.25, so ζ is multiplied by 4. The adjustment compensates for the reduced total probability by scaling up the selected probabilities, thereby preserving the relative distribution.

Figure 4.9 shows a visualization of the hybrid models circuit. This particular one has 8 qubits, where two of them are used for label and timestep embedding.

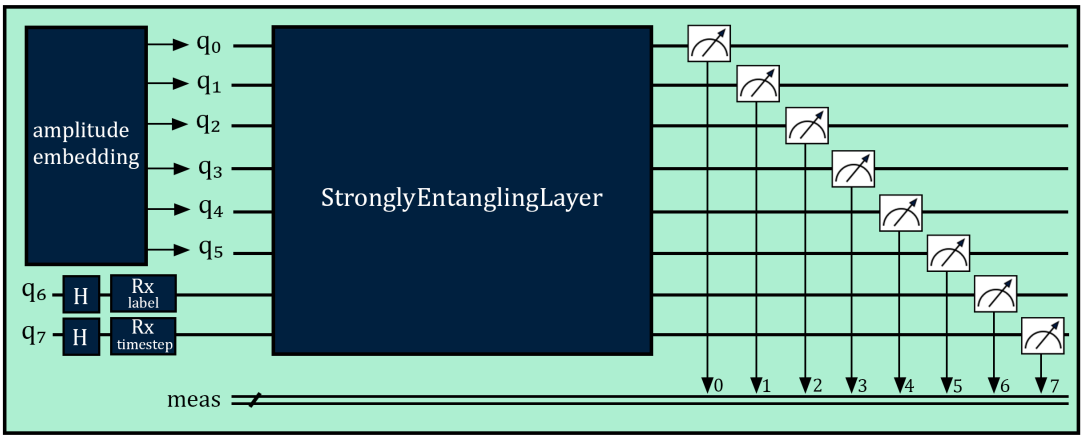


Figure 4.9: The hybrid circuit with 8 qubits.

4.7 Training

Every machine learning model requires a training phase. In this section, we examine the training process of the QDM, where it transforms the theoretical underpinnings of the model into practical functionality by optimizing its parameters.

Initially, the training data is subjected to noise for τ steps, creating t noisy versions as detailed in Section 2.2.1. Each of these images are then amplitude-encoded and fed into the quantum circuit. The circuit is designed to process the data in a step-wise manner, starting by attempting to reverse the noising process from $t = 1$ back to $t = 0$, then progressing from $t = 2$ to $t = 1$, and continuing in this fashion until the final step from $t = T$ to $t = \tau - 1$. In models using ancilla qubits, the normalized timestep t and/or the normalized label of the data is also sent into the circuit. Next up, all the predictions are post processed as explained in Section 4.5, transforming the data from an array of probabilities to an actual noise prediction. Following this, the model evaluates all τ denoised outputs against the corresponding levels of actual noise they aim to replicate using a loss function. This comparison forms the basis for computing an aggregated loss from all the τ versions, which guides the optimization and learning of the model.

The model is trained with the Mean Squared Error (MSE) loss function. The MSE function calculates the average of the squares of the differences between the predicted values ($\hat{y}_i$) and the actual values (y_i):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \tag{4.1}$$

This function is widely used in regression analysis to measure the quality of an estimator by quantifying the variance between predicted and observed outcomes. It provides a single value that represents the average of these squared differences, serving as a criterion to evaluate the performance of the predictive model. So in context of our model, the loss is calculated between the predicted image at a certain denoising step, and the actual image at that step. This loss function is effective here as it directly penalizes the model for deviations from the target image, encouraging it to produce outputs that are as close as possible to the real noise that was added to the particular time step.

Lastly, the parameters of the PQC are adjusted. This optimization is carried out by the ADAM algorithm, which updates the weights based on gradients derived from

the loss function through back-propagation. ADAM is well recognized for its efficiency in deep learning tasks. It combines momentum and adaptive learning rate techniques, maintaining two moving averages for each parameter: one for the gradients, aiding in momentum, and another for squared gradients, which adjusts the learning rates. This approach helps ADAM navigate the challenges of sparse or noisy gradients, making it a reliable choice for complex models where robust and effective optimization is important.

Lastly, the parameters of the PQC are adjusted using the ADAM algorithm. ADAM is a well-known optimization tool in deep learning, efficiently updating the weights by analyzing the gradients of the loss function through back-propagation. It combines momentum—which smooths out the updates—with adaptive learning rates, allowing it to handle situations with sparse or noisy gradients. These characteristics make ADAM a good fit for complex models where robust and effective optimization is important.

Figure 4.10 shows a visualization of the training process.

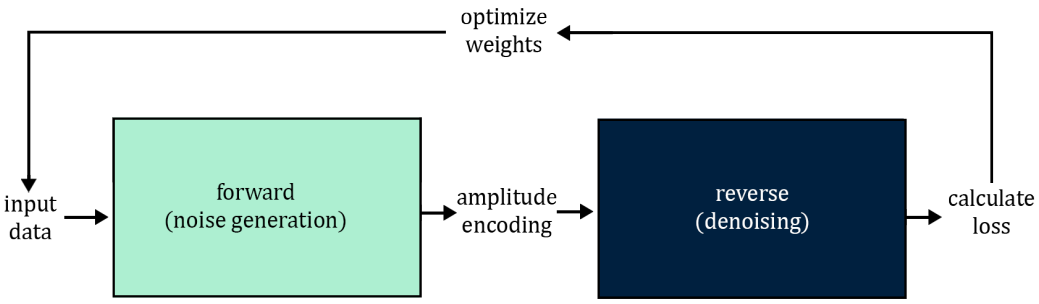


Figure 4.10: The training loop of the QDM.

A code snippet of the training loop can be found in Appendix B.

4.8 Sampling

The sampling stage is a core process within the diffusion model, responsible for generating new samples that reflect the distribution learned during training. It involves iteratively reversing the noise-addition process, gradually transforming noise into structured data.

The algorithm begins by creating a random noisy state, t, which is amplitude encoded and fed into the PQC. The circuit's output, processed as described in Section 4.5, estimates the noise needed to reduce to a less noisy state, $t-1$. This estimated noise is then subtracted from the input data. This process is repeated in a loop, decrementing the noisy state index t by one in each iteration, for a total of τ times. In each cycle, the increasingly refined image is re-introduced into the PQC, allowing the algorithm to gradually denoise the image until the desired state is achieved.

Upon completion of the τ iterations, when $t = 0$, the process yields a new synthetic image. This image, now devoid of the initial noise, represents a quantum-generated approximation of the target data distribution.

To amplify the denosising phase, we are using a higher ζ value when sampling. Here, `predicted_noise` is multiplied by `sample_zeta`. This value will be a multiple of the original ζ value, for example `zeta * 5`.

In scenarios where the model incorporates timestep or label embeddings, the PQC requires supplementary information during the sampling process. Specifically, for timestep embeddings, the current timestep t is normalized as explained in Section 4.6.1

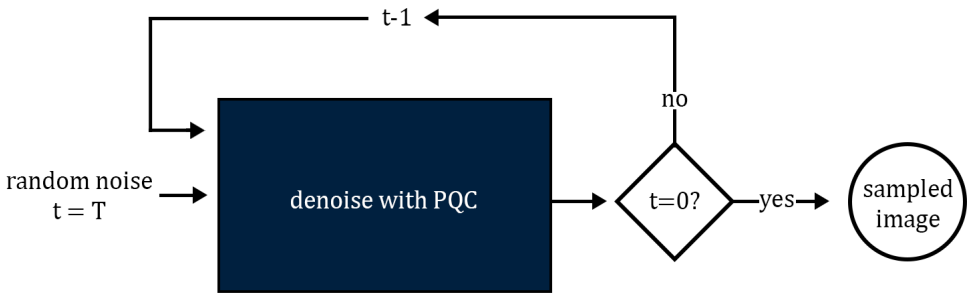


Figure 4.11: The sampling loop.

and incorporated alongside the primary data when inputting into the PQC. For label embeddings, we use a specialized function, `generate_labels`, which is devised to construct a normalized array (explained in Section 4.6.1) reflective of the training data labels. This array is then integrated into the PQC. The function operates by producing a balanced representation of each label within the array. For instance, if the task is to sample 9 images and the training was conducted on a dataset with labels `[1,2,3]`, `generate_labels` will output an array containing three instances of each label.

A code snippet of the sampling function can be found in Appendix A.

Chapter 5

Evaluation

In this chapter, we will be presenting the results produced by our different proposed model variations. We will be showcasing various performance metric graphs alongside the generated images our models produce. The primary focus lies in exploring the impact of various hyperparameters. All of the results presented in this chapter are the direct probability distributions that the quantum diffusion model outputs. We have not post-processed any of these results with the exception of the shift in value that we explained in Section 4.5.

All of the evaluation work was done on a system equipped with an Intel Core i7-10700K CPU, 32 GB of DDR4 RAM, and an MSI GeForce RTX 3070 Ti GPU. Notably, all the model variations ran slower while utilizing CUDA, so the model is solely run in the CPU. For all the evaluation work using the MNIST dataset, we have chosen a dataset size of 1024 and a batch size of 32. Moreover, we have kept the learning rate (η) static at $\eta = 0.001$.

5.1 FID Score

The Mean Square Error (MSE) metric, commonly used for assessing model performance, lacks the ability to capture the perceptual quality of images. MSE compares pixels but fails to consider structural and textural intricacies crucial for human perception. Consequently, low MSE values do not guarantee perceptual fidelity, leading to a disparity between numerical accuracy and visual quality.

Moreover, MSE is sensitive to the scale of compared values. Smaller values, even if inaccurately predicted, result in lower squared errors, skewing the loss assessment. For instance, comparing models trained with different scale values proves ineffective due to this sensitivity. This sensitivity is exemplified by the number of timesteps (τ) hyperparameter, which decreases in value with each increment as explained in Section 2.2.4 and visually depicted in Figure 2.9. Comparing models trained with different τ values would be illogical, given the inherently lower MSE loss for the model with the higher τ. This disparity in loss values may predominantly stem from the reduced scale of noise increments rather than indicating better model performance.

Predicted Noise	Actual Noise
-5.4377e-03	-0.4863
-5.7717e-03	-0.6287
1.3503e-02	-0.5439
-6.8551e-03	-0.3570
-2.6402e-03	-0.3730

Table 5.1: Predicted Noise and Actual Noise for $\tau = 1$

Predicted Noise	Actual Noise
-3.7581e-03	-2.6870e-02
-3.3970e-03	-1.5143e-02
-7.3282e-03	-1.0354e-02
-2.3657e-03	-1.8683e-02
-1.1101e-03	-1.7951e-02

Table 5.2: Predicted Noise and Actual Noise for $\tau = 30$

Tables 5.1 and 5.2 present values during the training of models with $\tau = 1$ and $\tau = 30$, respectively. These tables compare predicted versus actual noise for five images in the batch. Predicted noise denotes the model's estimated added noise for a specific timestep, whereas actual noise represents the noise actually added to that timestep. The actual noise added to the model with $\tau = 30$ are inherently lower, thus scores a much better score than $\tau = 1$ despite the samples from these models being equally as bad.

Thus, we need another way to objectively measure the efficacy of these models. This is where we introduced the Fréchet Inception Distance (FID), which is a more holistic approach to evaluating generated images. Unlike MSE, FID assesses the similarity between the distribution of generated images and that of real images, using deep features extracted by our self-trained feature extraction model. This method captures not only the pixel-level accuracy but also the deeper, more abstract features that contribute to the overall visual quality of images.

The FID score is computed as:

$$\text{FID} = ||\mu_r - \mu_g||^2 + T_r(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}}) \tag{5.1}$$

where:

- μ_r and μ_g are the feature-wise mean vectors of the real and generated images, respectively.
- Σ_r and Σ_g are the covariance matrices of the real and generated images, respectively.
- T_r denotes the trace of a matrix, the sum of its diagonal elements.

The FID score essentially gauges the distance between two multivariate Gaussian distributions—one fitted to feature vectors from real images and the other to feature vectors from generated images. These feature vectors are obtained by passing images through a pre-trained feature extraction model, capturing high-level aspects of the images' content and structure.

A lower FID score signifies closer quality and feature resemblance between generated and real images, indicative of enhanced visual fidelity and realism. However, subjective preferences may vary, occasionally diverging from FID score implications. Nonetheless, integrating FID into the evaluation framework creates a more holistic and objective assessment of model performance, augmenting the traditional MSE-based approach.

5.2 Base Model Experimentation

In this section, we will experiment with various hyperparameters for the base model to understand how they affect the generation process and identify optimal values that showcase the model's full potential.

We will begin by experimenting with the ζ value. This is crucial to establish before adjusting other hyperparameters, as ζ determines the intensity of noise added during the diffusion process. A suboptimal ζ value can significantly hinder the model's learning ability, potentially invalidating tests on other hyperparameters.

5.2.1 Zeta (ζ)

The ζ hyperparameter is a noise amplifier that we introduced to our model as we discussed in Section 4.5. To explore ζ and to find an optimal value, we tested eight models with values varying from 0.0025 to 0.02. These models were trained over three epochs, $L = 60$, and $\tau = 10$. The results are shown in Figure 5.1.

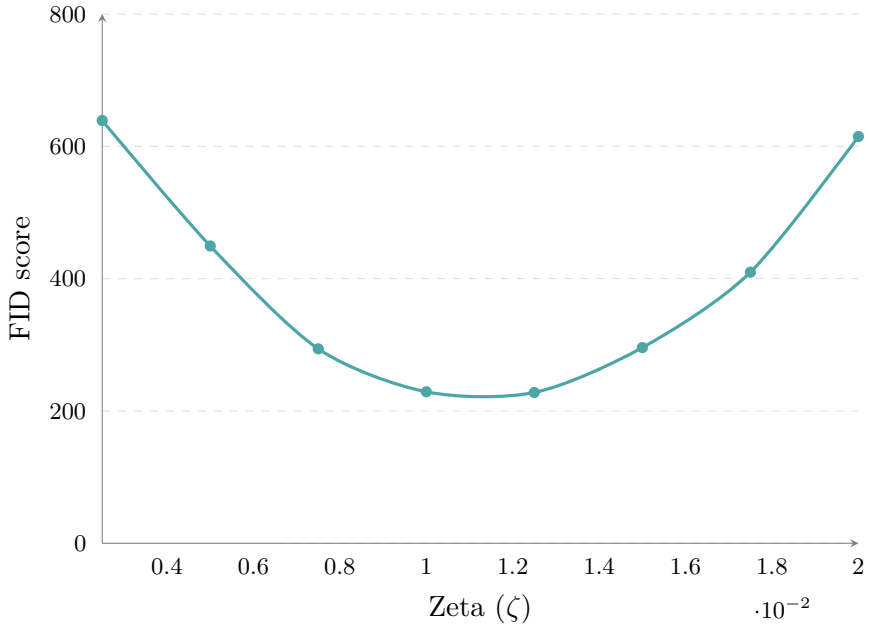
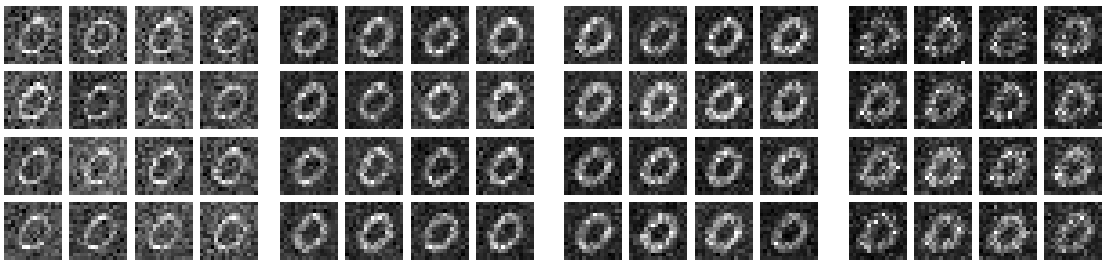


Figure 5.1: FID scores over ζ, averaged over 3 runs.

The data from Figure 5.1 tells us that as ζ increases from 0.005 to 0.01, there is a distinct decrease in FID scores. Specifically, the scores reduce sharply from 638.99 points at $\zeta = 0.0025$ to 228.97 points at $\zeta = 0.01$, indicating improved model performance. $\zeta = 0.0125$ performs similarly to $\zeta = 0.01$ in terms of FID score, however, beyond this the model performance starts to degrade. A subsequent rise in ζ beyond 0.0125 to 0.02 leads to an increase in FID scores up to 614.74. Based on the data from Figure 5.1, the ζ optima seems to be between 0.01 and 0.0125.

In Section 5.1 we mentioned that a lower FID score does not necessarily translate to more visually appealing images. Thus, we have decided to visualize four of these generated images.



(a) $\zeta = 0.0025$ FID=638.9 (b) $\zeta = 0.0075$ FID=293.9 (c) $\zeta = 0.01$ FID=228.9 (d) $\zeta = 0.02$ FID=614.7

Figure 5.2: Four different ζ values from Figure 5.1.

Generally, the images in Figure 5.2 align with the findings in Figure 5.1: models with $\zeta = 0.005$ and $\zeta = 0.02$ perform poorly compared to those with more moderate values. One observation is that models with lower ζ values tend to produce thinner zeroes, while higher values tend to result in thicker zeroes.

Based on the visual and quantitative results in Figure 5.2 and Figure 5.1, we have chosen $\zeta = 0.0075$ as the default value for our model. This selection, despite $\zeta = 0.01$ and $\zeta = 0.0125$ achieving slightly lower FID scores, is due to $\zeta = 0.0075$ producing thinner zeroes, increasing the visibility of subtle shape variations in the data. Although subjective, we believe that $\zeta = 0.0075$ offers a balanced approach, ensuring detailed and discernible data representations while maintaining competitive FID scores.

5.2.2 Sample Zeta (Z)

Sample zeta, or Z, is derived from multiplying ζ with a static number. In this section we will attempt to find a suitable number to amplify ζ with in order to minimize FID scores.

The data presented in Figure 5.3 demonstrates the impact of varying levels of Z on the FID score and training time in quantum diffusion models. We have trained twelve models that ranges from $Z = \zeta * 1$ up until $Z = \zeta * 12$.

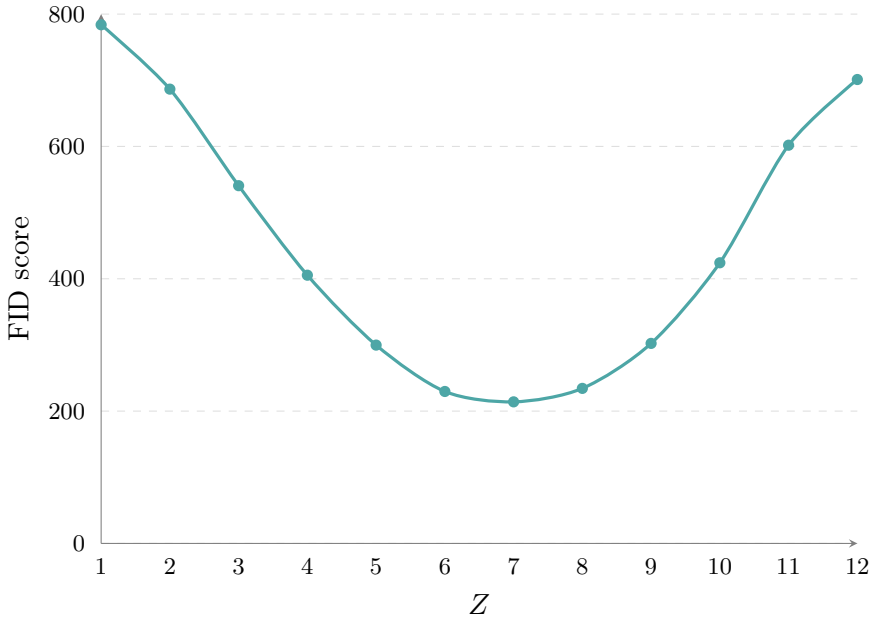
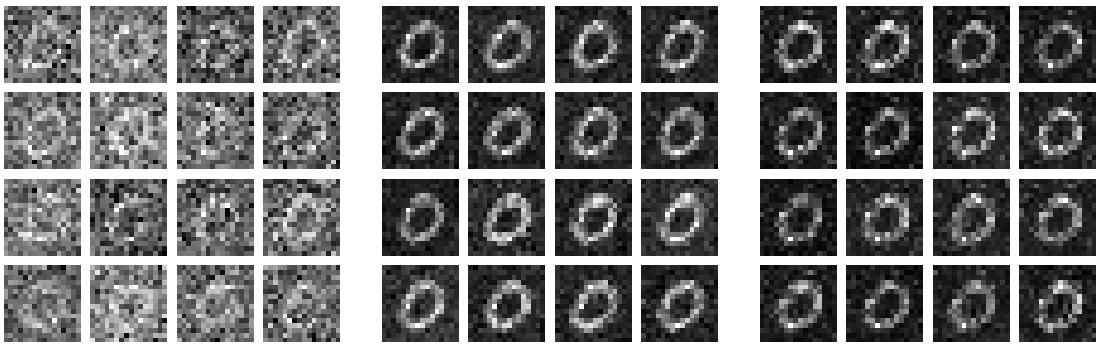


Figure 5.3: FID score and training time over Z, averaged over 3 runs.

The FID score exhibits a notable decline as Z increases from $\zeta * 1$ to $\zeta * 6$, decreasing from 783.84 points to 229.66 points. This trend suggests that a higher Z enhances the model's ability to generate more accurate representations. However, beyond $Z = \zeta * 6$, the FID score decrease begins to slow down, hitting the lowest at $Z = \zeta * 7$. Beyond this point, the FID score begins to rise sharply, and we see a reverse in the trend we saw previously. The FID score ultimately peaks at 701.13 at $Z = \zeta * 12$, which implies a degradation in model quality with excessively high Z values.

It is important to keep in mind that Z can be viewed as the intensity of the post-processing. The higher the Z, the more contrast the images will have. We can confirm this by looking at three of the generated images in Figure 5.3. These images will be shown in Figure 5.4.



(a) $Z = \zeta * 1$
FID=783.8

(b) $Z = \zeta * 7$
FID=213.9

(c) $Z = \zeta * 12$
FID=701.1

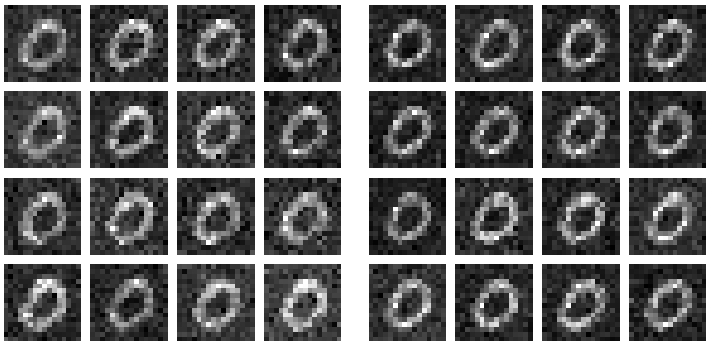
Figure 5.4: Three different Z values from Figure 5.1.

Figures 5.4a and 5.4c show a clear distinction in appearance. While Figure 5.2a appears very noisy, with only a faint outline of the digit visible, Figure 5.2b achieved the lowest FID score and appears to be a good sample. Figure 5.2c, similar to Figure

5.2b, displays a clear contrast between the dark background and white zeroes, but the zeroes appear darker and have a significantly higher FID score.

The intensity of the post-processing is very dataset dependant. For some datasets, like the MNIST dataset, a higher contrast is desirable. A high contrast would ensure a clear distinction between the white handwritten digit and the otherwise black background. For other datasets, however, high contrast is not necessarily as desirable. When you are generating a rainbow, for example, one would perhaps prefer the colors to blend together more naturally rather than have a clear-cut line between them. We will be experimenting more with this in Section 5.6.

For further experimentation, we have decided to proceed with $Z = \zeta * 5$ as our default value. This is partly because we do not want to over-adjust our data processing to fit the MNIST dataset, and would prefer to keep the Z value more moderate. Moreover, there is minimal distinction between $Z = \zeta * 5$ and $Z = \zeta * 7$ both visually and in FID score. These images can be seen side-by-side in Figure 5.5.



(a) $Z = \zeta * 5$
FID=299.7

(b) $Z = \zeta * 7$
FID=213.9

Figure 5.5: Three different L values from Figure 5.11.

Figures 5.6, 5.7 and 5.8 shows in detail how Z impacts every T step in the sampling process. More specifically, how Z heavily alters the τ steps of a denoising phase. The figures demonstrate that an increased ζ value significantly enhances the denoising process, thereby accelerating image sampling. This, in turn, improves both contrast and clarity. For instance, at step $\tau = 2$ in Figure 5.6, the image only exhibits noise. For this Z value, it is not until $\tau = 8$ or $\tau = 9$ where any notable structure. However, in Figure 5.8, when $Z = 5 * \zeta$, a circle becomes distinctly visible.

To further visualize how Z affects the images in the sampling process, refer to Figures 5.6, 5.7 and 5.8. These three images shows the entire denoising process from $\tau = 10$ to $\tau = 0$.

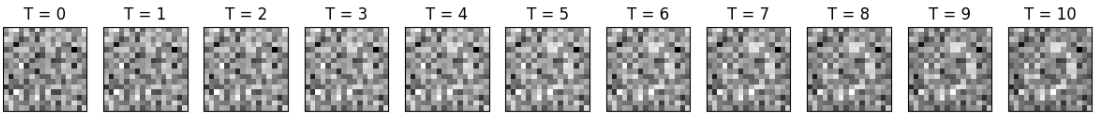


Figure 5.6: $Z = \zeta * 1$

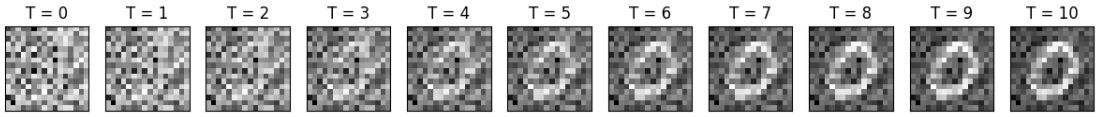


Figure 5.7: $Z = \zeta * 3$

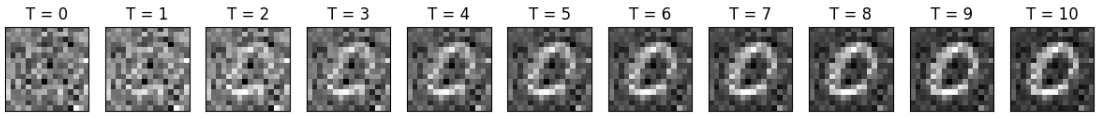


Figure 5.8: $Z = \zeta * 5$

5.2.3 Circuit depth (L)

In this section, we evaluate the impact of L on the performance of the QDM. The L hyperparameter determines the depth of the StronglyEntanglingLayers component in Pennylane, specifying the number of layers it contains, as explained in Section 4.4. The concept of circuit depth in quantum computing is analogous to the number of parameters in traditional neural network; deeper circuit can represent more complex data. We focus on how variations in L affect the FID scores, with the goal of providing insights into the optimal balance between circuit complexity and practical constraints.

The first thing we did was look at the relationship between E and L. We were curious as to whether it was beneficial to have a low E and a high L, a high E and a low L, or somewhere in the middle. To do this, we arbitrarily selected a value, 150, as the number of total layers we wanted the model to train on. We will call this value EL, and it is quite literally calculated by $EL = E * L$.

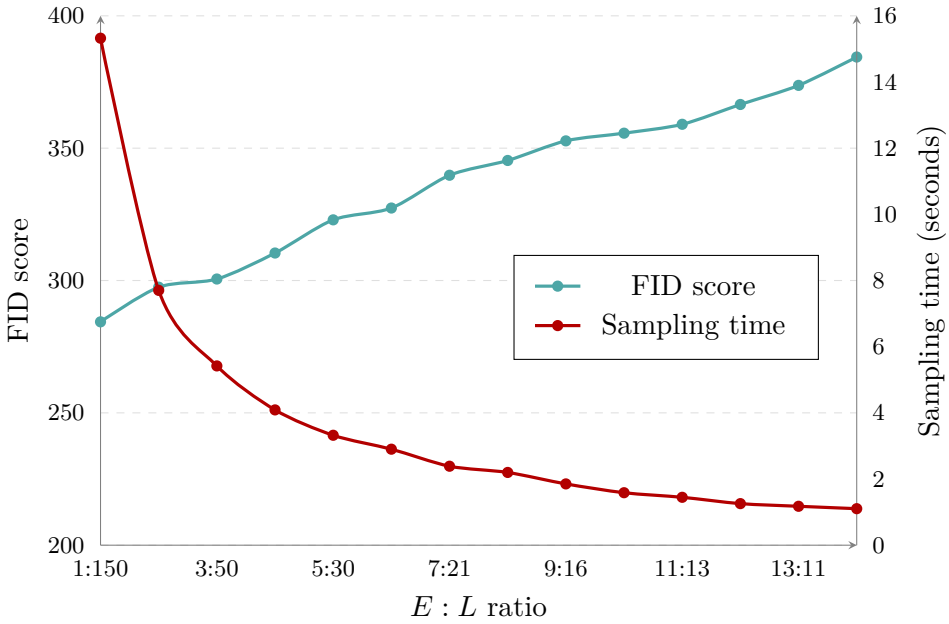


Figure 5.9: FID score and sampling time as a function of the $E : L$ ratio.

Figure 5.9 provides some interesting results. As the number of epochs increases, and subsequently the depth of the quantum circuit decreases, the FID score worsens. The

model with $EL = 1 : 150$ scores 284.386 points, while the model with $EL = 13 : 11$ scores 384.4 points.

We chose not to show training times for these models, as these appear unaffected by the various EL ratios.

Notably, there is a sharp decrease in sampling times. This can be attributed to the reduced complexity of the quantum circuit, as fewer layers lead to faster computation. Going forward, we will adopt 3 epochs as the standard for the number of epochs in our evaluations, as we find it a suitable middle-ground between adequate FID scores and low sampling times.

To further explore L, we ran ten models with varying L values. The data in Figure 5.10 presents a dual analysis of FID scores and training times across varying depths L of a quantum circuit, averaged over 3 runs.

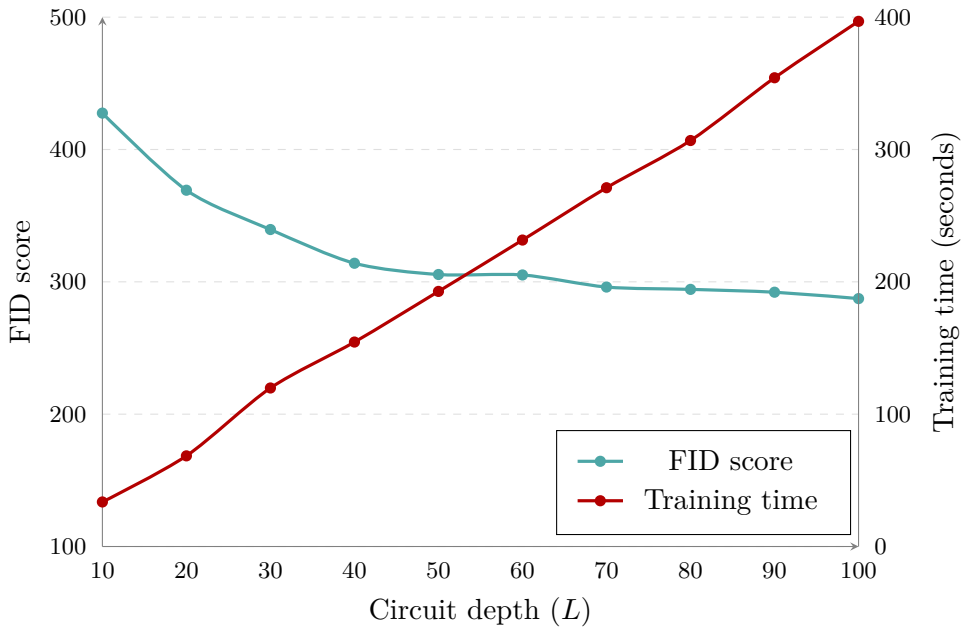
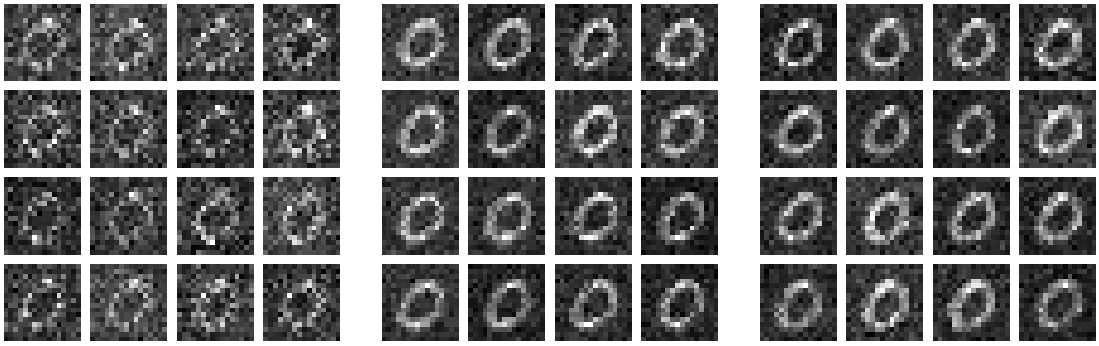


Figure 5.10: FID score and training time over L, averaged over 3 runs.

The FID score in Figure 5.10 shows a general decreasing trend as L increases from 10 to 100, suggesting that the model performance improves as the circuit depth increases. There is a notable drop in FID score between $L = 10$ and $L = 40$, followed by a more subtle decrease from $L = 50$ and onwards. This suggests increasing L has diminishing returns.

Conversely, the training time exhibits a consistent linear increase with L. This increase is expected as deeper circuits require more computation, reflecting directly in longer training durations. The gradual slope from $L = 10$ to $L = 100$ underlines the added computational overhead as the circuit depth is increased.

In Figure 5.11 we showcase the images generated from Figure 5.10. The results show a clear distinction between the lower depth model that produced Figure 5.11a and the model with the higher depth that produced Figure 5.11c. As L increases, so does the FID scores and the overall efficacy of the generated images.



(a) $L = 10$, FID=427.4 (b) $L = 50$, FID=305.5 (c) $L = 100$, FID=287.3

Figure 5.11: Three different L values from Figure 5.10.

Subjectively, we struggle to discern any notable difference between Figures 5.11b and 5.11c in respect to image quality. In terms of FID score, only 18.2 points separate the two models, despite the latter taking twice as long to train. For further experimentation with the model, we have thus chosen to continue using $L = 50$ as our default parameter.

5.2.4 Number of timesteps (τ)

In Section 2.2.4 we discussed that τ directly influences the quality of the generated images. The aforementioned section concludes that the choice of τ is problem-specific. In this section, we will attempt to find a suitable τ value for our model.

Figure 5.12 serves as a foundation for understanding how τ modulates the performance across different L. The graph reveals a bell-shaped relationship between τ and FID scores.

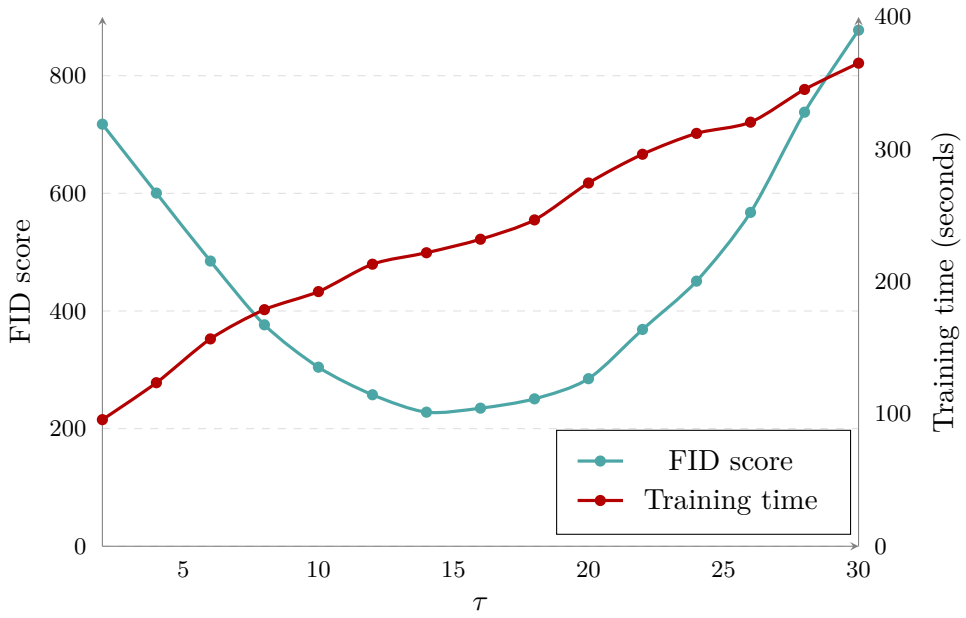


Figure 5.12: Three models over τ, averaged over 3 runs.

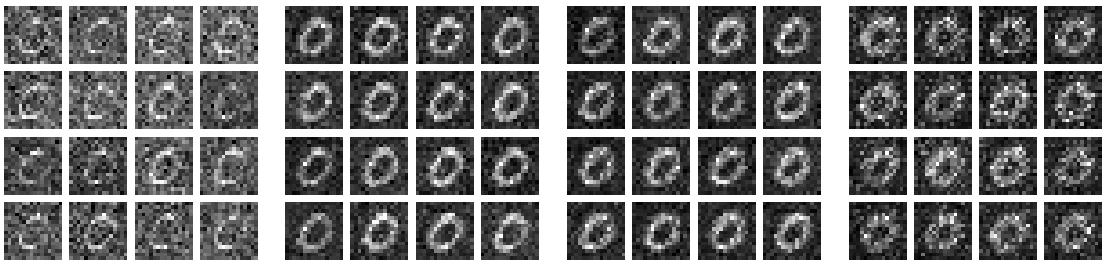
The initial decrease in FID scores as τ increases from 2 to 14 suggests that the model benefits from additional timesteps to refine its internal representations and stabilize the

generation process. This optimal performance at $\tau = 14$ demonstrates a sweet spot where the model has sufficiently learned to reduce noise and enhance detail without yet succumbing to overfitting or instability.

As τ further increases beyond 14, the subsequent rise in FID scores signals deteriorating model performance. This increase could be indicative of overparameterization, where the model begins to "memorize" rather than "generalize", thus decreasing its ability to produce varied and high-quality outputs. This phase of the curve is crucial as it highlights the diminishing returns of additional timesteps.

It is also important to consider the training time required at different timesteps. The training time graph in Figure 5.12 provides a more holistic view of the trade-offs involved in selecting the optimal τ for quantum diffusion models. The data demonstrates an almost linear increase in training time with the number of timesteps across all quantum depths. This linear relationship underscores the incremental cost of computational resources as τ increases. While higher timesteps can lead to improved model performance up to a point, as previously discussed in the context of FID scores, this must be balanced against the additional time and computational overhead involved.

In Figure 5.13, we have plotted four different T values: the lowest ($\tau = 2$), the highest ($\tau = 30$), the value yielding the lowest FID score ($\tau = 14$), and $\tau = 10$. We specifically include $\tau = 10$ as this will be the value used for subsequent experiments. Experimental results indicate that this hyperparameter value offers a favorable trade-off between generating images of acceptable quality with a low FID score, while maintaining manageable training and sampling times.



(a) $\tau = 2$, FID=717.5 (b) $\tau = 10$, FID=304.4 (c) $\tau = 14$, FID=220.9 (d) $\tau = 30$, FID=877.4

Figure 5.13: Four different τ values from Figure 5.12.

5.2.5 Standard deviation (σ)

In this section we will show the impact that σ has on the generated images and attempt to find a suitable value. The σ value refers to the standard deviation in the Gaussian noise distribution that we are applying to the images in the forward phase.

In Figure 5.14 we will plot the FID value of models trained on 6 different σ values.

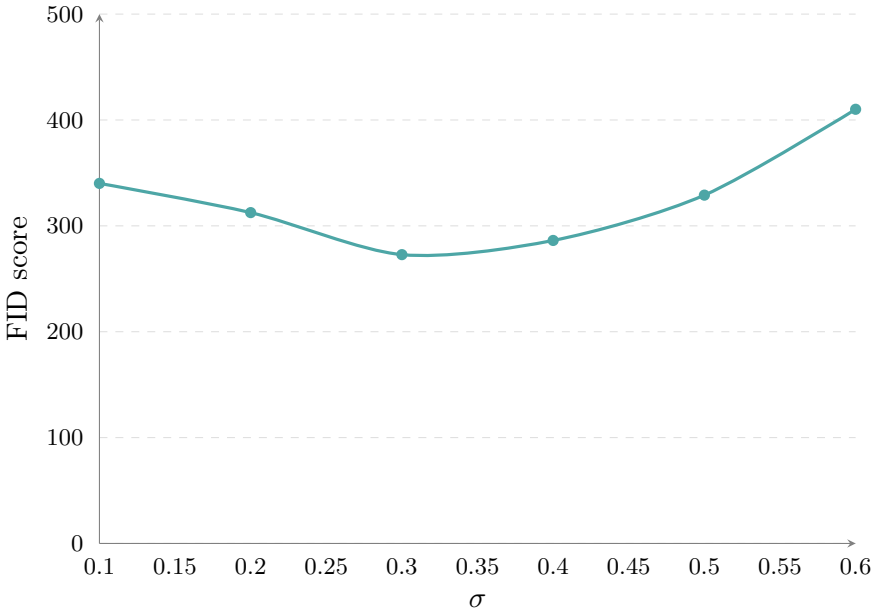
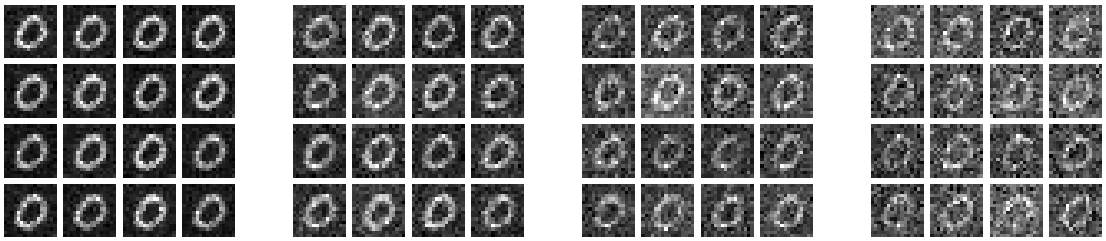


Figure 5.14: FID scores over σ, averaged over 3 runs.

Initially, the FID scores decrease moderately as σ increases from 0.1 to 0.3, reaching a low of 272.8 points. As σ continues to rise, FID scores increase significantly, particularly after $\sigma = 0.4$, peaking at 410.1 points at $\sigma = 0.6$.

Figure 5.15 displays the results of four models, each trained with a different σ value based on the evaluations shown in Figure 5.14.



(a) $\sigma = 0.1$,FID=340.1 (b) $\sigma = 0.2$, FID=312.5 (c) $\sigma = 0.3$, FID=272.8 (d) $\sigma = 0.4$, FID=286.2

Figure 5.15: Four different σ values from Figure 5.14.

The images indicates that a higher σ value produces more varied samples, although at the cost of image quality. The images with $\sigma = 0.1$ have a very pronounced and clear imagery with almost no surrounding noise. However, the results are all very homogeneous. Conversely, the images with $\sigma = 0.4$ exhibit a lot of noise but the samples themselves are very varied in shape. One of the FID score's key strengths is its ability to consider the diversity of generated output in relation to the training dataset. This is why the image with $\sigma = 0.1$, which looks clear and smooth, gives a worse FID score than $\sigma = 0.4$, which is more noisy. An ideal σ value should strike a balance between these extremes. Therefore, we will continue using $\sigma = 0.2$, as it appears to offer a favorable compromise between visual clarity and sample diversity.

5.3 Temporal Model

In this section, the temporal model will be evaluated and its performance will be compared to the base model. Figure 5.16 illustrates how FID scores vary as a function of L for both models. The base model scores are those presented in Figure 5.10.

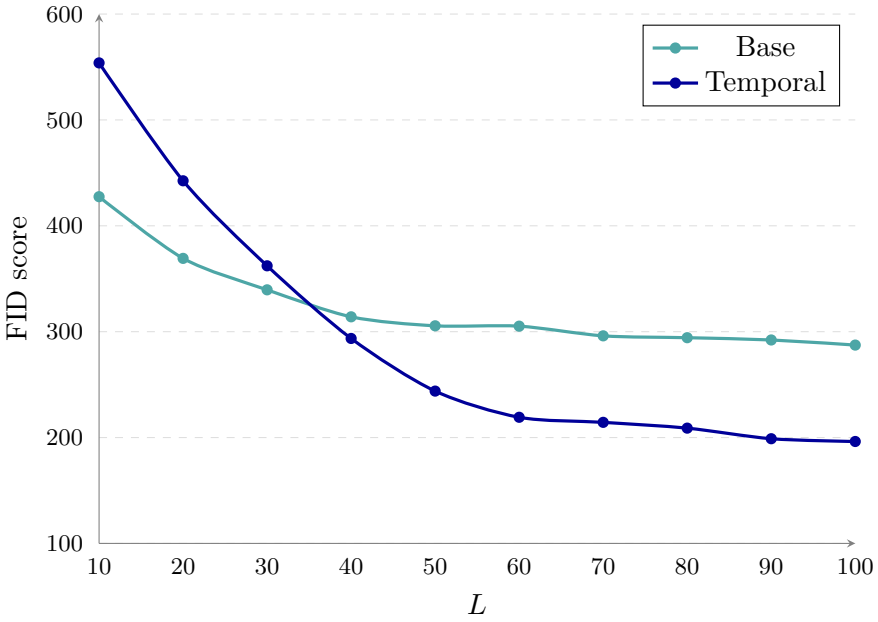


Figure 5.16: FID scores over L for the base model and the temporal model.

While both models exhibit improved efficacy with increasing L, the temporal model demonstrates a significantly sharper performance increase compared to the base model.

The temporal model initially performs worse than the basic model, showing a higher FID score of 553.8 at $L = 10$ compared to 427.4 for the base model. This trend persists up to $L = 40$, after which the temporal model outperforms the base model. The temporal model continues to enhance its performance from $L = 40$ to $L = 100$, whereas the base model appears to have reached a plateau from $L = 50$ onwards. This suggests that even at $L = 100$, the temporal model exhibits ongoing improvement, contrasting with the basic model's stagnant trend.

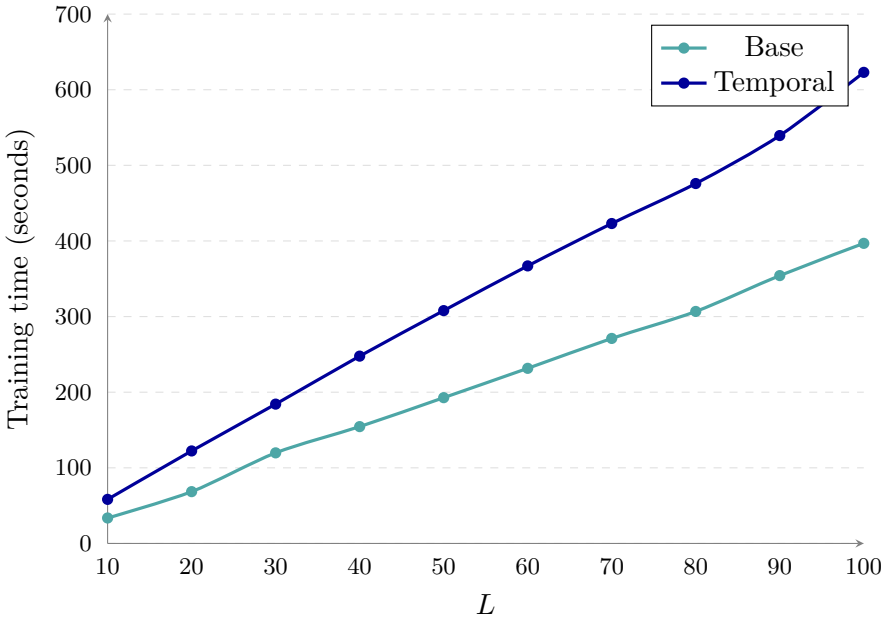
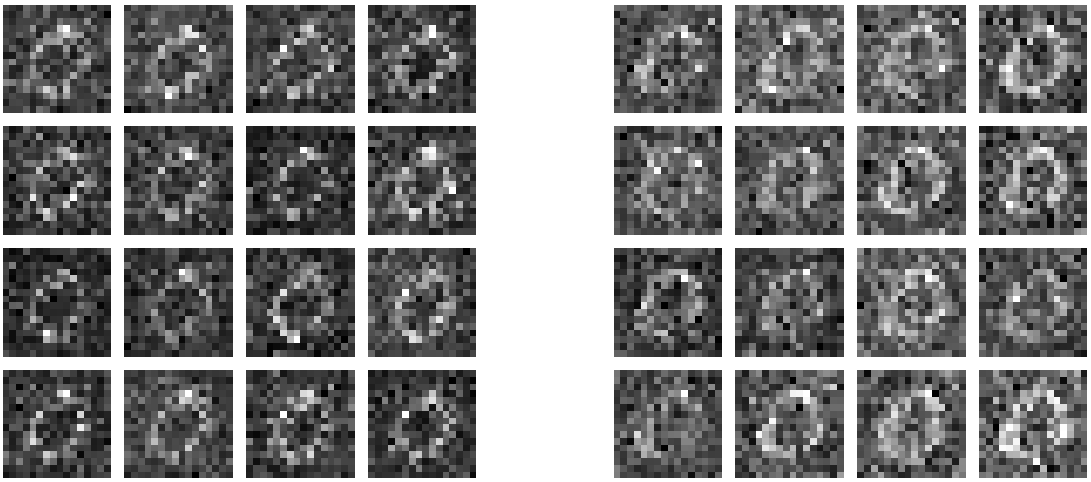


Figure 5.17: Training time over L for the base model and temporal models.

Figure 5.17 presents the training times for both the base and temporal models as L increases. As L increases, both models exhibit longer training times. The base model shows a gradual increase in training time, starting at 33.6 seconds at $L = 10$ and rising steadily to 396.8 seconds at $L = 100$. The temporal model starts at a higher initial time of 58.2 seconds at $L = 10$ and escalates to 622.9 seconds at $L = 100$. The temporal model incurs a 60% increase in average training time compared to the base model, due to the added complexity of the ancilla qubit.

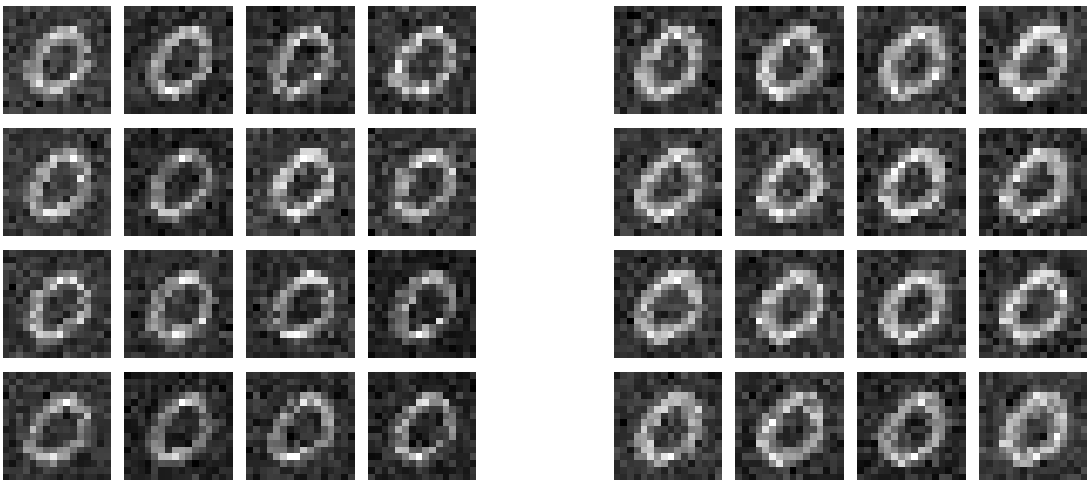
Next up we will be visually comparing the images generated with and without timestep embedding. Figures 5.18, 5.19 and 5.20 shows the two models trained on $L = 10$, $L = 50$, $L = 100$ respectively.



(a) Base model, FID=427.4

(b) Temporal model, FID=553.8

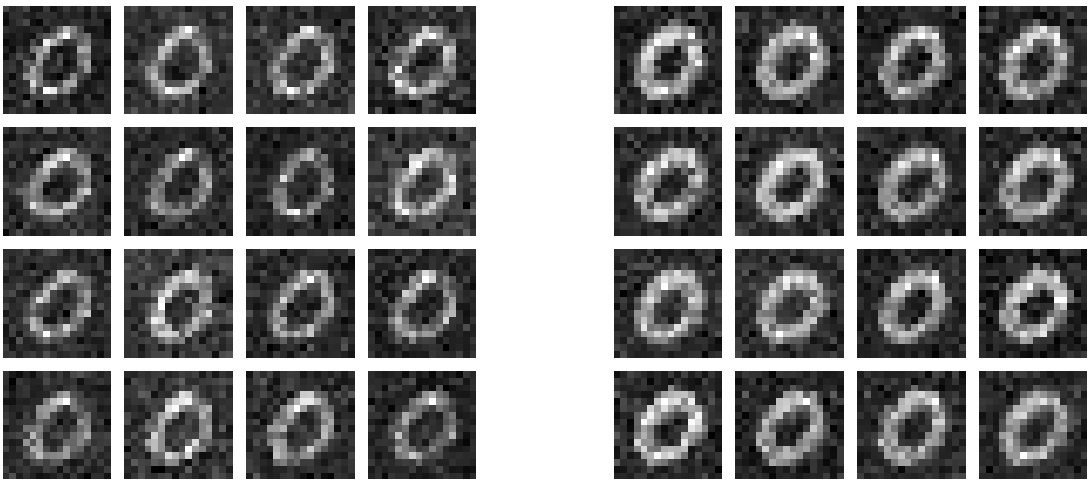
Figure 5.18: Results generated with the base and temporal models where $L = 10$.



(a) Base model, FID=305.5

(b) Temporal model, FID=243.8

Figure 5.19: Results generated with the base and temporal models where $L = 50$.



(a) Base model, FID=287.3

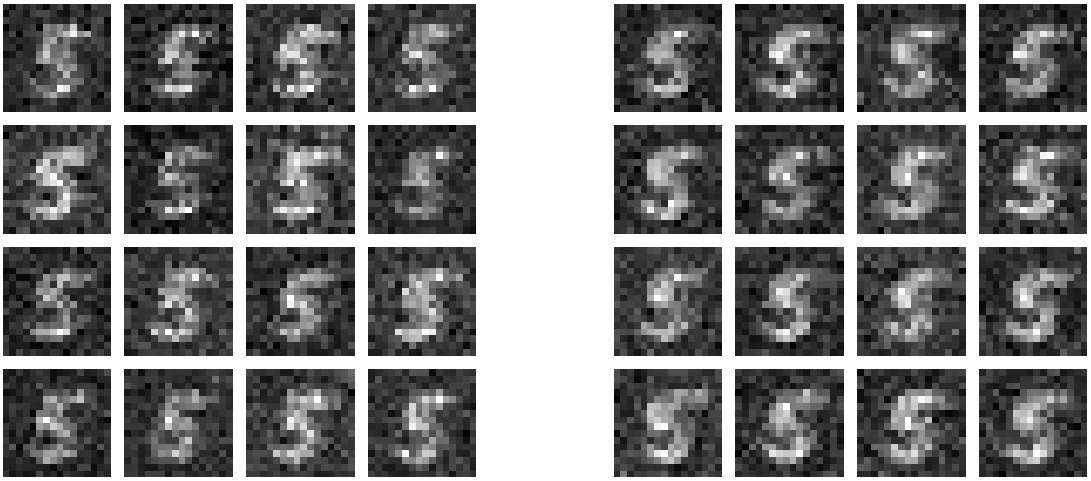
(b) Temporal model, FID=196.2

Figure 5.20: Results generated with the base and temporal models where $L = 100$.

Consistent with the FID scores in Figure 5.16, the temporal model initially underperforms the base model at low circuit depth ($L = 10$), as seen in Figure Figure 5.18. However, as the circuit depth increases, the temporal model surpasses the base model. This is evident in Figure 5.19 ($L = 50$), where the image generated with timestep embedding exhibits greater contrast and more aesthetically pleasing zeros.

In Figure 5.20 where the $L = 100$, the models produces some interesting results. The FID score in Figure Figure 5.16 indicates that the basic model's performance plateaus around $L = 50$. This aligns with the visual appearance of the image as there is no visual improvement in Figure 5.20a compared to Figure 5.19a. For the temporal model on the other hand, the FID score indicates that the model steadily keeps improving after $L = 50$. Comparing Figure 5.20b to Figure 5.19b, it can be observed that the image with $L = 100$ provides even greater clarity and contrast.

In Figure 5.21 we have included a comparison of the two models trained on the number "5". This is to illustrate how the different models react to more complex shapes.

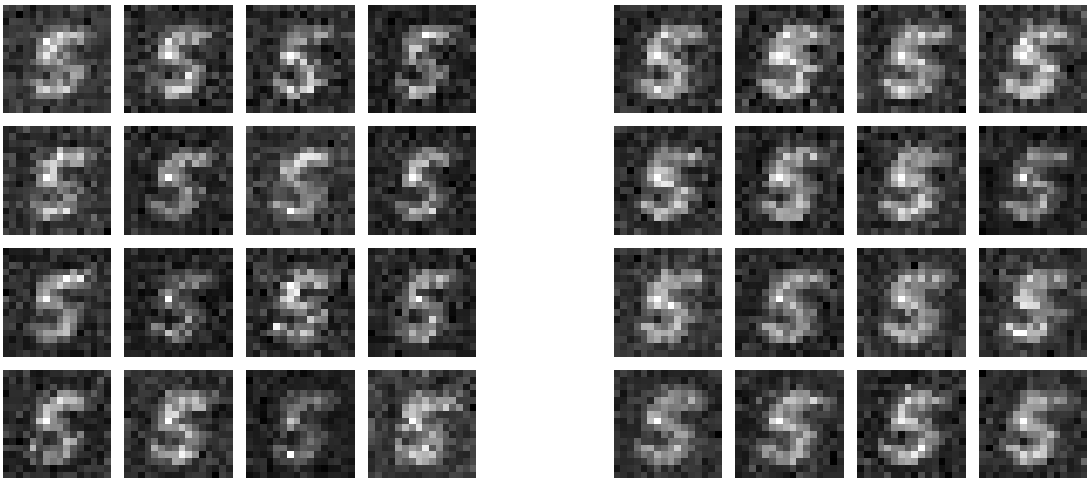


(a) Base model, FID=523.4

(b) Temporal model, FID=488.20

Figure 5.21: Results generated with the base and temporal models where $L = 50$.

The model with timestep embedding gets a FID score of 488.20, while the basic model gets 523.4. The image with timestep embedding looks consistently cleaner. The timestep embedding out-performance becomes more evident in Figure 5.22, where $L = 100$.



(a) Base model, FID=502.7

(b) Temporal model, FID=395.7

Figure 5.22: Results generated with the base and temporal models where $L = 100$.

Exploring the relationship between the number of timesteps (τ) and performance, analogous to the study in Figure 5.12, can give valuable insights. Figure 5.23 shows a comparison between the base model and the temporal model tested with different τ values.

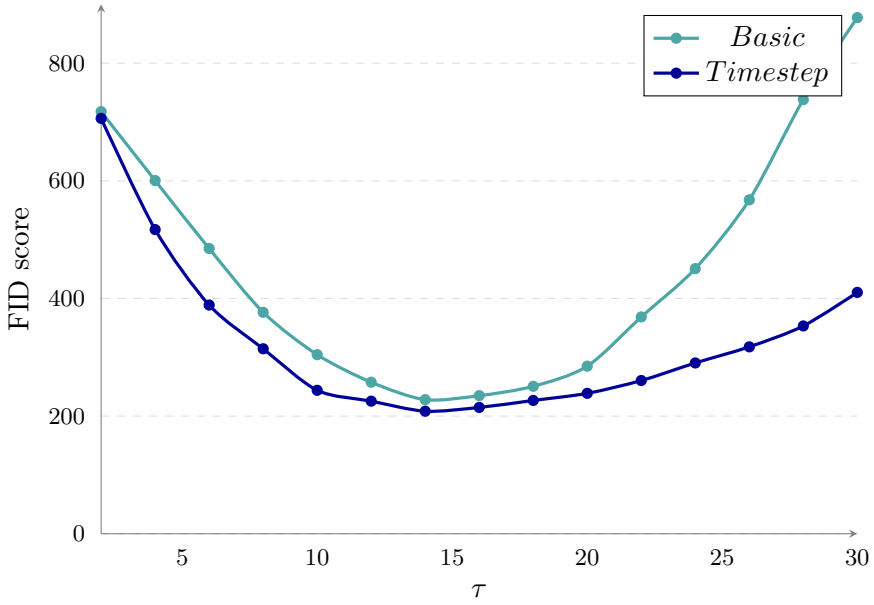


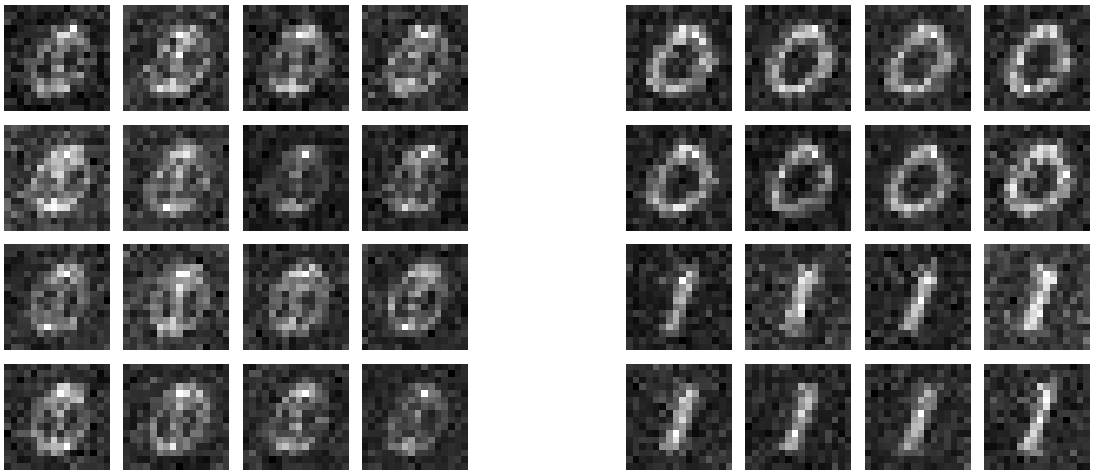
Figure 5.23: FID scores over τ for the base model and the temporal model.

The temporal model initially exhibits similar performance trends to the base model, with both improving up to $\tau = 14$. However, at higher timestep values, the temporal model demonstrates greater stability. This is evident in its slower rate of FID score increase compared to the base model.

5.4 Conditional Model

This section evaluates the conditional model, which has label embedding that enables the model to generate data conditioned on specific labels.

For our base model, we have only trained on data with the same label: 0. In this section, we will extend our training data to include multiple other digits. Figure 5.24a represents what the generated images would look like without label embedding, when trained on both "0"s and "1"s. The model lacks the ability to discriminate between the two digits, resulting in visually merged samples. Conversely, in Figure Figure 5.24b, we have asked the model to sample eight "0"s and eight "1"s, and it demonstrates its ability to generate distinct outputs for each label.



(a) The base model (b) The conditional model

Figure 5.24: Data labels [0,1] for the base and the conditional model.

Figure 5.25 displays FID scores and training time for the model with varying L. The model is trained on two labels (the digits "0" and "1").

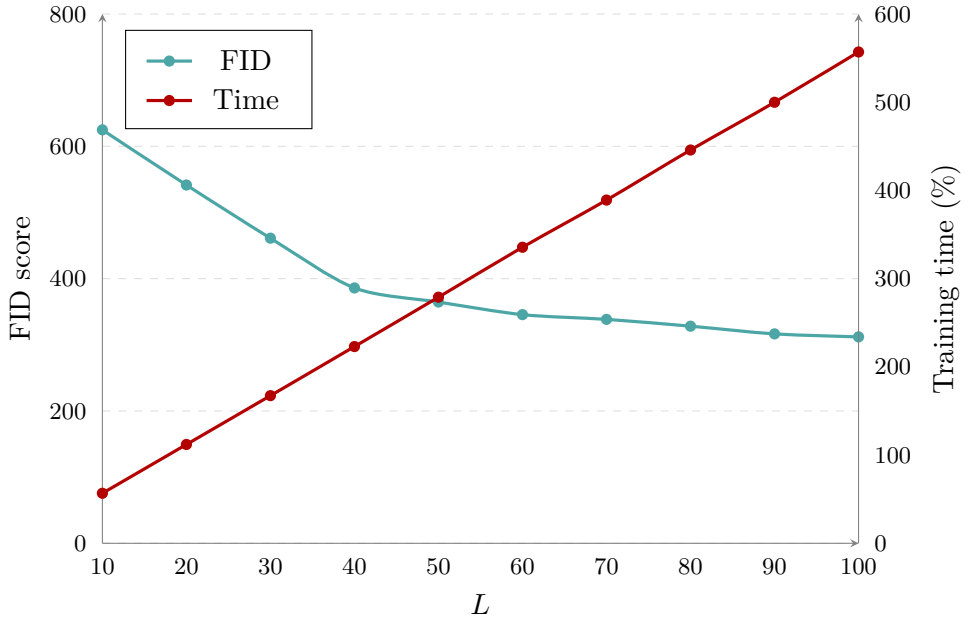
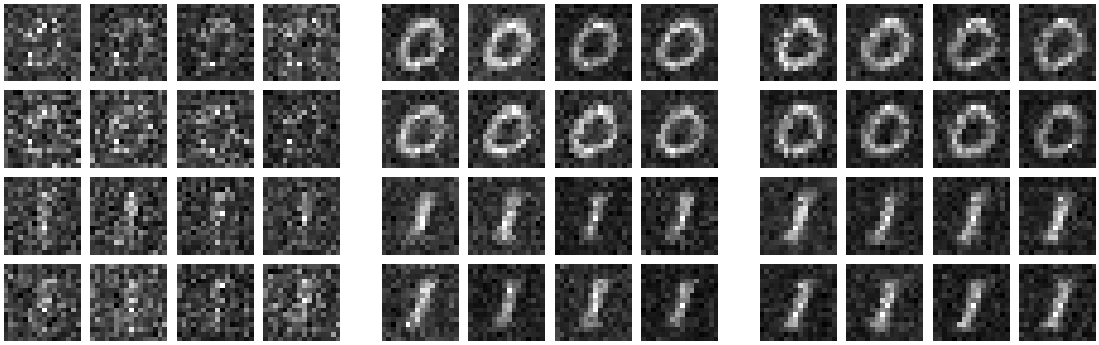


Figure 5.25: FID score and training time over L with two labels, averaged over 3 runs.

The FID scores in Figure 5.25 gradually improve from $L = 10$ all the way to $L = 100$. The slope of FID score decrease is slowly shrinking with each additional L increase, suggesting diminishing returns. Moreover, and consistent with the rest of the models, increasing L shows a linear increase in training time for the conditional model as well.

Figure 5.26 shows some of the images the model sampled throughout the training process in Figure 5.25. The first eight images are of the digit "0", and the last eight images are of the digit "1".



(a) $L = 10$, FID=624.8 (b) $L = 50$, FID=364.6 (c) $L = 100$, FID=311.9

Figure 5.26: Three different L values from Figure 5.25.

While the model successfully distinguishes between two labels, it is crucial to evaluate its performance on a wider range. Figure 5.27 presents six different models, each trained on an increasing number of labels (1 through 6).

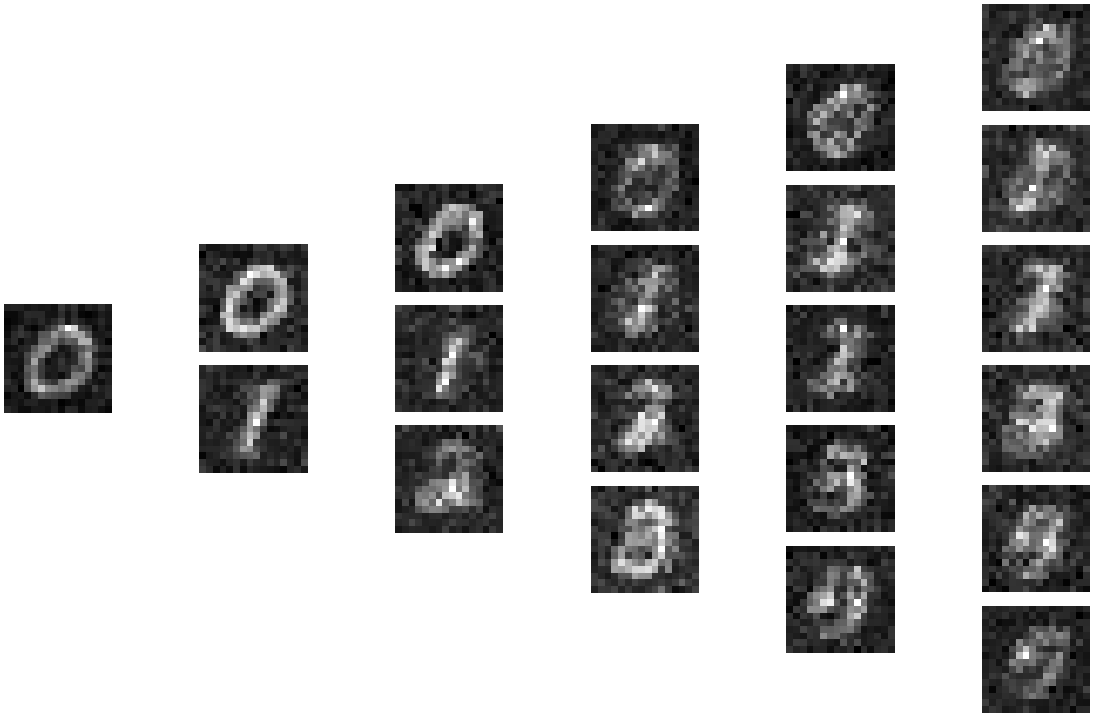


Figure 5.27: The conditional model with increasing amount of labels.

Figure 5.27 suggests a trend where the model's performance degrades with the introduction of more labels. The first few models show distinct digits, while the last few struggle to produce efficacious results, illustrating significant degradation in image quality as the number of conditions increases.

Figure 5.28 is a shows a model trained on all ten labels in the MNIST dataset. In this image, the first row is generated with the label "0", the second row is generated with the label "1", and so on so forth.

Figure 5.28: Trained on labels 0 through 9. 1 in upper row and 9 in lowest row.

As expected, none of the images in Figure 5.28 exhibit recognizable digits. Instead, the outputs appear blurry, indicating the model's inability to effectively distinguish between such a large number of labels.

5.5 Hybrid Model

After evaluating models with timestep embedding and label embedding separately, this section focuses on the hybrid model, combining both techniques using two ancilla qubits. Figure 5.29 shows a comparison between the conditional and hybrid models with varying L values.

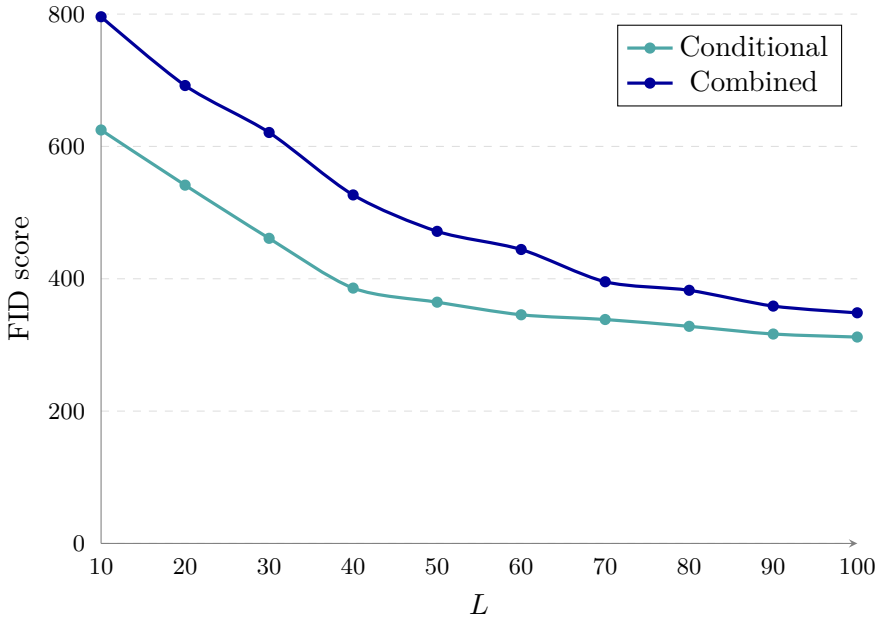


Figure 5.29: FID scores over L for the conditional and hybrid models.

The conditional model consistently achieves a lower FID score across all L values. However, the hybrid model's FID score steadily approaches that of the conditional model as L increases. This suggests the possibility that the hybrid model's training was prematurely terminated. To investigate this hypothesis, the models were retrained over 10 epochs instead of 3, with the results presented in Figure 5.30.

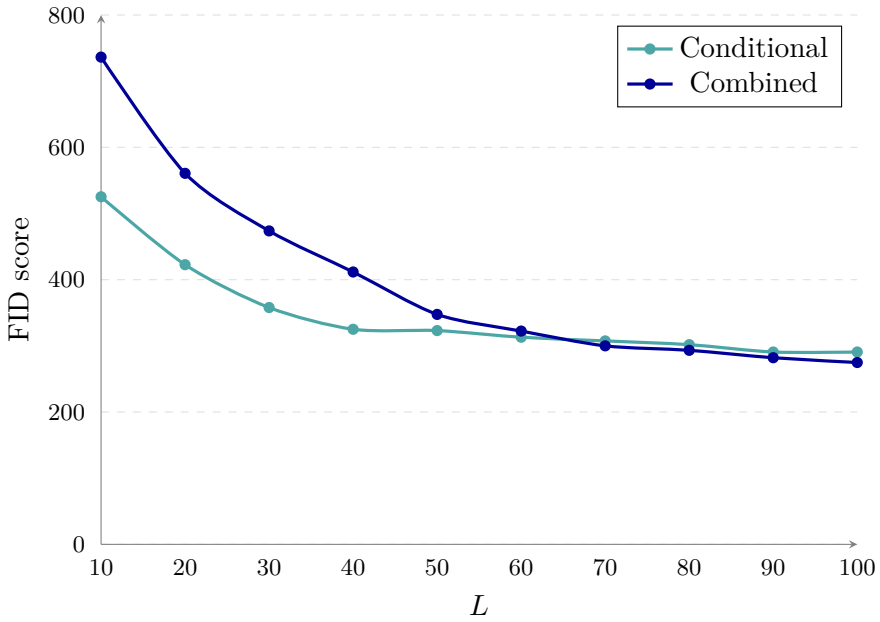
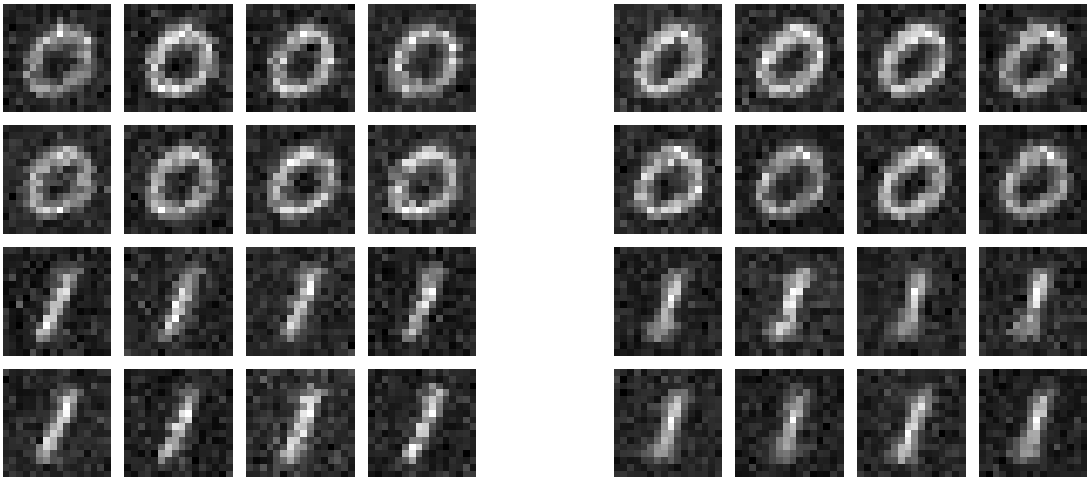


Figure 5.30: FID scores over L for the conditional and hybrid models where $E = 10$.

With longer training, the hybrid model surpasses the conditional model in terms of FID score at $L = 70$ and beyond.

Now it is time to visually compare the images generated by the hybrid model, with images generated by the conditional model. Figure 5.31 illustrates this comparison after

both models have been trained for 10 epochs.

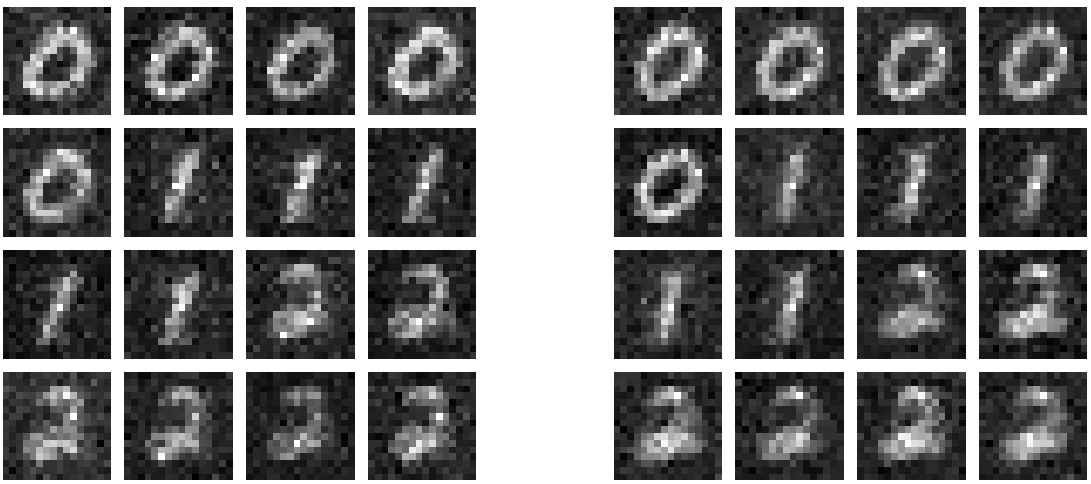


(a) Conditional model, FID=290.7

(b) Hybrid model, FID=274.8

Figure 5.31: Results generated with the conditional and hybrid models where $E = 10$ and $L = 100$.

Consistent with the FID scores, the initial visual comparison reveals minimal differences between the images. To explore the hybrid model's potential for greater gains, we will train a model with deeper circuits. Figure Figure 5.32 presents the results after 3 epochs with a circuit depth of 200 ($L = 200$).



(a) Conditional model, FID=296.3

(b) Hybrid model, FID=286.2

Figure 5.32: Results generated with the conditional and hybrid models where $E = 3$ and $L = 200$.

The image generated with the conditional model gets a FID score of 296.3 while the hybrid model's image gets 286.2, and the visual difference becomes more evident. The hybrid models image looks cleaner, especially if you compare the more complex shape of the "2".

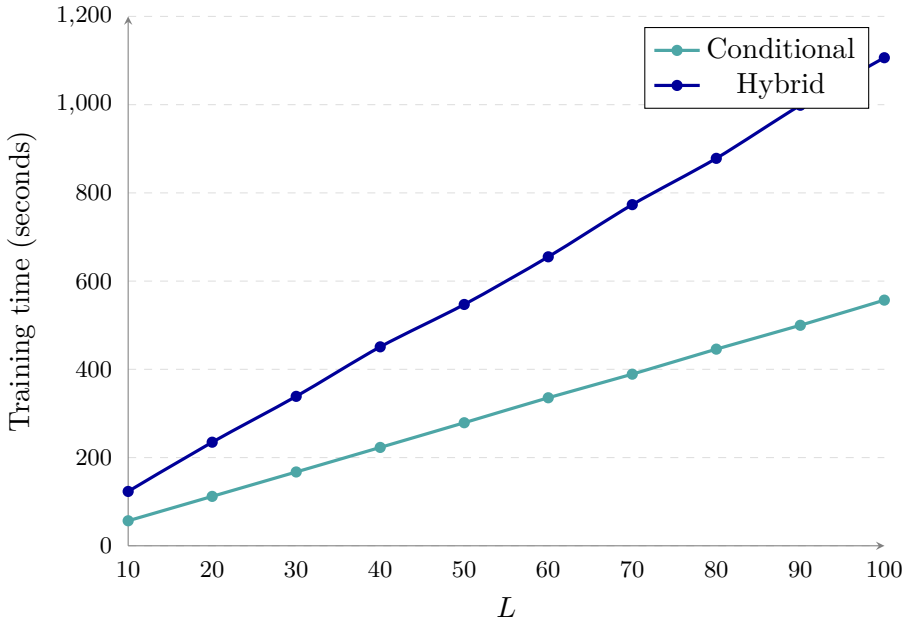


Figure 5.33: Training time over L for the conditional and hybrid models where $E = 3$.

Figure 5.33 plots the training times for the two models with an increasing L. The conditional model starts at 56.6 seconds at $L = 10$, and ends at 556.8 seconds at $L = 100$, similar results to the temporal model which also utilizes one ancilla qubit, seen in Figure 5.33. The hybrid model starts at 123.2 seconds on $L = 10$ and finishes on 1106.3 seconds on $L = 100$. Overall, the hybrid model averages at 101.6% increased training times compared to the conditional model.

5.6 RGB Images

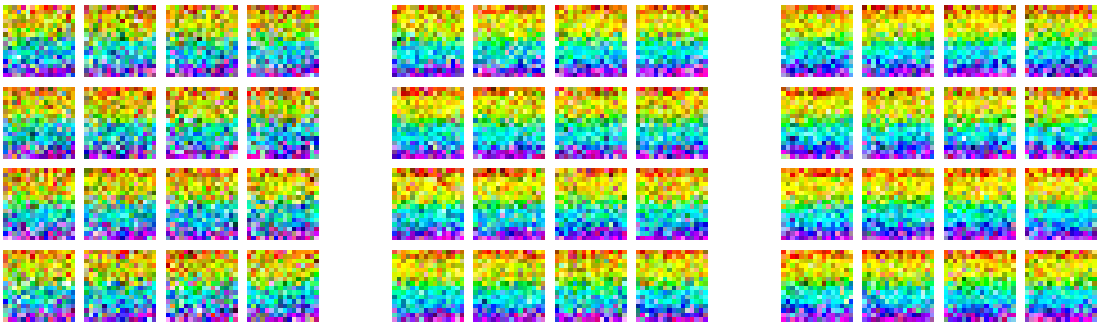
This section demonstrates our model variations' ability to generate colored images. Due to the small size of our custom colored datasets, as shown in Section 4.2, reliable FID calculations are not feasible. Instead, we will focus on a visual assessment of the models' capabilities. Moreover, given our extensive prior evaluation of the models, this section will center on exploring their color generation potential.

In Section 5.2.2 we determined the best Z value for the MNIST 16x16 dataset to be between $\zeta * 5$ to $\zeta * 8$. Although we cannot quantitatively confirm this, we have found that $\zeta * 2$ is a suitable value for the colored images. Thus, in this chapter, all of the generated images are using $Z = \zeta * 2$.

Additionally, we have increased the L parameter. The colored images are three times larger in terms of data points than the grayscale images in the MNIST dataset, due to its three channels. Thus, we found that increasing the L helped with generating efficacious results. Moreover, due to the limited amount of training data, we will be increasing the E parameter significantly.

5.6.1 Base Model

Figure 5.34 shows the results of the model trained on the rainbow dataset. The generated images are sampled from models trained with $E = 50$, and with three different circuit depths of $L = 60$, $L = 120$, and $L = 200$.

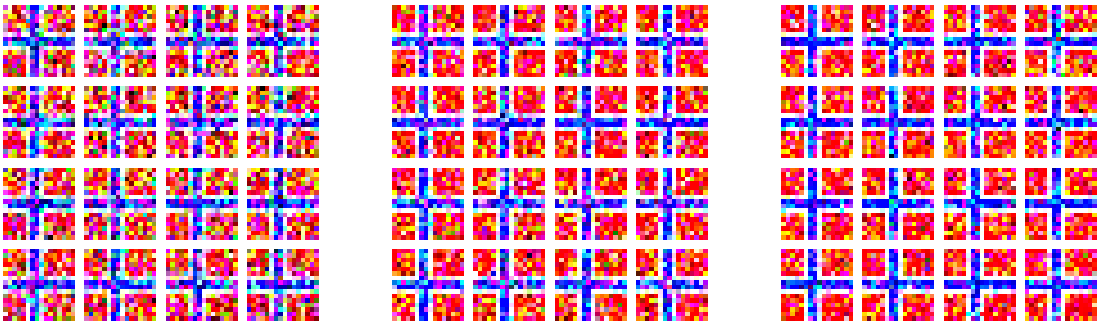


(a) $L = 60$ (b) $L = 120$ (c) $L = 200$

Figure 5.34: 50 epoch rainbow.

The results from Figure 5.34 do align with trends seen from Section 5.2.3, where increasing L generally produces better images. Notably, it appears that increasing L also increases the saturation of the image. Figure 5.34a appears to have generally dull colors, while Figure 5.34c has more vibrant and saturated colors.

Testing the models capability of recreating comform images, we have trained it on the Norwegian flag. Figure 5.35 shows the results with the same three circuit depths as used in Figure 5.34.



(a) $L = 60$ (b) $L = 120$ (c) $L = 200$

Figure 5.35: Only trained on Norwegian flag, 50 epoch.

Figure 5.35 confirms the assumption that a higher L produces better clarity and color accuracy. There is a clear distinction in noisiness as L increases, clearly visible in Figures 5.35a and 5.35c.

5.6.2 Timestep Embedding

When timestep embedding is added to the model, slight variations occur between the base model and the embedded one. Figures 5.36 and 5.37 shows the difference between these two models.

(a) Base model (b) Temporal model

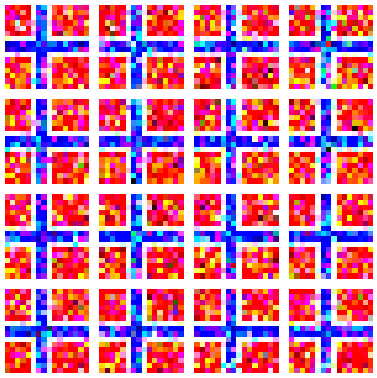
Figure 5.36: $E = 50$, $L = 120$.

(a) Base model (b) Temporal model

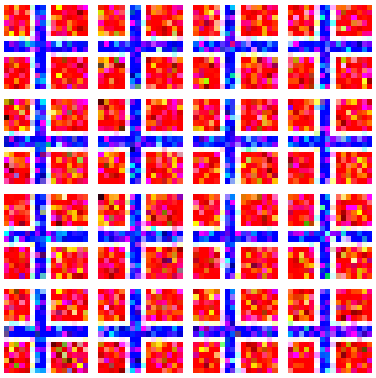
Figure 5.37: $E = 50$, $L = 200$.

It is difficult to concern any notable difference in quality between Figures 5.36a and 5.36b. However, in Figures 5.37a and 5.37b there is a clear difference in saturation, the temporal model appearing more saturated. Moreover, when comparing the two temporal model against each other (Figures 5.36b and 5.37b, the model trained with $L = 200$ displays a more balanced rainbow color distribution.

In Figure 5.38, the models are trained on only the Norwegian flag with $L = 200$. With these conform images, it becomes apparent that the images produced by the model with timestep embeddings inhibit much less noise.



(a) Base model

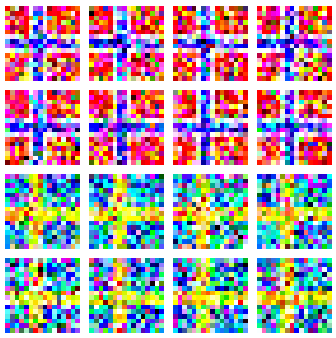


(b) Temporal model

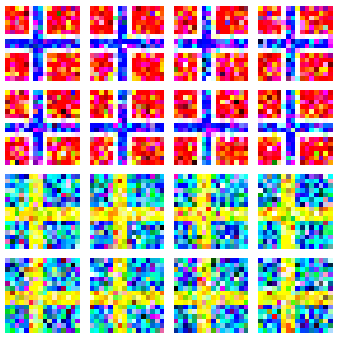
Figure 5.38: $E = 50$, $L = 200$.

5.6.3 Label Embedding

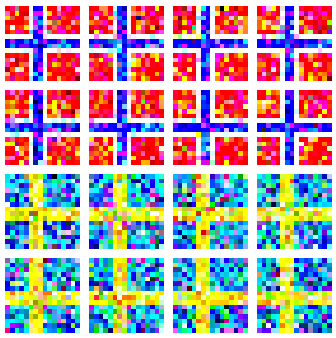
Figure 5.39 is generated by the conditional model, trained on the flags dataset containing both the Norwegian and Swedish flag. The model is fully capable of separating the two labels. Moreover, the model with $L = 120$ evidently performs better than the model with $L = 60$. This quality increase seemingly does not translate to the model with $L = 200$, where there is little to no difference to the images with $L = 120$.



(a) $L = 60$



(b) $L = 120$



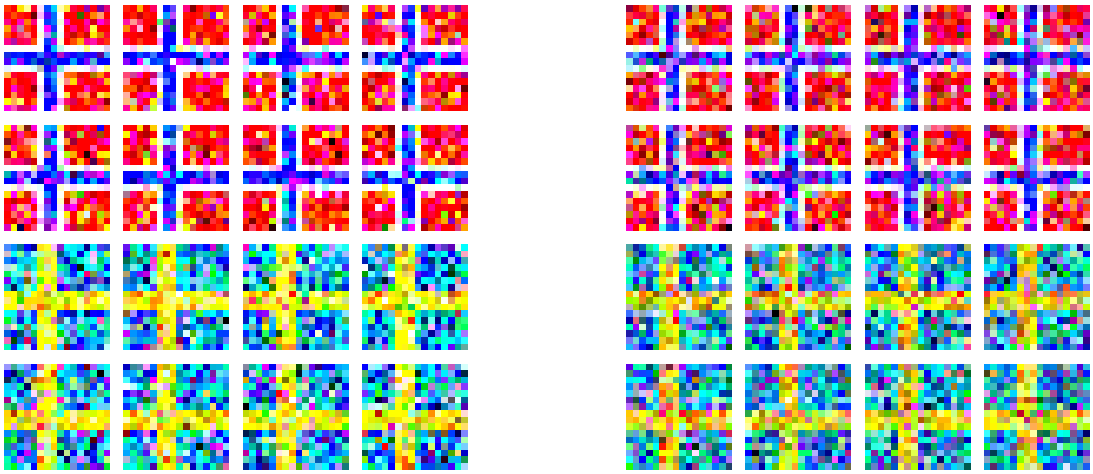
(c) $L = 200$

Figure 5.39: $E = 100$.

5.6.4 Hybrid

When we are testing the Hybrid model, the model is compared to the conditional model using the flags dataset. The models are trained over $E = 100$, due to the hybrid model requiring two ancillary qubits. From testing, we determined that using $E = 50$ as we have done in previous chapters led to an under-trained hybrid model.

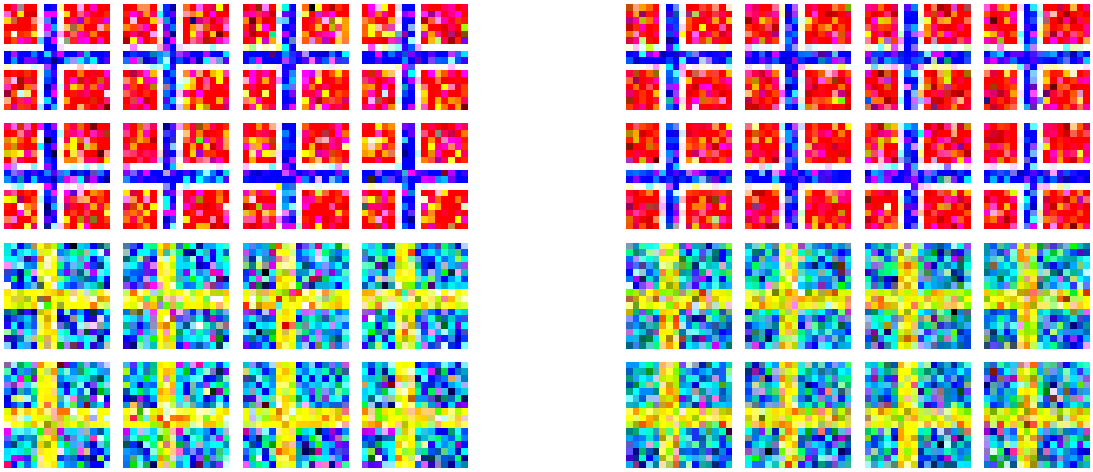
In Figure 5.40 we have trained both the conditional and the hybrid model with $L = 120$. Visually, these results appear very similar, indicating no clear benefit of using timestep embedding.



(a) Without timestep embedding (b) With timestep embedding

Figure 5.40: $E = 100$, $L = 120$.

For the results in Figure 5.41 we increased L to 200. In these images, however, there is a notable distinction between the two model variations. Timestep embedding does seem to play a role in reducing noise, particularly evident in the red portion of the Norwegian flag.



(a) Without timestep embedding (b) With timestep embedding

Figure 5.41: $E = 100$, $L = 200$.

Chapter 6

Discussion

In this section, we will discuss the key findings from the evaluation (Chapter 5) and address the research questions outlined in the introduction (Chapter 1). We will analyze the strengths and limitations of the models, compare them to existing work, and explore potential future directions for the models based on the insights gained.

6.1 Exploring the QDM Capabilities and Performance

This section presents an analysis of our QDM's performance. In particular, we will assess the QDM's ability to generate high-quality images, investigate how different hyperparameter configurations influence its performance, and analyze the role of computational resources and training data in shaping its effectiveness. This analysis directly addresses **RQ 1**:

> *What are the generation capabilities of our QDM, and how do factors such as computational resources, hyperparameters, and training data influence its performance?*

To address this question, we will analyze the impact of various hyperparameters on the QDM's image generation capabilities. This includes the ζ and Z hyperparameters responsible for noise modeling, quantum circuit depth L, the number of timesteps T, and the standard deviation σ controlling noise injection. We will then discuss the computational considerations and limitations that arise in the context of NISQ quantum devices.

6.1.1 Zeta and Sample Zeta

The ζ hyperparameter is an important factor in tailoring the QDM to effectively model the noise patterns present in image data. Our evaluation of ζ in Section 5.2.1 demonstrates a clear relationship between ζ and the QDM's image generation fidelity, as measured by the FID score. A well-calibrated ζ value is essential for achieving optimal results. An undersized ζ fails to fully represent the noise variability, while an inflated ζ leads to unrealistic levels of distortion.

Furthermore, the ζ hyperparameter and its behavior shed light on the nuanced challenges associated with noise modeling in a quantum computing context. The need for a flexible scaling mechanism like ζ reflects the diverse and dataset-specific nature of real-world image noise. Optimal zeta values will vary based on factors like image complexity, resolution, and potentially the underlying characteristics of the data. By

fine-tuning ζ, the QDM can adapt its noise generation to match the characteristics observed in a given dataset. Deeper investigations into this phenomenon could pave the way for the development of increasingly robust and versatile quantum-based image generation models with a wide range of potential applications.

The introduction of Z reveals an unexpected and intriguing aspect of the QDM's behavior during the image generation process. Experimental results from Section 5.2.2 indicate that inflating Z during sampling leads to superior model performance. Optimal Z values tend to be in the range of five to eight times the base ζ value for the MNIST 16x16 dataset, suggesting a critical distinction between the noise scaling requirements at different stages of the model's operation.

The necessity for a separate Z value raises a compelling question about the underlying mechanisms of the QDM. A few possible theories include:

Optimization Advantage The separate Z might provide a significant practical advantage during the sampling process. This could lead to faster convergence, improved overall denoising results, or reduced sensitivity to other hyperparameters during the sampling procedure.

Implicit Architectural Bias During training, the model architecture might inherently work best with small, incremental noise changes. The sampling process, with its initial burst of large-magnitude noise, might require adjustments to overcome potentially less optimal model behavior in this regime.

Gradient Guidance The inflated Z might enhance gradient signals used to guide the image generation process. Denoising steps can become less informative as the noise level decreases. As the noise level approaches zero, the model's task of predicting the clean image nears an identity mapping. This can diminish the error signal provided by gradients, potentially slowing convergence or making the model more susceptible to becoming trapped in local optima [13, 20]. Artificially magnifying the noise prediction could help maintain stronger gradients, offering clearer directions for the model to "walk back" towards a realistic image.

The distinct role of the Z hyperparameter highlights the complexities of noise modeling within the quantum generative diffusion framework. While our experimental results demonstrate its significance in achieving high-quality image generation, the exact reasons behind its effectiveness remain an area of future research. Further investigations are needed to fully understand the underlying principles and potentially reveal optimization techniques or architectural enhancements tailored for the sampling process in QDMs.

The overall idea of a sampling noise amplification technique was drawn from the work of Michael Kölle et al. on "Quantum Denoising Diffusion Models" [26]. Their model amplifies the noise in their sampling method by sampling over $T * 2$ steps. While both $T * 2$ sampling and the use of a Z value may influence the resulting output in a similar fashion, they are fundamentally different. When sampling $T * 2$ times, you take more small steps toward the final image output. In contrast, using Z sampling involves taking larger steps with an unchanged T value. Although similar in concept, our approach amplifies sampling convergence rather than increasing the number of sampling steps. We chose this method, rather than the proven $T*2$ approach, primarily to reduce sample time. The drawbacks of Z sampling over $T * 2$ sampling is that these larger strides may lead to less refined sampled data.

Looking beyond the QDM research field, examining the codebases of widely used classical diffusion model implementations, such as Stable Diffusion [45] and Imagen [46], also reveals varied sampling strategies. This variance highlights the ongoing exploration of noise manipulation techniques and underscores the need for further systematic analysis to establish optimal practices within the image generation process. The relationship between noise schedules, model architectures, and dataset characteristics remains a rich area for both theoretical and empirical investigation within the field of quantum-based generative modeling.

6.1.2 Quantum Circuit Depth

Understanding the significance of quantum circuit depth is crucial for optimizing quantum computations. It has a direct impact on the model's ability to represent complex quantum states and ultimately its performance. A shallow circuit might limit the complexity of functions the model can express, potentially hindering its effectiveness. Conversely, a deeper circuit, while allowing greater expressivity, increases the potential for errors due to noise and decoherence. This is especially critical on current NISQ devices, where noise levels are significant. Keeping the circuit shallow helps mitigate these issues and makes the computation more manageable and reliable. Striking the right balance between circuit depth and computational power is essential for achieving optimal results.

In Figure 5.9, we illustrate the relationship between circuit depth and training epochs, highlighting the consequence of minimizing circuit depth while preserving model performance. PQCs are particularly well-suited for this task. Their ability to dynamically adjust gate parameters allows for more compact encoding of information. This, combined with the training process of iteratively optimizing these parameters, enables the use of shallower circuits while still achieving the desired computational outcome. In these experiments we observed a significant advantage when reducing circuit depth while increasing training epochs: substantially lower sampling times. For the MNIST dataset (1024 images), a circuit depth of approximately 50 and training for 3 epochs provided a good balance between quality and low sampling times. Naturally, the optimal circuit depth and training time will vary across datasets, depending on factors such as image size, dataset size, and data complexity. It is crucial to recognize that pushing this strategy too far by using extremely shallow circuits and excessive training epochs can result in a deterioration of FID scores, indicating lower image quality. Therefore, there is a trade-off between lower sampling times and image quality.

Our hypothesis that increasing the value of the L parameter would enhance model efficacy was supported by our experiments. This aligns with the behavior observed in classical diffusion models, where deeper neural networks often yield better results [19]. As shown in Figure 5.10, increasing circuit depth led to a decrease in the FID score, indicating improved image quality. However, the graph also reveals a linear relationship between training time and circuit depth. Therefore, while deeper circuits might enhance image generation, this comes at the cost of increased computational demands. Striking a balance between image quality and computational efficiency is crucial when determining the optimal depth.

6.1.3 Number of Timesteps

Our experiments reveal that the optimal number of timesteps T for our QDM, trained on MNIST 16x16 images, lies within the range of 10 to 20. This is evidenced in Section 5.2.4,

where the FID score exhibits a parabolic trend. Lower and higher T values correlate with poorer FID scores, indicating lower image quality, while intermediate values demonstrate superior performance. Furthermore, the figure indicates a linear relationship between training times and the value of T; as T increases, training times proportionally increase. A similar linear relationship extends to sampling times, although not explicitly depicted in the graph.

The low T number stands in contrast with typical classical diffusion models, where the number of timesteps frequently extends into the hundreds [20, 46]. Several factors likely contribute to this phenomenon:

Reduced representational complexity means that smaller images inherently possess lower representational complexity compared to larger images. The diffusion process may require fewer iterative steps to converge on a satisfactory representation when the input space is less complex.

The PQC is the backbone of our QDM. In the context of smaller images, the size of the PQC can inherently be kept smaller, reducing the number of parameters. This reduction in circuit depth and parameter count might potentially accelerate convergence within the diffusion process. A smaller PQC likely requires fewer timesteps within the diffusion process for the model's internal representation to stabilize and converge.

Overfitting is a common problem for diffusion models, especially when trained for an excessive number of timesteps. Smaller image datasets might heighten this tendency. Limiting the T value serves as a regularization technique, helping to prevent the model from memorizing and improving its generalization ability. This phenomenon is observed in foundational diffusion model research. For instance, Ho et al. demonstrate that after a certain number of timesteps, image quality improvement may plateau or even decline, suggesting a form of overfitting [20].

While a precise, quantitative relationship between image size and the optimal T value remains an open research question, our findings resonate with established concepts in classical diffusion models. Larger images, due to their greater representational complexity, may benefit from a higher number of timesteps within the diffusion process to achieve thorough denoising and detail refinement. For instance, the work on Imagen by Saharia et al. [46] highlights the use of multiple cascaded diffusion steps for generating large, high-quality images, implying a link between complexity and more iterations [46]. Conversely, images with low complexity may require less T steps. While further research is needed to explicitly quantify this relationship, this intuition aligns with the principles of diffusion models and image representation.

Recognizing the relationship between image complexity and the optimal T hyperparameter carries implications for the development and deployment of QDMs. Further research is warranted to explore this in greater depth. Investigations across a broader range of image sizes and a comparison with classical diffusion model behavior would provide valuable insights. Additionally, exploring the interplay between quantum circuit design and the optimal number of timesteps could lead to the development of more efficient and effective QDMs.

The sensitivity of current NISQ devices to various forms of noise introduces an additional layer of complexity when selecting the optimal T value. As quantum circuits execute over multiple timesteps, errors can accumulate, potentially degrading the quality of generated images. This effect might be particularly pronounced for models with higher

T values. This concern resonates with our earlier discussion of the L hyperparameter in Section 6.1.2. Increasing circuit depth often leads to greater vulnerability to noise as errors can compound with each additional gate. Similarly, more timesteps increase the risk of noise adversely impacting the diffusion process, potentially offsetting the theoretical benefits of a larger T.

6.1.4 Standard Deviation

In Section 5.2.5, we discussed how different standard deviation (σ) values influence diffusion model performance. This hyperparameter is important, as it directly controls the level of randomness introduced during the image generation process. Our experiments highlighted its direct influence on the model's output quality.

When using a low σ value, the resulting images tend to be clean and sharp. However, this approach can limit the diversity of the generated samples. Think of it like this: adding noise with a low σ is similar to slightly scrambling a puzzle before reassembling it. Since the scrambling is minimal, the effort required to solve the puzzle is also minimal. In the context of diffusion models, a short denoising process means the model tends to follow similar paths each time, leading to highly generic and repetitive output.

In contrast, employing a high σ significantly increases the randomness of the data. Think of it as scrambling a puzzle so thoroughly that putting it back together becomes a much more complex task. In diffusion models, this means the denoising process becomes considerably more challenging. While this approach encourages greater diversity in the generated images, it also introduces significant noise into the final output.

A diffusion model that is limited to generating generic, similar samples undermines the fundamental purpose of this technique – the creation of diverse, novel synthetic images. Therefore, striking a balance between image quality and variation through careful noise level selection is crucial for successful diffusion model implementation.

As evident in Figure 5.15, our current computational resources limit the model's ability to effectively handle higher noise deviation levels. We found $\sigma = 0.2$ to be an optimal value for further experimentation. With this value, the generated samples exhibit a desirable degree of variation, while maintaining good visual quality.

6.1.5 Summary

Our investigation into the QDM's performance provides valuable insights that directly address **RQ 1**. Here are some of the key findings:

- **The Zeta Hyperparameters (ζ and Z):** Fine-tuning the ζ value is critical for properly modeling dataset-specific noise. We discovered that a separate, magnified Z value during sampling significantly boosts performance. This raises questions about optimization, model architecture biases, and gradient guidance within QDMs.

- **Quantum Circuit Depth (L):** PQCs allow us to balance shallow circuits with longer training epochs for faster sampling. A depth of around 50 with 3 epochs offered a good balance for MNIST. Deeper circuits (higher L values) improved quality at the cost of increased computational time.

- **Timesteps (T):** Surprisingly, we found that the optimal number of timesteps T for our QDM on the MNIST 16x16 dataset is relatively low (10-20). This contrasts with classical diffusion models and is likely due to a combination of

reduced representational complexity in smaller images and the inherent advantages of the PQC design. Limiting T may also help mitigate potential overfitting.

- **Standard Deviation (σ):** Aligning with diffusion model principles, the ideal σ balances image quality with diversity. Our resources limited the QDM's ability to handle high noise, with $\sigma = 0.2$ being a pragmatic choice.

These findings highlight both the unique strengths and limitations of our QDM implementation. They underscore the importance of hyperparameter tuning and the interplay between noise modeling, quantum circuit design, and computational constraints in the context of quantum-based generative diffusion models. These results, obtained using a noiseless simulator, may vary with different experimental setups. Further investigation with alternative quantum hardware or quantum simulators may yield different findings.

6.2 The Impact of Timestep Embedding

In this section, we look into the specific techniques we used to encode temporal information within our QDM. Our focus lies in understanding how this encoding impacts the characteristics of generated images. This investigation directly speaks to **RQ 2**:

> *How does timestep embedding influence the quality and coherence of images generated by quantum diffusion models?*

Specifically, we examine the effects of our ancilla qubit-based timestep embedding approach. We analyze its performance impact, discuss its inherent trade-offs, and consider its interplay with other crucial model parameters. We will also contrast our method with alternative embedding techniques proposed in the QDM field.

6.2.1 Improved Performance

In Section 5.3, we evaluated the performance of our temporal model using the MNIST 16x16 dataset. Our analysis reveals improved image synthesis due to the timestep embedding technique (see Figure 5.16). This improvement is evident both in quantitative metrics (FID score) and through visual inspection of the generated images. Compared to the base model, the temporal model consistently produces images with superior clarity and fidelity. Figure 5.16 also indicates that the temporal model initially performs worse than the base model, but after more training, it surpasses. This aligns with the established understanding in classical computing that deeper neural networks, despite increased training complexity, often yield superior results when optimized effectively [19].

The inclusion of temporal information also enhances the model's capacity to generate more complex shapes. As illustrated in Figure 5.22, the timestep-embedded model demonstrates superior ability in replicating the intricate details of the number "5" compared to the base model. We hypothesize that the timestep embedding provides the model with a finer-grained understanding of the denoising process, enabling it to better reconstruct the subtle curves and angles that define the shape of "5".

Visually, the generated numbers appear bolder and thicker compared to the output of the base model. This difference is particularly noticeable in Figure 5.22, where the contrast between the base and temporal models highlights the increased line

weight. However, this boldness effect seems less pronounced in the directed model (see Figure 5.31).

Moreover, the expanded qubit space of our model leads to a significantly larger space of possible states, thus contributing to slower training times [57]. One additional qubit is needed to embed temporal information into our model, which, due to superposition, means that the total number of probabilities the model predicts is 2^{n+1}, where n is the number of qubits needed to represent the image. In essence, this means our output probabilities are double in size when using one ancillary qubit, naturally increasing the computational requirements required to train the model.

To transition back into the original size of the image, we simply cut the size of the probabilities in half. Based on all of our testing, the model demonstrates good adaptability despite the fact that half of the collapsed values are discarded.

6.2.2 Over-Reliance on Embedding

While timestep embedding enhances the distinctness, clarity, and contrast of generated images, it comes with a notable downside: increased homogeneity in the outputs compared to the base model. This potential for decreased diversity may stem from the way timestep information is incorporated.

Classical diffusion models offer flexibility by allowing researchers to directly control the dimensionality of their timestep embedding vectors, enabling fine-grained adjustments to the amount of temporal context provided. In contrast, our quantum model uses an ancilla qubit for timestep embedding, potentially introducing a proportionally large amount of temporal context, particularly when working with small datasets like MNIST. This relatively strong emphasis on timestep information could lead to a degree of rigidity during the generation process, resulting in the observed increase in homogeneity. As discussed in Section 6.1.4, the variety of generated images is crucial.

6.2.3 Continuous Rotation Embedding

In our diffusion model, we encode temporal information within the ancilla qubit using rotations. The ancilla-data normalizing technique in Section 4.6.1 is crucial for this approach, ensuring that different timesteps are clearly distinguishable when applying Rx gates. Normalization uniformly distributes the T timesteps around the rotation axis, resulting in a continuous rotation direction as the timestep progresses. We hypothesize that this continuous rotation may bias the sampling process towards a narrower range of destinations, leading to less diversity in the generated samples.

While continuous rotation might initially seem like a limitation, we argue that its advantages outweigh potential drawbacks. Traditional conditional models, as discussed in Section 5.4, face challenges when handling increasing numbers of distinct labels. Their performance degrades significantly beyond 4 labels (see Figure 5.27). The nature of timestep encoding fundamentally differs from categorical label embeddings. In categorical models, labels like "0" and "1" have no intrinsic relationship. In contrast, timesteps within a diffusion process represent a continuous progression; steps $t = 4$ and $t = 5$ (out of $T = 10$) are neighboring positions in the denoising timeline. Therefore, the continuous rotation of the ancilla qubit may provide a more holistic temporal context that guides the denoising process, potentially contributing to high-quality image generation.

6.2.4 Performance Increase on High T

Our experiments revealed that the timestep embedding model exhibits greater resilience to higher timestep values compared to the base model. This is evident in Figure 5.23, where the FID score of the base model deteriorates rapidly as T exceeds 20, while the timestep model's decline is much more gradual.

The integration of temporal information likely allows the timestep model to maintain a clearer understanding of its position within the denoising process, enabling it to tolerate longer noising and denoising chains. This is promising, as generating larger, high-resolution images often necessitates a greater number of T steps. Our findings suggest that timestep embedding could be a valuable tool for generating such images due to its adaptability to higher T values.

6.2.5 Alternative Embedding Approach

In their work on "Quantum Generative Diffusion Model", Chuangtao Chen & Qinglin Zhao discuss a timestep embedding circuit [8]. This circuit, denoted $\tau(\omega, t')$, includes trainable parameters. Their circuit uses multiple layers consisting of single-qubit Rx and Ry rotation gates, in addition to a two-qubit ZZ gate. Their work is based on the quantum embedding method.

Our timestep embedding method differs slightly from this. Our approach involves normalizing the timestep value to fit within the 0 to 2π range for the Rx gate. Moreover, our approach only uses a single Rx gate compared to their multi-layer circuit.

The QGDM timestep circuit proposed by Chuangtao Chen & Qinglin Zhao offers more complex transformation of temporal information than our approach. Moreover, their inclusion of learnable parameters enables the circuit to learn optimal ways to encode this information during the training process. However, this added complexity comes at a cost, and can lead to higher computational costs and potentially make the model harder to train and optimize. However, our simpler method is well-suited to the experimental nature of this thesis, maintaining computational efficiency without sacrificing adequate timestep representation.

6.2.6 Summary

Our exploration of timestep embedding techniques offers valuable insights and addresses **RQ 2**.

The integration of timestep embedding has yielded promising results. By entangling an ancilla qubit with data qubits, we successfully encoded temporal information into our QDM. Experiments on the MNIST 16x16 dataset demonstrate the performance-enhancing nature of this approach, as evidenced by improved image quality (e.g., clarity, fidelity) and more complex shape representation.

However, certain trade-offs have been observed. While the distinctness and contrast of generated images increase with timestep embedding, there is a potential downside of decreased diversity. We speculate that the ancilla qubit approach might introduce a disproportionately large amount of temporal context, potentially limiting the range of possible output images.

The continuous rotation embedding technique we employ is thought to contribute to both advantages and limitations. While potentially reducing sample diversity, it may offer a more holistic temporal context that guides the denoising process. Additionally,

the timestep-embedded model shows greater tolerance to higher numbers of timesteps, a desirable trait when aiming to generate larger, higher-resolution images.

We acknowledge that alternative, more complex timestep embedding circuits exist, such as the one proposed by Chuangtao Chen & Qinglin Zhao. However, our simpler method aligns well with this thesis's experimental nature.

In conclusion, timestep embedding has proven to be an improvement for our QDM. It offers clear benefits in terms of image quality and coherence, but it is important to be mindful of potential trade-offs with respect to output diversity and training complexity. Further exploration of timestep encoding techniques promises to advance the field of quantum image generation.

6.3 Suitability for Colored Images

In this section, we will explore the model's performance on generating colored (RGB) images, directly addressing **RQ 3**:

> *Is the model suited for generating RGB images, and what are the potential limitations?*

The computational demands of modern RGB datasets like CIFAR-10, coupled with the scarcity of small (16x16) color datasets, led us to create our own datasets for our experiments. Do note that these datasets are very small: 10 rainbows and 2 flags. Though small, these datasets are perfectly suited for evaluating the practicality and feasibility of an RGB-supported QDM. The rainbow dataset tests the model's ability to recreate gradients, where hundreds of combinations red, green, and blue must be apparent. In contrast, the flags dataset measures its ability to learn specific structural features, such as the distinct shapes of flags.

Our evaluation of the RGB-adapted QDM revealed that its samples exhibited a generally noisier appearance compared to those generated with the MNIST dataset. The difference in how objectionable luminance (brightness) and chrominance (color) noise are perceived lies in the way the human visual system interprets them. Luminance noise can sometimes resemble traditional film grain, which may have an aesthetically pleasing quality. In contrast, chrominance noise introduces color distortions, which tend to be significantly more distracting to the eye. Since the MNIST dataset is grayscale, its noise is restricted to variations in brightness, resulting in subtle shades of gray that are generally less noticeable. Color images, however, encode information across multiple channels.

Noise within these channels can lead to significant deviations in hue and saturation, making the noise much more visually disruptive in color images. As color images have three channels (red, green, blue), the model must combine three circuit predictions to generate a single color. Even slight inaccuracies in any channel will result in a noticeable color deviation. This challenge is evident in the "flags" images generated in Chapter 5 (for example Figure 5.41). The Norwegian flag's solid red corners require a straightforward (255, 0, 0) prediction. In contrast, the Swedish flag's blue-green corner (0, 106, 167) necessitates a more complex combination, making it harder to predict accurately. The image demonstrates that the solid red exhibits significantly less noise than the Swedish flag's blended color.

Furthermore, the way noise affects our perception of an image depends heavily on the underlying structure of that image. This observation is underscored by the

contrast between the flag and the rainbow examples. When representing images with strict lines and distinct color blocks, like flags, any deviation from the intended pattern becomes readily apparent, making noise highly disruptive. Conversely, the rainbow's inherent blending of colors and lack of rigid boundaries creates a more forgiving visual environment. While noise might still be present, it is less noticeable because it does not clash with a strictly defined structure.

Considering these complexities, we are impressed by the model's ability to handle colored images. As seen in Section 5.6, the model demonstrates accurate color scope prediction for both the rainbow and flag datasets. While achieving realistic and noise-free color generation is inherently challenging (even in classical machine learning), we believe that with increased computational power, this model has the potential to produce visually appealing, high-definition color images in the future.

In this section, we have addressed **RQ 3** by shedding light on the intricacies of generating RGB images with our QDM implementation. While expanding to three channels poses challenges, the core technique remains unchanged from grayscale image generation. However, the perceptual impact of color inaccuracies intensifies, making the creation of visually pleasing colored images more demanding. Despite this, we believe our model is equally capable of handling color images with further training and deeper circuits.

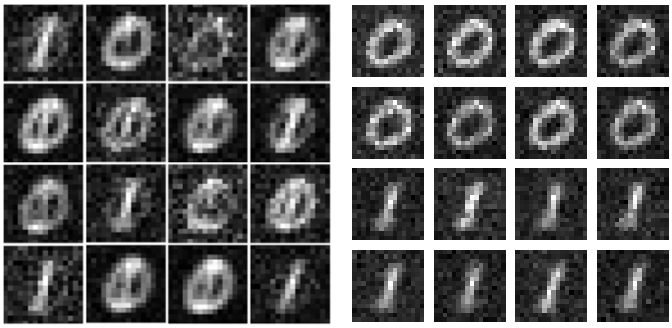
6.4 Comparative Evaluation

Quantum machine learning is a rapidly expanding field. When we began our thesis work, research uncovered only one relevant article: "Quantum-Noise-driven Generative Diffusion Models" [41]. This paper aimed to pave the way for new quantum-based generative diffusion algorithms. Their proposed "Classical-Quantum Generative Diffusion Model" (CQGDM) captured our attention, ultimately inspiring the core focus of our thesis. This work was purely theoretical, and our goal was then to fill the gap by creating a practical model implementation.

In the work titled "Quantum Diffusion Models" by Andrea Cacioppo et al. [7], they present a method of encoding classical data into quantum states using amplitude embedding. Additionally, they utilize a pretrained variational autoencoder to train the QDM with a low-dimensional representation of the data. While we find this technique interesting, and their paper outlines the positives of such an architecture, our comparison will focus solely on the quantum computations in the denoising process and not the latent models.

The authors introduce a label encoding technique for conditioning their quantum diffusion model. Their approach involves leveraging $log2(N)$ ancilla qubits to encode additional information for N labels. While this technique appears effective for their latent model, it did not seem to yield significant performance gains for the quantum model. Their experimenting with conditioning adding did not give relevant results differences.

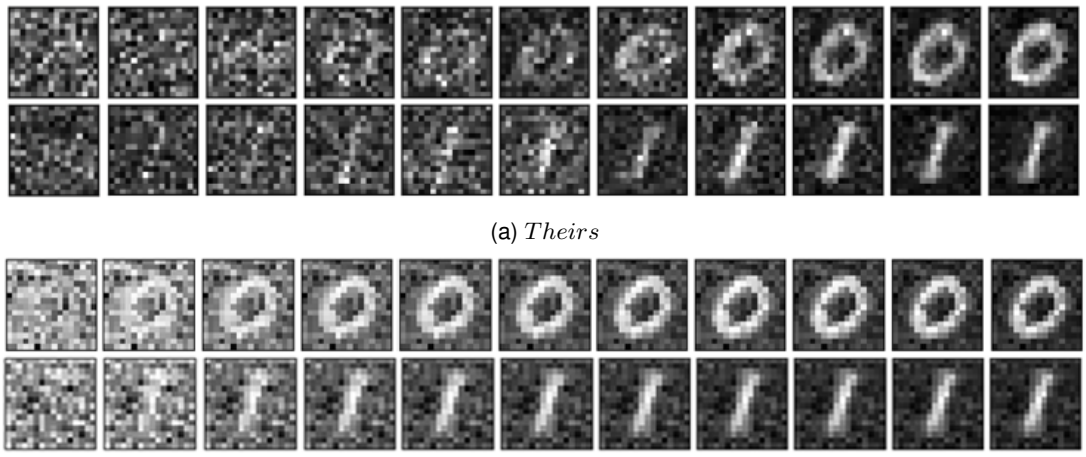
In Figure 6.1 we have displayed a side-by-side view of their and our models when trained on "0" and "1". The result shows a clear distinction of the two label embedding techniques.



(a) *Theirs* (b) *Ours*

Figure 6.1: Comparison between Andrea Cacioppo et al. and our QDM.

Furthermore, Figure 6.2 shows the model by Cacioppo et al. vs our model trained on 0 and 1, respectively.



(a) *Theirs*

(b) *Ours*

Figure 6.2: Comparison between Andrea Cacioppo et al. and our QDM.

The paper "Quantum Denoising Diffusion Models" by Michael Kölle et al. [26] presents an implementation of a quantum diffusion model. Their open-source approach and the early access we were privileged to have offered invaluable insights for our own Pennylane implementation.

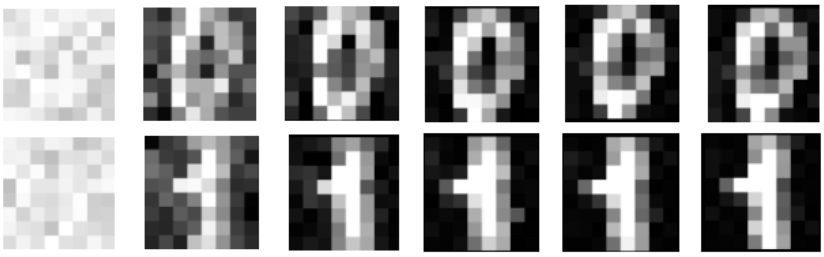
While our code draws inspiration from their work, it diverges in several fundamental ways. Their approach involves subtracting a constant from the predicted probability distribution, and then multiplying it by another constant. We observed that this method introduces a bias in the predicted noise, skewing it towards a particular direction. While this bias might align well with the MNIST dataset's predominantly dark background, it limits generalizability. Our solution addresses this by dynamically centering the noise average around 0, ensuring a more balanced distribution.

Another technique their model is utilizing which limits its genarailzability, is the usage of clamping. Every T iterations, the output values are clamped between 0 and 1. This means that after each noise prediction step and image update, pixels outside the 0 and 1 range are forced to be either entirely black or entirely white. While this approach might create high-contrast MNIST images with starkly defined backgrounds, it poses limitations in other tasks. For example, if the model needs to generate images

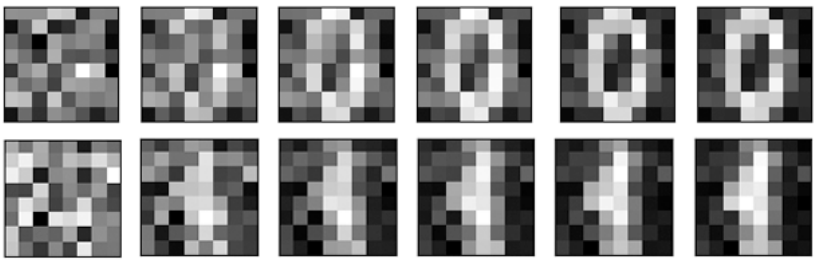
with subtle gradients or where fine-grained differences in shading are crucial, clamping introduces a significant constraint. The hard thresholding imposed by clamping can obscure these nuanced details. We have adaped our model to be independent of this constraint.

This paper presented the simple yet powerful idea of using only one ancilla qubit to encode label embedding. Their technique is to normalize the label array between the values 0 and 1. However, as explained in Section 4.6.1, normalizing within this range can be problematic, especially for more than two labels. This is due to the difficulty in differentiating between rotation angles, since a qubit's state is defined by a rotation of up to 2π radians. To address this, we developed a label normalization technique that evenly spreads the labels across their designated rotation angles. This improvement also facilitates the implementation of our timestep embedding enhancement, which further utilizes the ancilla qubit and its rotation scheme.

Figure 6.3 compares our model with the one presented in the paper. To facilitate direct comparison, we trained our model on 8x8 images, matching their format. Images are sampled at every other iteration (τ). The difference in background clarity is evident, highlighting the impact of their clamping technique.



(a) $Theirs$



(b) $Ours$

Figure 6.3: Comparison between Michael Kölle et al. and our QDM.

6.5 Challenges and Considerations

The quantum computing field is exhilarating, and developing one of the world's first quantum diffusion models marks a significant milestone. However, it is crucial to temper the excitement of innovation with a critical perspective. While we are excited about the model's success, we must also acknowledge its limitations and challenges. In this section, we will shed light on some of these.

6.5.1 Lack of Variation

An essential characteristic of diffusion models is their ability to generate new and varied data. However, we have observed that our model tends to produce outputs with limited

variety. This restricted variability can impact the model's effectiveness in broader applications, where a wide range of outcomes is necessary to simulate real-world quantum phenomena accurately.

We believe there are several contributing factors as to why our model tends to generate relatively homogeneous samples:

- As we discussed in Section 6.1.3, we use a comparatively low number of timesteps τ. This limits the model's ability to explore different possibilities during generation, leading to less variation in the output.
- The deterministic nature of DDIMs, which our diffusion model type Section 3.1, reduces the model's ability to explore the full probability distribution compared to DDPMs, potentially leading to less variety in outputs.
- As mentioned in Chapter 5, we limited our training to 1024 data. This number might be too low, and the model is overfitting.
- We are using a linear noise scheduling algorithm when applying noise to the training images. Different schedules might encourage more exploration during the denoising process.
- Our PQC is very architecturally simple, and thus might have limited expressivity. Adding more diverse layers, such as convolutional and attention-based, would provide a wider range of transformations potentially enabling the model to learn more nuanced patterns.

Addressing the lack of variation in our generated samples will be crucial for further experimentation with the model. Strategies such as increasing the number of timesteps (along with careful hyperparameter tuning), switching to a DDPM approach, enlarging the training dataset, exploring alternative noise schedules, and enhancing architectural complexity all hold the potential to significantly improve the diversity of our model's output. These modifications may enable the model to sample more efficacious results, although at a cost of computational resources.

6.5.2 Increasing the Number of Labels

The observations in Section 5.4 highlighted the limitations of label embedding with a single ancillary qubit. We believe the core problem lies in the nature of quantum rotations: attempting to cram numerous labels into closely spaced rotation angles within the 0 to 2π range fundamentally limits the ability to distinguish these unique labels. This constraint becomes even more problematic as the number of labels, and potentially the complexity of the dataset, increases. This is evident from the numerous images produced by models trained on various labels as shown in Figure 5.27.

In their work, Cacioppo et al. address the challenge of label encoding by leveraging the principle of quantum superposition [7]. A set of n qubits, capable of existing in 2^n unique states concurrently, provides the representational capacity. They use a recursive circuit structure to efficiently map a large number of labels onto this superposition. The base case lies in encoding two labels with a single qubit, where its basis states ($|0\rangle$ and $|1\rangle$) correspond to the labels. For larger label sets, the circuit adds qubits in a tree-like structure. This strategy enables the representation of numerous labels while minimizing the required number of qubits.

For scenarios with a modest number of labels (e.g., three), we have shown that a single ancilla qubit approach can still be suitable. This can reduce the number of qubits needed for label embedding, which is something we believe can be impactful during the NISQ-era, as the number of qubits is particularly low. The third column in Figure 5.27 highlights the fact that this technique works, and shows three distinct numbers produced by the model with single-qubit label embedding. For scenarios with a high number of labels, the $\log_2 n$ encoding scheme presented by Cacioppo et al. is a more appropriate, scalable choice.

6.5.3 Running on Real-World Quantum Hardware

In our work with PennyLane, we use `qml.probs` to simulate quantum measurement and derive the probability distribution across qubit states. This function effectively simulates the collapse of qubit states into measurable outcomes, providing a highly efficient way to explore our model behavior within a perfect simulator. This probability distribution turns out to be our generated image as explained in Section 4.5. However, real-world quantum hardware presents a fundamental constraint: it does not have direct state vector access.

One solution to this constraint is to turn to quantum state tomography (QST) [53]. QST allows us to reconstruct the state of our qubits through repeated executions of the circuit and statistical analysis of the outcomes. While it allows us to gain information from real devices, it has limitations. Each run produces a definite outcome, not the ideal probability distribution. Approximating the distribution requires many runs, introducing computational overhead. Furthermore, traditional QST implementations suffer from exponential scaling with the number of qubits, which can become a significant bottleneck.

Fortunately, QST research is a vibrant field with a focus on scalability and reduction in computational cost. Newer QST methods uses techniques like thresholding and deep learning to offer more efficient reconstruction [5, 11, 27, 51]. Exploration within this space could help mitigate the overhead associated with running QDMs on real quantum devices.

In addition to the potential benefits of efficient QST methods, our QDM approach inherently requires few qubits, further reducing the overhead for practical implementations. This translates to a significant advantage for QDMs in generating large-scale images. While classical diffusion models face computational bottlenecks due to complex neural network operations, QDMs offer a remarkably scalable solution. Each additional qubit exponentially increases the potential image size. For example, a full-color 1920x1080 HD image can be represented using just $log2(1920 * 1080 * 3) = 23$ qubits.

Noise poses a considerable obstacle for running QDMs on current quantum hardware. While simulations in idealized (noiseless simulation) environments are useful, real-world devices, especially those in the NISQ era, introduce noise that can potentially affect the fidelity of generated images. Sources like gate errors, decoherence, and readout errors can introduce imperfections into the QDM's processes. Gate errors might lead to the application of slightly incorrect operations, causing the qubit states to subtly shift over time. Decoherence can partially disrupt quantum superpositions, making it harder for the QDM to maintain the complex probability distributions essential for a clear image. Readout errors may add slight inconsistencies in the final measurements, possibly affecting individual pixel intensity values. While the extent of their impact is difficult to predict, the presence of these errors could contribute to varying degrees of

blurriness, artifacts, or inaccuracies in the final result. QDMs are particularly vulnerable to these issues due to their iterative nature, where errors can potentially compound with each step of the diffusion process.

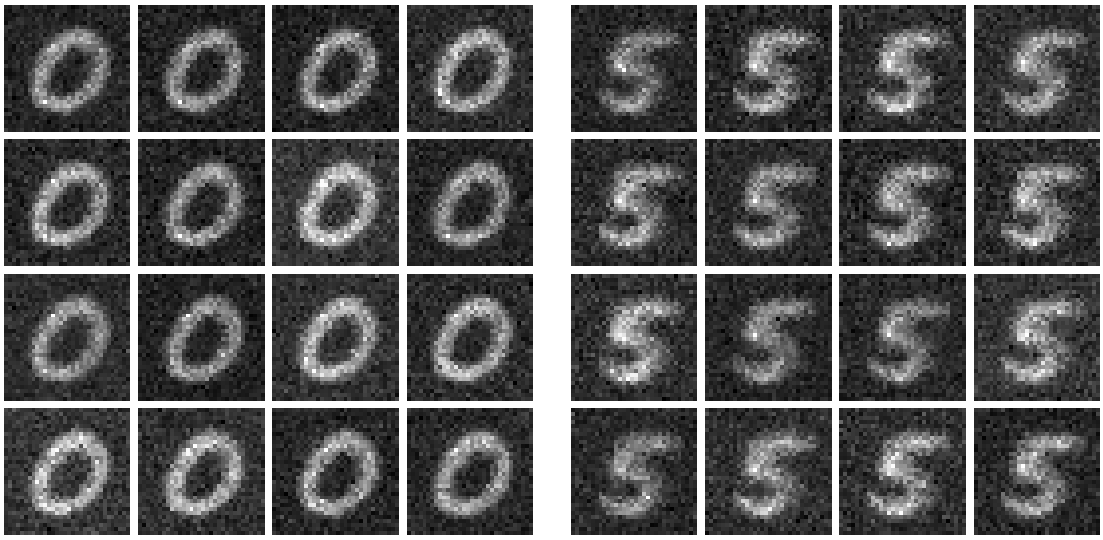
To mitigate these effects, techniques for noise reduction and error correction are important. In the near-term, while full-scale error correction may be out of reach, noise reduction and error mitigation offer practical strategies to improve the performance of noisy quantum circuits. These strategies aim to reduce the impact of noise on quantum computations, either by modifying circuit design or by applying specialized control techniques [14]. Additionally, machine learning offers promising avenues for designing quantum circuits that are inherently more resilient to noise [10], and for learning quantum computer noise effects for filtering [38].

In conclusion, while the convenience of probability estimation in noiseless simulation environments is not directly mirrored on real-world quantum devices, QST techniques provide a means to extract crucial information. Ongoing research into scalable QST solutions is essential for unlocking the potential of QDMs on future quantum hardware. Moreover, addressing noise through error mitigation strategies and the exploration of machine learning-based circuit optimization may be important for ensuring the model produces high-quality, visually appealing results is important for near-term execution on real hardware.

6.6 Pushing the Models

Due to the experimental focus of previous chapters, namely Sections 5.2, 5.3, 5.4, 5.5 and 5.6, the model was not trained to its full potential. In this section, we will assess its maximum performance within our hardware constraints, using deeper circuits and higher-resolution images. For optimal results, we will employ our top-performing model variants incorporating timestep embedding and, where applicable, label embedding.

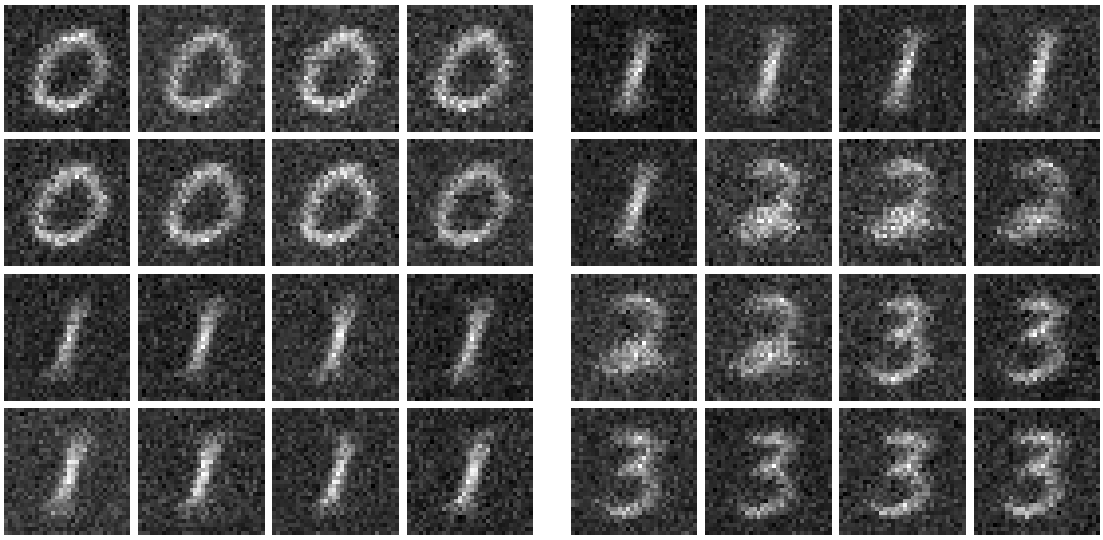
In Figure 6.4, we showcase two generated images temporal model, one trained on the digit "0" (Figure 6.4a) and one trained on the digit "5" (Figure 6.4b). Moreover, we scaled the MNIST dataset up to 32x32 pixels.



(a) $E = 3$, $L = 200$ (b) $E = 3$, $L = 200$

Figure 6.4: The temporal model trained on MNIST.

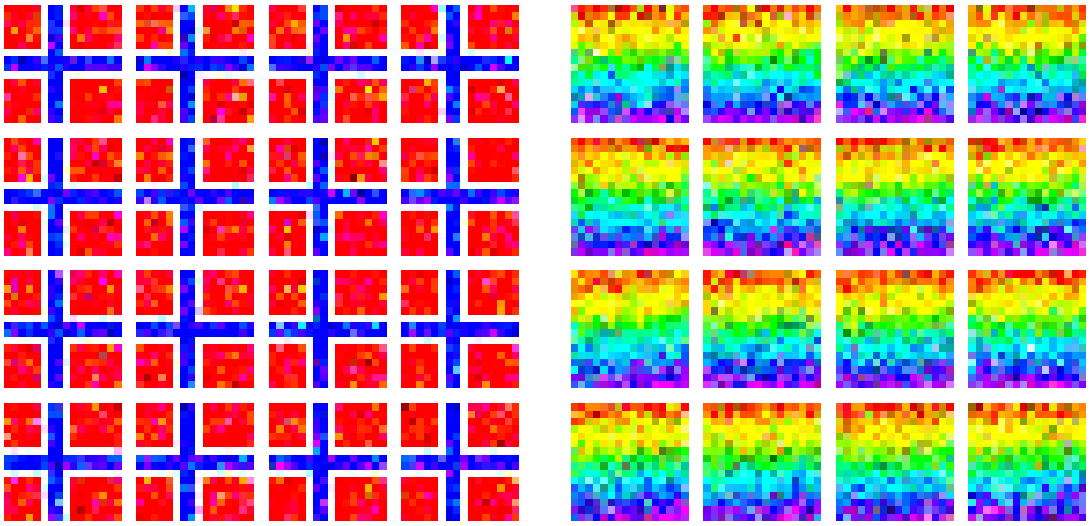
Figure 6.5 shows two images generated by the hybrid model. Contrary to the previous images, we sample multiple labels. Figure 6.5a is trained on the labels "0" and "1", while Figure 6.5b is trained on the labels "1", "2", and "3".



(a) $E = 3$, $L = 100$ (b) $E = 5$, $L = 100$

Figure 6.5: The hybrid model trained on MNIST.

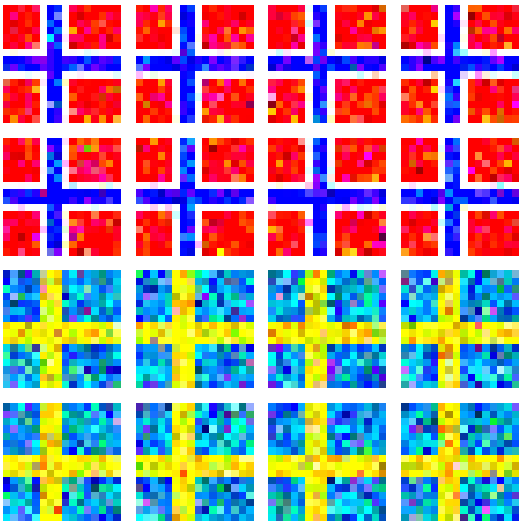
In Figure 6.6, we generate two colored images with the temporal model. Figure 6.6a is trained on the flag dataset, and Figure 6.6b is trained on the rainbows dataset.



(a) $E = 500$, $L = 300$ (b) $E = 200$, $L = 200$

Figure 6.6: The temporal model trained on the flag and rainbows datasets.

Figure 6.7 shows the hybrid model trained on the flags dataset.



(a) $E = 300$, $L = 250$

Figure 6.7: The hybrid model trained on the flags dataset.

Chapter 7

Conclusion

This thesis investigated the potential of QDMs for image generation tasks. Guided by specific research questions, we analyzed the generative abilities of our QDM, the influence of timestep embedding techniques, and their potential for extending into RGB image synthesis.

Our investigation into the generation capabilities of our QDM has demonstrated the ability of our QDM to generate images while uncovering how it is shaped by computational resources, hyperparameters, and training data. The findings suggest that quantum techniques hold potential for enhancing diffusion-based models, although practical advantages regarding resource reduction and speed will require further research, both theoretical and practical.

Our exploration highlighted the importance of adapting timestep embedding techniques for use within the quantum domain. We prove that timestep embedding has a positive influence on the quality and coherence of generated images by our QDM, at the cost of computational resources. Continued refinement of quantum timestep embedding holds the key to unlocking greater temporal modeling capabilities within QDMs.

While our model focused on grayscale image generation, we also extended our work into the complexities involved in adapting the QDM for RGB image synthesis. We outline the computational demands and potential limitations, including the increased visibility of noise and its potential impact on image quality. Despite these challenges, the model demonstrated promising adaptability and yielded impressive RGB results, suggesting a future potential for QDMs in this domain.

Furthermore we have implemented a novel qubit rotation normalization scheme, allowing embedding of multiple timesteps or labels using a single ancilla qubit. We devised a technique where the quantum probability distribution are mean-centered giving the ability to simulate both positive and negative noise. We introduced the constant ζ as a noise amplifier for more accurate noise modeling. Experimentation revealed that increasing ζ during sampling yielded superior results; we termed this amplification process Z.

Finally, after pushing our models to their limits (within our hardware constraints) in Section 6.6, we observed the generation of high-quality, visually impressing images. Qualitative analysis demonstrates the superior visual quality of these images when compared to those produced by comparable quantum diffusion models.

Overall, this thesis explores the potential of quantum diffusion models. It turns theoretical ideas into working examples, suggesting novel techniques, and pinpoints key challenges. As quantum computers become more powerful, we believe QDMs could

become important tools for generating images and similar tasks. This field deserves continued research and development.

7.1 Future Work

The findings within this thesis offer a strong basis for continued advancement of QDMs in the domain of image generation. To fully realize their potential, we will discuss several research directions that we find to be of importance.

Firstly, the importance of embedding techniques is reinforced by our results. Timestep embedding demonstrably improved the quality and coherence of generated images. However, as we target larger images and potentially greater τ values, the single-qubit approach may limit scalability. Continued investigation into more advanced embedding methods, potentially those with trainable parameters [8], is essential for future progress. Similarly, the challenges observed with label encoding highlight the need to explore multi-qubit encoding strategies [7]. Such techniques could provide the capacity necessary for image generation tasks with diverse class sets.

Addressing the observed homogeneity within generated images (see Section 6.5.1) is another important area. Architectural changes within the model, such as switching from a DDIM to a DDPM diffusion process and incorporating more intricate layer structures, could be explored to encourage greater diversity. Additionally, investigating the impact of larger datasets and images with higher τ values could shed light on their potential to increase variation in the outputs. Utilizing more powerful quantum simulation environments, and ultimately real quantum hardware, will be key to understanding both the model's ability to address the variability issue and its capacity for generating larger, more complex images.

A fascinating direction lies in the potential interplay between QDMs and quantum state tomography (QST) research as discussed in Section 6.5.3. The development of QST techniques specifically tailored to QDMs could greatly improve the efficiency and accuracy of state estimation, especially as model complexity grows. Understanding the structures used by QDMs in image encoding could lead to optimized state reconstruction methods. Conversely, the possibility of designing QDM circuits with QST constraints in mind could pave the way for reducing the required number of repetitions, ultimately enhancing efficiency.

Lastly, while this work has centered on noiseless simulations, future studies investigating QDM performance under more realistic conditions will be crucial. Training and testing the model on noisy quantum simulators will shed light on its robustness and could inspire the development of noise mitigation strategies. Ultimately, evaluating QDM behavior on actual quantum hardware will be essential to fully comprehend their potential and the challenges that may arise in the context of real-world applications.

Bibliography

[1] "A Survey on Generative Diffusion Models." In: *arXiv preprint arXiv:2209.02646* (2022). URL: https://ar5iv.labs.arxiv.org/html/2209.02646.

[2] Omer Bar-Tal et al. *Lumiere: A Space-Time Diffusion Model for Video Generation*. 2024. arXiv: 2401.12945 [cs.CV].

[3] Marcello Benedetti et al. "Parameterized quantum circuits as machine learning models." In: *Quantum Science and Technology* 4.4 (Nov. 2019), p. 043001. ISSN: 2058-9565. DOI: 10.1088/2058-9565/ab4eb5. URL: http://dx.doi.org/10.1088/2058-9565/ab4eb5.

[4] Ville Bergholm et al. *PennyLane: Automatic differentiation of hybrid quantum-classical computations*. 2022. arXiv: 1811.04968 [quant-ph].

[5] Daniele Binosi et al. *Threshold Quantum State Tomography*. 2024. arXiv: 2401.12864 [quant-ph].

[6] Chris M. Bishop. "Training with Noise is Equivalent to Tikhonov Regularization." In: *Neural Computation* 7.1 (1995), pp. 108–116. DOI: 10.1162/neco.1995.7.1.108.

[7] Andrea Cacioppo et al. *Quantum Diffusion Models*. 2023. arXiv: 2311.15444 [quant-ph].

[8] Chuangtao Chen and Qinglin Zhao. *Quantum Generative Diffusion Model*. 2024. arXiv: 2401.07039 [quant-ph].

[9] Isaac L. Chuang, Neil Gershenfeld, and Mark Kubinec. "Experimental Implementation of Fast Quantum Searching." In: *Phys. Rev. Lett.* 80 (15 1998), pp. 3408–3411. DOI: 10.1103/PhysRevLett.80.3408. URL: https://link.aps.org/doi/10.1103/PhysRevLett.80.3408.

[10] Lukasz Cincio et al. "Machine Learning of Noise-Resilient Quantum Circuits." In: *PRX Quantum* 2 (1 2021), p. 010324. DOI: 10.1103/PRXQuantum.2.010324. URL: https://link.aps.org/doi/10.1103/PRXQuantum.2.010324.

[11] Marcus Cramer et al. "Efficient quantum state tomography." In: *Nature communications* 1.1 (2010), p. 149.

[12] 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. URL: https://doi.org/10.1109%2Ftpami.2023.3261988.

[13] Prafulla Dhariwal and Alex Nichol. *Diffusion Models Beat GANs on Image Synthesis*. 2021. arXiv: 2105.05233 [cs.LG].

[14] Suguru Endo, Simon C. Benjamin, and Ying Li. "Practical Quantum Error Mitigation for Near-Future Applications." In: *Phys. Rev. X* 8 (3 2018), p. 031027. DOI: 10.1103/PhysRevX.8.031027. URL: https://link.aps.org/doi/10.1103/PhysRevX.8.031027.

[15] Richard P Feynman. "Simulating physics with computers." In: *International Journal of Theoretical Physics* 21.6 (1982), pp. 467–488.

[16] Zhujin Gao et al. *Empowering Diffusion Models on the Embedding Space for Text Generation*. 2024. arXiv: 2212.09412 [cs.CL].

[17] Ian Glendinning. "The bloch sphere." In: *QIA Meeting*. 2005.

[18] Lov K. Grover. *A fast quantum mechanical algorithm for database search*. 1996. arXiv: quant-ph/9605043 [quant-ph].

[19] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG].

[21] Jonathan Ho and Tim Salimans. *Classifier-Free Diffusion Guidance*. 2022. arXiv: 2207.12598 [cs.LG].

[22] L. Holmstrom and P. Koistinen. "Using additive noise in back-propagation training." In: *IEEE Transactions on Neural Networks* 3.1 (1992), pp. 24–38. DOI: 10.1109/72.105415.

[23] Takahiro Inagaki et al. "A Coherent Ising Machine for 2000-node Optimization Problems." In: *Science* 354.6312 (2016), pp. 603–606.

[24] Everypixel Journal. *AI Has Already Created As Many Images As Photographers Have Taken in 150 Years. Statistics for 2023*. 2023. URL: https://journal.everypixel.com/ai-image-statistics.

[25] Richard Jozsa and Noah Linden. "On the role of entanglement in quantum-computational speed-up." In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 459.2036 (Aug. 2003), pp. 2011–2032. ISSN: 1471-2946. DOI: 10.1098/rspa.2002.1097. URL: http://dx.doi.org/10.1098/rspa.2002.1097.

[26] Michael Kölle et al. *Improving Convergence for Quantum Variational Classifiers using Weight Re-Mapping*. 2023. arXiv: 2212.14807 [quant-ph].

[27] Dominik Koutný et al. "Neural-network quantum state tomography." In: *Physical Review A* 106.1 (July 2022). ISSN: 2469-9934. DOI: 10.1103/physreva.106.012409. URL: http://dx.doi.org/10.1103/PhysRevA.106.012409.

[28] Don S. Lemons. "An Introduction to Stochastic Processes in Physics." In: John Hopkins University Press, 2002. Chap. 5.2.

[29] Lijiang Li et al. *AutoDiffusion: Training-Free Optimization of Time Steps and Architectures for Automated Diffusion Model Acceleration*. 2023. arXiv: 2309.10438 [cs.CV].

[30] Wei Li et al. "An Image Classification Algorithm Based on Hybrid Quantum Classical Convolutional Neural Network." In: *Quantum Engineering* 2022 (July 2022), pp. 1–9. DOI: 10.1155/2022/5701479.

[31] Xin Li et al. *Diffusion Models for Image Restoration and Enhancement – A Comprehensive Survey*. 2023. arXiv: 2308.09388 [cs.CV].

[32] Dianhao Liu. "Principles and Characteristics of Quantum Computing." In: *Journal of Physics: Conference Series* 2014.1 (2021), p. 012012. DOI: 10.1088/1742-6596/2014/1/012012. URL: https://dx.doi.org/10.1088/1742-6596/2014/1/012012.

[33] Qiyu Liu. "Comparisons of Conventional Computing and Quantum Computing Approaches." In: *Highlights in Science, Engineering and Technology* 38 (Mar. 2023), pp. 502–507. DOI: 10.54097/hset.v38i.5875.

[34] Calvin Luo. *Understanding Diffusion Models: A Unified Perspective*. 2022. arXiv: 2208.11970 [cs.LG].

[35] I.U.I. Manin. *Vychislimoe i nevychislimoe*. Sov. radio, 1980. URL: https://books.google.no/books?id=pAo-zgEACAAJ.

[36] Gordon E. Moore. "Lithography and the future of Moore's Law." In: *IEEE Solid-State Circuits Society Newsletter* 11.3 (2006), pp. 37–42. DOI: 10.1109/N-SSC.2006.4785861.

[37] Emad Mostaque. *Stable Diffusion training time*. 2022. URL: https://twitter.com/EMostaque/status/1563870674111832066.

[38] Asmar Muqeet et al. *Mitigating Noise in Quantum Software Testing Using Machine Learning*. 2024. arXiv: 2306.16992 [cs.SE].

[39] Alex Nichol and Prafulla Dhariwal. *Improved Denoising Diffusion Probabilistic Models*. 2021. arXiv: 2102.09672 [cs.LG].

[40] OpenAI. *DALL-E 2*. 2022. URL: https://openai.com/dall-e-2.

[41] Marco Parigi, Stefano Martina, and Filippo Caruso. *Quantum-Noise-driven Generative Diffusion Models*. 2023. arXiv: 2308.12013 [quant-ph].

[42] Pennylane. *What are Bell states?* 2024. URL: https://pennylane.ai/qml/glossary/what-are-bell-states/.

[43] Prashant. *A Study on the basics of Quantum Computing*. 2007. arXiv: quant-ph/0511061 [quant-ph].

[44] John Preskill. "Quantum Computing in the NISQ era and beyond." In: *Quantum* 2 (Aug. 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. URL: http://dx.doi.org/10.22331/q-2018-08-06-79.

[45] Runway, CompVis, and Stability AI. *Stable Diffusion*. 2022. URL: https://stability.ai/.

[46] Chitwan Saharia et al. *Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding*. 2022. arXiv: 2205.11487 [cs.CV].

[47] Domain Of Science. *The Map Of Quantum Computing*. 2021. URL: https://www.flickr.com/photos/95869671@N08/51721957923/.

[48] sentdex. *Quantum Computer Programming w/ Qiskit*. Youtube. 2019. URL: https://youtu.be/aPCZcv-5qfA?si=2RvQC5WLl5R_cNQr&t=674.

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

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

[51] Roman Stricker et al. “Experimental Single-Setting Quantum State Tomography.” In: *PRX Quantum* 3.4 (Oct. 2022). ISSN: 2691-3399. DOI: 10.1103/prxquantum.3.040310. URL: http://dx.doi.org/10.1103/PRXQuantum.3.040310.

[52] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

[53] K. Vogel and H. Risken. “Determination of quasiprobability distributions in terms of probability distributions for the rotated quadrature phase.” In: *Phys. Rev. A* 40 (5 1989), pp. 2847–2849. DOI: 10.1103/PhysRevA.40.2847. URL: https://link.aps.org/doi/10.1103/PhysRevA.40.2847.

[54] Wikipedia. *BlochSphere*. 2024. URL: https://en.wikipedia.org/wiki/Bloch_sphere/.

[55] Wikipedia. *Quantum entanglement*. 2024. URL: https://en.wikipedia.org/wiki/Quantum_entanglement.

[56] Colin P. Williams. “Quantum Gates.” In: *Explorations in Quantum Computing*. London: Springer London, 2011, pp. 51–122. ISBN: 978-1-84628-887-6. DOI: 10.1007/978-1-84628-887-6_2. URL: https://doi.org/10.1007/978-1-84628-887-6_2.

[57] “Copyright.” In: *Quantum Machine Learning*. Ed. by Peter Wittek. Boston: Academic Press, 2014, p. iv. ISBN: 978-0-12-800953-6. DOI: https://doi.org/10.1016/B978-0-12-800953-6.00018-9. URL: https://www.sciencedirect.com/science/article/pii/B9780128009536000189.

[58] Julia Wolleb et al. *Diffusion Models for Medical Anomaly Detection*. 2022. arXiv: 2203.04306 [eess.IV].

[59] Xanadu. *PennyLane*. 2018. URL: https://pennylane.ai/.

[60] Xanadu. *PennyLane Devices*. 2018. URL: https://pennylane.ai/plugins/.

[61] Tianshuo Xu et al. *Towards Faster Training of Diffusion Models: An Inspiration of A Consistency Phenomenon*. 2024. arXiv: 2404.07946 [cs.LG].

Appendix A

QDM Sample Phase

Below is a snippet from the function responsible for handling sampling in our QDM. This function is shared between all of the various model variations. A more detailed walkthrough of the functionality of this function is found in Section 4.8.

Of note is that the `self._predict_noise()` function is shared between the sampling and the training phase. This function is found in Appendix C.

```
def sample(self, n, num, save=False) -> torch.Tensor:
    """
    Samples n_sample images from the model.

    Args:
        n (int): The number of samples to generate.
        save (bool): Whether to save the samples to /results.
    """
    self.eval()

    # Generate labels
    if self.embeds.label_embedding():
        labels = self.embeds.generate_labels(self.dataset.labels, n)

    # Sample T steps
    with torch.no_grad():

        # Generate random noise
        x = torch.normal(
            mean=self.hyperparams.mean,
            std=self.hyperparams.std,
            size=(n, self.dataset.num_features()),
        )

        # Predict the noise
        for i in range(self.hyperparams.T - 1, -1, -1):
            predicted_noise = self._predict_noise(
                x=x,
                labels=labels if self.embeds.label_embedding() else None,
                ts=self.ts[i] if self.embeds.timestep_embedding() else None,
            )
```

```
            # Remove the noise from the image
            x = x - predicted_noise

    # Reshape x from (n, num_features) to (n, 1, w, h)
    x = x.view(n, self.dataset.c, self.dataset.h, self.dataset.w)

    return x
```

Appendix B

QDM Training Phase

The following code snippet demonstrates the core training function of our QDM. This function is utilized across all model variations. For a comprehensive explanation of its operation, refer to Section 4.7.

The `self._predict_noise()` function is shared with the sampling phase. Details on this can be found in Appendix C.

```
def train_model(self, save_model=False):
    """
    Trains the model.

    Args:
        save_model (bool): Whether to save the model to "model.pth".
    """
    loss_func = torch.nn.MSELoss()
    optimizer = torch.optim.Adam(self.net.parameters(), lr=self.hyperparams.lr)

    self.train()

    for _ in tqdm(range(self.hyperparams.epochs), desc="Epoch"):
        for x, y in tqdm(self.dataset.dataloader, desc="Data", leave=False):
            optimizer.zero_grad()

            predicted_noise, actual_noise = self(x, y)

            loss = loss_func(predicted_noise, actual_noise)
            loss.backward()

            optimizer.step()
```

Appendix C

QDM Noise Prediction

This function is responsible for both calling and post-processing the outputs from the quantum circuit. It takes any input tensor x as input, and returns the noise prediction for that specific input. Subsequently, both the mean-centering post-processing and the ζ noise amplification detailed in Section 4.5 takes place here.

```
def _predict_noise(self, x, labels=None, ts=None):
    """
    Predicts the noise for the given input.

    Args:
        x (torch.Tensor): The input tensor.
        training (bool): Whether the model is training mode.
        labels (torch.Tensor): The labels tensor.
        ts (torch.Tensor): The timesteps tensor.
    """
    predicted_noise = self.net(
        inp=x,
        labels=labels,
        timesteps=ts,
    )

    # Sum the noise
    sums = predicted_noise.sum(dim=1)

    # Subtract the mean from the noise
    predicted_noise -= (sums / self.dataset.num_features()).view(-1, 1)

    # Multiply the noise by zeta
    if self.training: predicted_noise *= self.hyperparams.zeta
    else: predicted_noise *= self.hyperparams.sample_zeta

    return predicted_noise
```