Breaking the Limits of Diffusion Models via Continuous Dynamical Systems

Sergio Alfonso Calvo Ordoñez

Supervisor: Dr A.I. Aviles-Rivero
Prof C.B. Schönlieb

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy

Selwyn College August 2023

I dedicate this thesis to my parents and siblings. This is our work.

Declaration

I, Sergio Alfonso Calvo Ordoñez of Selwyn College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose. This dissertation contains approximately 14,956 words excluding declarations and bibliography, but including tables, footnotes, figure captions and appendices.

Experiments performed throughout this thesis were implemented from scratch using Python v3.8 and standard libraries (i.e. *pandas, NumPy, PyTorch, etc*). The software used for the diffusion mechanisms in this project originates from the *denoising_diffusion_pytorch* package located in the following repository: https://github.com/lucidrains/denoising-diffusion-pytorch and is modified to fit our needs. As for the differential equations solver, we used the *torchdiffeq* package (Chen et al., 2018b). Our code has been made publically available at https://github.com/Sergio20f/cDDPM.

Sergio Alfonso Calvo Ordoñez
August 2023

Acknowledgements

This work marks the end of my most challenging academic year. This, however, came together with many new friendships, hobbies, and experiences. Throughout these few paragraphs, my intention is to show a spark of my gratitude to some of the people who helped me go through it and succeed by the end.

Firstly, the team that helped me put forward this project. Dr Aviles-Rivero, Jiahao, and Lipei, thanks for believing in me and the idea. For your constant support in the form of supervisions and email replies at late hours of the night. This work would not have been possible without your expertise and advice.

I cannot look back to this year without highlighting the people that erased the term *loneliness* from my dictionary. Pau and Martin, I am glad our journeys remained together after QM. Cambridge would have not been the same without you - we made it. Dan and Tomas, thanks for keeping my motivation constantly up and always being real with me. Richard, Emma, and Connall, you managed to turn the biggest struggles into unforgettable memories, thanks for being great friends. Lastly the MLMI cohort, I feel privileged to have been a part of it. Not only brilliant students but also great people.

Lastly, the easiest thanks I will ever give. Thanks to my family - my parents and siblings - for raising me the way I am, with my values and the strength that took to complete this MPhil. Thanks for your encouragement, and support, and for always being understanding.
I love you.

Abstract

Diffusion probabilistic models (DPMs) stand as a critical tool in generative modelling, enabling the generation of complex data distributions. The concept underlying these models draws inspiration from non-equilibrium statistical physics. It involves gradually destroying the structure of a data distribution using a sequential forward mechanism to then learn a reverse diffusion process that aims to reconstruct the original data structure. They have the capacity to encapsulate stochastic processes within complex systems by accommodating the inherent variability observed in these systems and achieving a representative analytical framework. This family of deep generative models yields record-breaking performance in image synthesis, video generation, and molecule design. Despite their capabilities, their efficiency, especially in the reverse denoising process, remains a concern due to slow convergence rates, high computational costs, and long training and inference times.

In response to these challenges, in this dissertation, we introduce a new approach building upon the work in Cheng et al. (2023). By leveraging continuous dynamical systems, we present a denoising network for diffusion models that is more parameter-efficient, exhibits faster convergence, and demonstrates increased noise robustness. Experimenting with denoising probabilistic diffusion models (DDPMs), our framework operates with approximately a quarter of the parameters and $\sim$30% of the Floating Point Operations (FLOPs) compared to standard U-Nets in DDPMs. Furthermore, our model is up to 70% faster in inference than the baseline models when measured in equal conditions, while converging to better quality solutions, marking a notable acceleration and improvement of the reverse diffusion process. Finally, we argue that our method is compatible with existing performance enhancement techniques, enabling further improvements in DPM efficiency, quality, and speed.

Table of contents

List of figures **xiii**

List of tables **xvii**

1 Introduction **1**

1.1 Contributions . . . 3

1.2 Outline . . . 4

2 Background **5**

2.1 Diffusion Probabilistic Models . . . 5

2.1.1 Denoising Diffusion Probabilistic Models . . . 6

2.2 Neural Differential Equations . . . 9

2.2.1 Actually, what are neural differential equations? . . . 9

2.2.2 Emergence in deep learning . . . 9

2.2.3 Neural Ordinary Differential Equations . . . 10

2.2.4 Augmentations in Neural ODEs . . . 12

2.2.5 Connection with Diffusion Models . . . 13

2.3 U-Net and Continuous U-Net . . . 14

2.3.1 U-Net . . . 15

2.3.2 Continuous U-Net: Neural ODEs + U-Net . . . 16

2.3.3 Do U-Nets fit inside Diffusion Models? . . . 19

3 Methodology **21**

3.1 Overview . . . 21

3.2 Dynamic Blocks for Diffusion . . . 23

3.2.1 Foundational Blocks . . . 23

3.2.2 Mathematical Formulation . . . 25

3.3 Architecture Overview . . . 26

3.3.1 Baseline: U-Net . . . 26

3.3.2 Proposal: Modified Continuous U-Net 27
3.4 Residual Connections and Time Embeddings 28
3.5 Attention Mechanism 30
3.6 Training Procedure 31

4 Evaluation and Results 35
4.1 Experimental Setup 35
4.1.1 Dataset Description 36
4.1.2 Evaluation Protocol 37
4.1.3 Training Scheme 39
4.2 Image Synthesis 41
4.3 Denoising 44
4.4 Efficiency 51
4.5 Ablation Studies 53
4.6 Limitations 56

5 Conclusion 59
5.1 Future Work 60

References 61

Appendix A 65

Appendix B 69

List of figures

2.1 Graphical model of the diffusion process including the probabilities in the forward and reverse diffusion (Das, 2021). 6
2.2 Schematic showing the diffusion process (Weng, 2021). 7
2.3 Computation graph for a simple neural ODE (Kidger, 2022). 11
2.4 Representation of the probability flow ODE as a neural ODE. It enables the use of advanced ODE solvers and has deterministic encoding and generation processes. (Kreis et al., 2022) 14
2.5 Illustration of the U-Net architecture for a base resolution of 32x32 pixels. Blue boxes denote multi-channel feature maps, with the number of channels displayed above each box. The x-y dimensions are shown at each box's bottom left. Copied feature maps are in white. Arrows represent different operations performed. (Ronneberger et al., 2015) 16
2.6 Diagram comparing continuous U-Net and existing U-Net variants. The difference in the composition of the blocks is highlighted. While U-Net uses convolutional blocks, continuous U-Net models the dynamics through dynamical blocks. (Cheng et al., 2023) 17

3.1 We used a modified continuous U-Net architecture tailored for denoising in the reverse process of a diffusion model to help reconstruct the original data from a noise-corrupted version. 22
3.2 Schematic of our custom block for approximating the derivative in an ODE. This design integrates group normalization, convolutional operations, time embeddings via multi-layer perceptrons, and custom residual connections. . 24
3.3 Illustration of our custom ODE block capturing continuous-time dynamics. The block processes outputs from the derivative function approximator and uses them to delineate the data's evolutionary trajectory. It is equipped to work with various ODE solvers. 25

3.4 Given queries Q, keys K, and values V, the attention mechanism computes a weighted sum of values based on the similarity (dot-product) between the queries and keys. The similarity scores are scaled by a factor and then passed through a softmax to produce the attention weights (left). Multi-head attention (right) computes attention multiple times in parallel (Vaswani et al., 2023). 30

4.1 Sample images from the MNIST dataset: Grayscale representations of hand-written digits ranging from 0 to 9. Each image is of size 28x28 pixels. . . . 37
4.2 Selected images from the CelebA dataset: Portraits of celebrities showcasing diversity in facial attributes such as age, ethnicity, and features. Images are resized to 64x64 pixels for our experiments. 37
4.3 Example images from the LSUN Church dataset: Various churches depicting architectural intricacies and designs. Images are standardized to 64x64 pixels for analysis. 38
4.4 Image borrowed from Zhang et al. (2018) showcasing the effect of perceptual versus distortion metrics. Traditional metrics (L2/PSNR, SSIM) consistently diverge from human evaluations. In contrast, deep network models, irrespective of their architecture or type of supervision, yield embeddings that align remarkably well with human judgments. 39
4.5 Comparison of FID values as a function of sampling timesteps for both our proposed diffusion model and the U-Net-based model. The graph demonstrates the efficiency of our model in achieving optimal FID scores with fewer timesteps. 42
4.6 Randomly selected generated samples from our baseline diffusion model. . 43
4.7 Randomly selected generated samples from our continuous U-Net-based model. 43
4.8 Randomly selected generated samples from our baseline diffusion model. . 44
4.9 Randomly selected generated samples from our continuous U-Net-based model. 44
4.10 Randomly selected generated samples from our baseline diffusion model. . 45
4.11 Randomly selected generated samples from our continuous U-Net-based model. 45

4.12 Visualization of noise accumulation in images over increasing timesteps. As timesteps advance, the images exhibit higher levels of noise, showcasing the correlation between timesteps and noise intensity. The progression highlights the effectiveness of time embeddings in predicting noise magnitude at specific stages of the diffusion process. 46

4.13 SSIM scores plotted against diffusion steps for varying noise levels for one image. The graph underscores the consistently superior performance of U-Net over cU-Net in terms of SSIM, particularly at high noise levels. This dominance in SSIM may be misleading due to the inherent Distortion-Perception tradeoff and the tendency of our model to predict features instead of distorting the images. 49

4.14 LPIPS score versus diffusion - or denoising timesteps for one image. We can observe how our continuous U-Net consistently achieves better LPIPS score and does not suffer from such a significant *elbow effect* observed in the U-Net model in which the quality of the predictions starts deteriorating after the model achieves peak performance. The discontinuous lines indicate the timestep at which the peak LPIPS was achieved. Here, we see that, especially for a large number of timesteps, continuous U-Net seems to converge to a better solution in fewer steps. This is because of the ability it has of predicting facial features rather than simply settling for over-smoothed results (Figure A.3). 50

4.15 Bar chart illustrating the total number of parameters in different variations of U-Net and Continuous U-Net (cU-Net) models. The models include the original U-Net, the cU-Net, and three variations of the cU-Net with different components removed: attention, resblocks, and both attention and resblocks. Note that the y-axis is on the 10^7 scale. 52

A.1 Tracking intermediate model denoising predictions. The images on the left are the outputs of our continuous U-Net, and the ones on the right are from the conventional U-Net. 65

A.2 Tracking intermediate model denoising predictions: The images on the left depict the outputs of our continuous U-Net, which successfully removes noise in fewer steps compared to its counterpart (middle images) and maintains more fine-grained detail. The images on the right represent outputs from the conventional U-Net. 66

A.3 Tracking intermediate model denoising predictions: The images on the left depict the outputs of our continuous U-Net, which attempts to predict the facial features amidst the noise. In contrast, the images on the right, from the conventional U-Net, struggle to recognise the face, showcasing its limitations in detailed feature reconstruction. 67
A.4 Original image (left), original image after adding Gaussian noise (second), and denoised image after conditioning our continuous U-Net model on the noisy image (third and fourth). We can easily appreciate how our continuous U-Net is able to recover the details of the image. As we increase the noise, U-Net is unable to recover the glasses. 68

B.1 Representative samples from the version of our model trained without attention mechanism. The decrease in quality can often be appreciated in the images generated. 69
B.2 Representative samples from the version of our model trained without residual connections within our ODE block. We can see artifacts and inconsistencies frequently. 69
B.3 Representative samples from the basic version of our model which only includes time embeddings. As one can appreciate, sample quality suffers considerably. 69

List of tables

4.1 Performance metrics across datasets: FID scores, sampling timesteps (Steps), and average generation time for both the U-Net and continuous U-Net (cU-Net) models. The table highlights the efficiency and effectiveness of the proposed cU-Net in image generation. 42
4.2 Comparative average denoising performance between U-Net (left values) and cU-Net (right values) for different noise levels over the test dataset. While U-Net predominantly achieves higher SSIM scores, cU-Net often outperforms in LPIPS evaluations, indicating differences in the nature of their denoising approaches. 47
4.3 Comparison of average performance for U-Net (left) and cU-Net (right) at different noise levels in terms of the specific timestep at which peak performance was attained and time taken. These results are average across all the samples in our test set. 51
4.4 Comparative average performance of various denoising methods at select noise levels across the test set. Results demonstrate the capability of diffusion-based models (Diff U-Net and Diff cU-Net) in handling a broad spectrum of noise levels without retraining. 51
4.5 Number of GigaFLOPS (GFLOPS) and Megabytes in Memory (MB) for Different Models. 53
4.6 Ablation study examining the impact of various components on model performance. The table provides FID scores, optimal sampling steps, and average generation time over 100 samples for each model configuration. There is a moderate impact of the attention mechanism and the pivotal role of residual connections in the generation quality. 54
4.7 Comparison of average denoising performance between U-Net and cU-Net variants at different noise levels across the test set. 55

4.8 Average denoising performance of the cUNet, cUNet without attention, and cUNet without residual connections based on SSIM metric in the test dataset. 55
4.9 Average denoising performance of the cUNet, cUNet without attention, and cUNet without residual connections based on the LPIPS metric in the test dataset. 56

Chapter 1

Introduction

Diffusion Probabilistic Models (DPMs), grounded in the work of Sohl-Dickstein et al. (2015) and expanded upon by Song and Ermon (2020), Ho et al. (2020), and Song et al. (2021), have achieved remarkable results in various domains, including image generation (Dhariwal and Nichol, 2021; Nichol and Dhariwal, 2021; Ramesh et al., 2022; Saharia et al., 2022; Rombach et al., 2022), audio synthesis (Kong et al., 2021; Liu et al., 2022), and video generation (Ho et al., 2022; Ho et al., 2021). These score-based generative models utilise an iterative sampling mechanism to progressively denoise random initial vectors, offering a controllable trade-off between computational cost and sample quality.

Although the iterative process offers a method to balance quality with computational expense, it often leans towards the latter for state-of-the-art results. Producing samples of competitive quality generally demands a substantial number of iterations. For every generated sample, rather than executing a single forward pass, as in many alternative generative models, a diffusion model undergoes multiple forward passes - one for each denoising iteration (Sohl-Dickstein et al., 2015; Song and Ermon, 2020). When contrasted with single-step generative models such as Generative Adversarial Networks (Goodfellow et al., 2014), Variational Autoencoders (Kingma and Welling, 2022; Rezende et al., 2014), and normalizing flows (Rezende and Mohamed, 2015; Kingma and Dhariwal, 2018), diffusion models demand significantly higher computational resources. Specifically, their sampling process typically requires between 10 and 2000 times more compute for sample generation as compared to the aforementioned models (Song and Ermon, 2020; Ho et al., 2020; Song et al., 2021). This heightened computational demand results in slower inference speeds.

Recent research has delved into strategies to enhance the efficiency and speed of this reverse process. A notable advancement is the Early-Stopped Denoising Diffusion Probabilistic Models (ES-DDPM) proposed by Lyu et al. (2022). The main idea is to stop the diffusion process early. Instead of diffusing the data distribution into a Gaussian distribution via hundreds to thousands of iterative steps, ES-DDPM considers only the initial few diffusion steps so that the reverse denoising process starts from a non-Gaussian distribution. Another significant contribution is the Analytic-DPM framework (Bao et al., 2022). This training-free inference framework estimates the analytic forms of variance and Kullback-Leibler divergence using Monte Carlo methods in conjunction with a pre-trained score-based model. The results from this paper show improvements not only in the log-likelihood of various DPMs but authors also achieve a speed-up ranging from 20x to 80x, marking a substantial leap in efficiency. Furthermore, an intriguing approach was studied in Chung et al. (2022), here, authors incorporate manifold constraints to improve diffusion models for inverse problems. By introducing an additional correction term inspired by manifold constraints, this method achieves a significant performance boost, highlighting the potential of geometric considerations in refining diffusion models.

Other lines of work have focused on modifying the sampling process during inference while keeping the model unchanged. Song et al. (2022) proposed Denoising Diffusion Implicit Models (DDIM) where the reverse Markov chain is altered to take deterministic "jumping" steps composed of multiple standard steps. This reduces the number of required steps but may introduce discrepancies from the original diffusion process. Nichol and Dhariwal (2021) proposed timestep respacing to non-uniformly select timesteps in the reverse process. While reducing steps, this can cause deviation from the model's training distribution. In general, these methods provide inference-time improvements but do not accelerate model training.

A different approach trains diffusion models with continuous timesteps and noise levels to enable variable numbers of reverse steps after training (Song and Ermon, 2020). However, models trained directly on continuous timesteps often underperform compared to discretely-trained models (Song et al., 2021), and training must be repeated for each desired step count. Kong et al. (2021) approximate continuous noise levels through interpolation of discrete timesteps, but lacks theoretical grounding. Orthogonal strategies accelerate diffusion models by incorporating conditional information. Preechakul et al. (2022) inject an encoder vector to guide the reverse process. While effective for conditional tasks, it provides limited improvements for unconditional generation. Salimans and Ho (2022) distil a teacher model into students taking successively fewer steps, reducing steps without retraining, but distillation cost scales with teacher steps.

Throughout this thesis, we construct and evaluate an approach that rethinks the reverse process in diffusion models by fundamentally altering the denoising network architecture. Current literature predominantly employs U-Net architectures for discrete denoising of diffused inputs over a specified number of steps. Many reverse process limitations stem directly from constraints inherent to the chosen denoising network. Building on the work in Cheng et al. (2023), we leverage continuous dynamical systems to design a novel denoising network that is parameter-efficient, exhibits faster convergence - and to better solutions, demonstrates robustness against noise, and outperforms conventional U-Nets while providing theoretical underpinnings. We show that our architectural shift directly enhances the reverse process of diffusion models by offering comparable performance in image synthesis but an improvement in inference time in the reverse process, denoising performance, and operational efficiency. Importantly, our method is orthogonal to existing performance enhancement techniques, allowing their integration for further improvements in DPPM efficiency, quality, and speed.

1.1 Contributions

The specific contributions of this thesis are as follows:

1. We provide a systematic overview of the literature on Denoising Diffusion Probabilistic Models (DDPMs), Neural Differential Equations, and U-Net architectures. Our review highlights the mathematical relationships among these concepts and integrates insights from different sources to establish a unified understanding of their interconnection.

2. Our work builds upon Cheng et al.'s foundational concepts by adapting their dynamic blocks to contemporary methodologies. We are the first to investigate the adaptability of continuous U-Nets by examining its scalability, and compatibility with modern deep learning elements such as attention mechanisms and residual connections, further extending its capabilities. Additionally, our contribution expands upon the foundation established by Cheng et al. (2023), expanding the framework to handle denoising as a new downstream task.

3. We present a new Neural ODE block that seamlessly incorporates residual connections and time embeddings into the existing continuous U-Net architecture. This is achieved by explicitly modifying the block structure, resulting in the ability to adapt to the temporal variations in terms of the diffusion steps.

4. We propose a novel family of diffusion models that employ deep implicit layers using Neural ODEs in the reverse process. This design is characterized by improved computational efficiency, enhanced robustness, and the reduction of data-inherent noise.

5. We provide an extensive evaluation of our framework to show that it achieves comparable performance in image synthesis, and perceptually outperforms the baseline in

denoising with approximately 4x fewer parameters, smaller memory footprint, and shorter inference times.

Overall, this work presents developments in deep implicit layers for reverse diffusion processes, with innovations including the integration of attention mechanisms, residual connections, and time embeddings, coupled with empirical demonstrations of parameter efficiency and competitive performance.

1.2 Outline

Chapter 2: This chapter explores the relevant theoretical underpinnings. It starts with an examination of diffusion probabilistic models. Then, we move on to an analysis of neural differential equations, detailing their definitions and their natural rise in deep learning. We emphasise neural ordinary differential equations and their augmentations due to their importance in our approach. We also describe the relationship between diffusion models and these neural differential equations. Lastly, this discussion leads to U-Nets and continuous U-Nets which represent the main topic of discussion further on in the thesis and where the success of our framework lies.

Chapter 3: Chapter 3 explains our methodology. We introduce our framework and how it differs from standard DDPMs explaining why our approach is more suited for diffusion models. We then present our modifications to existing continuous U-Net architectures, including residual connections, time embeddings, and attention mechanisms, emphasising their role in denoising. Finally, the chapter ends with a detailed explanation of our training procedure.

Chapter 4: This chapter showcases our experimental results, both quantitative and qualitative. We demonstrate the viability of our approach and its potential to substantially improve diffusion models in several relevant metrics. In addition, we explore the limitations of our framework.

Chapter 5: The thesis concludes with a comprehensive summary of our findings, emphasizing their relevance in the broader context. We also explore future research trajectories, including further opportunities for computational optimization, and alternative methodologies for quantifying uncertainty.

Chapter 2

Background

This chapter provides an overview of the theoretical foundations of our approach, combining the strengths of continuous dynamical systems, U-Net architectures, and diffusion models. Section 2.1 conducts an in-depth analysis of the mathematical structure and principles of Diffusion Probabilistic Models (DPMs) (Sohl-Dickstein et al., 2015), before introducing Denoising Diffusion Probabilistic Models (DDPMs) (Ho et al., 2020), a primary framework within diffusion models. Section shifts focus to Neural Differential Equations, examining their advantages, limitations, and natural connection to diffusion models. Lastly, section 2.3 explores U-Nets (Ronneberger et al., 2015) and continuous U-Nets (Cheng et al., 2023), a vital component of our proposed method. We provide their definitions, and mathematical basis, and present a strong motivation for the development of our framework.

2.1 Diffusion Probabilistic Models

Diffusion Probabilistic Models (DPMs) (Sohl-Dickstein et al., 2015) constitute a family of generative models that leverage the theory of non-equilibrium thermodynamics in order to build a probabilistic bridge between structured data and random noise. Inspired by the diffusion processes in statistical physics and Markov chain Monte Carlo (MCMC) methods, these models undertake a systematic and incremental perturbation of the data distribution gradually destroying its inherent structure, and then learning to reverse this process, thereby generating a flexible and tractable representation of the data.

The foundation of DPMs stems from the observation that while transforming structured data into noise is relatively straightforward, the converse - generating structured data from noise - poses a significant challenge. This dichotomy led to the development of a two-pronged approach in DPMs encompassing both a forward diffusion process, which stochastically destructs the data and a learned reverse diffusion process that aims to restore the structure of the sample (Figure 2.1).

Consider data $\mathbf{x}_0$, which we aim to destroy. We employ a forward diffusion process, a Markov chain of T diffusion steps driven by a stochastic encoder $q(x_t|x_{t-1})$. This process is iteratively applied, transforming the data into progressively noisier versions until reaching x_T, the state where the data conforms to a standard Gaussian distribution $\mathcal{N}(0, \mathbf{I})$ (or any other convenient reference distribution). Namely, this process systematically adds random noise, gradually converting the structured data distribution into an unstructured one. On the other hand, the reverse process employs a decoder $p_\theta(x_{t-1}|x_t)$ that is learned from the data and operates iteratively for T steps, starting from the noise state x_T and working its way back to the original state of the data x_0. Importantly, unlike Variational Autoencoders (VAEs) (Kingma and Welling, 2022) or flow models (Rezende and Mohamed, 2015), DPMs employ a fixed procedure, and the latent variable maintains a high dimensionality.

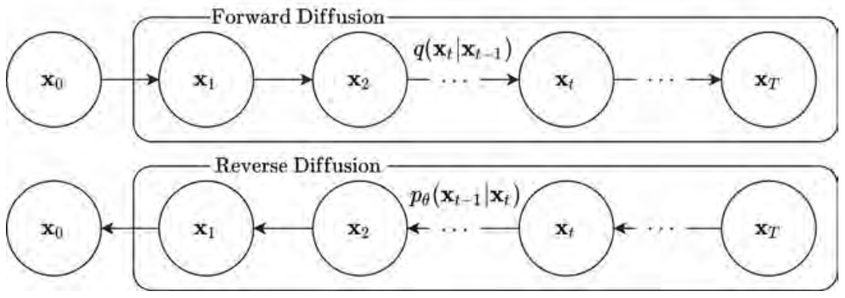


Fig. 2.1 Graphical model of the diffusion process including the probabilities in the forward and reverse diffusion (Das, 2021).

Fundamentally, each step in the Markov chain represents a small perturbation in the diffusion process, making the learning more tractable than attempting to describe the full distribution with a single, non-analytically-normalizable potential function.

2.1.1 Denoising Diffusion Probabilistic Models

Denoising Diffusion Probabilistic Models (DDPMs) (Ho et al., 2020) extend the framework of DPMs through the inclusion of a denoising mechanism (Figure 2.2). The latter is used as an inverse mechanism to reconstruct data from a latent noise space achieved through a stochastic process (reverse diffusion). This relationship emerges from Song et al. (2021), which shows that a certain parameterization of diffusion models reveals an equivalence with denoising score matching over multiple noise levels during training and with annealed Langevin dynamics during sampling.

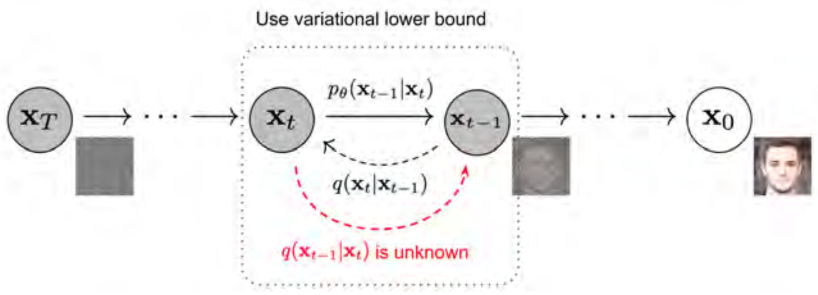

Use variational lower bound
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_T$
$\mathbf{x}_t$
$\mathbf{x}_{t-1}$
$\mathbf{x}_0$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is unknown

Fig. 2.2 Schematic showing the diffusion process (Weng, 2021).

DDPMs can be thought of as analogue models to hierarchical VAEs (Cheng et al., 2020), with the main difference being that all latent states, x_t for $t = [1, T]$, have the same dimensionality as the input x_0. This detail makes them also similar to normalizing flows (Rezende and Mohamed, 2015), however, diffusion models have hidden layers which are stochastic and do not need to use invertible transformations.

As any other diffusion model, DDPMs use two Markov chains: a forward chain that perturbs the training data into noise, and a reverse chain that attempts to carry out the inverse transformation. Focusing on the forward diffusion - or encoder, this is defined to be a simple linear Gaussian model. Then, given a sample from a real distribution $\mathbf{x}_0 \sim q(\mathbf{x})$, we can define the forward diffusion process as:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), \tag{2.1}$$

where the β_t terms act as a variance scheduler and follow $\{\beta_t \in (0,1)\}_{t=1}^{T}$. Because of the Markov properties, we also know that the joint distribution over all the latent states conditioned on the input is:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}). \tag{2.2}$$

From this definition, we can see how in the limit $T \to \infty$, the distribution of $\mathbf{x}_t$ becomes an isotropic Gaussian distribution as $\mathbf{x}_0$ gradually loses its distinguishable features. Since this is the definition of a linear Gaussian stochastic process, we can compute its marginals at any time step t in closed form by using the reparametrization trick. More concretely, letting $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}), \tag{2.3}$$

as:

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\varepsilon_{t-1} = \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\varepsilon_{t-2} = \cdots = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon. \tag{2.4}$$

This distribution, $q(\mathbf{x}_t|\mathbf{x}_0)$, is also known as the diffusion kernel. This is because applying it to the input data distribution and then computing the result unconditional marginals, is equivalent to a Gaussian convolution (Murphy, 2023):

$$q(\mathbf{x}_t) = \int q_0(\mathbf{x}_0)q(\mathbf{x}_t|\mathbf{x}_0)d\mathbf{x}_0 \tag{2.5}$$

Therefore, the marginals progressively simplify as t increases. When applied in the context of images, this process initially erases finer details such as texture, classified as high-frequency content. As it continues, it eventually starts to remove the broader, low-frequency content which often carries substantial semantic information like the shape.

Having completed the forward diffusion process, we are now interested in sampling from $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ in order to achieve a sample from the true distribution starting from simply Gaussian noise. Knowing the input $\mathbf{x}_0$, one can derive reverse steps using Bayes' rule and the fact that the reverse conditional probability is tractable when conditioned on the input sample:

$$q(\mathbf{x_{t-1}}|\mathbf{x_t},\mathbf{x_0}) = \mathcal{N}(\mathbf{x}_{t-1};\tilde{\mu}_t(\mathbf{x}_t,\mathbf{x}_0),\tilde{\beta}_t\mathbf{I}), \tag{2.6}$$

where,

$$\tilde{\mu}_t(\mathbf{x}_t,\mathbf{x}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t, \quad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \tag{2.7}$$

A couple of things to note are that, firstly, we work under the assumption that β_t is small enough so that $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ will also be Gaussian. Secondly, the assumption that we know $\mathbf{x}_0$ is unrealistic, however, in practice we can train a generative model p_θ, to approximate the above distribution averaged over the input. Therefore, the generator should be chosen to have the form:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1};\mu_\theta(\mathbf{x_t},\mathbf{t}),\Sigma_\theta(\mathbf{x_t},\mathbf{t})), \tag{2.8}$$

the corresponding joint distribution over all generated variables is given by:

$$p_\theta(\mathbf{x_{0:T}}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \tag{2.9}$$

2.2 Neural Differential Equations

Neural Differential Equations are a recent and powerful addition to the deep learning toolset, offering a continuous-time approach to data modelling (Chen et al., 2019). They are unique in their ability to model complex systems over time while efficiently handling memory and computation (Rubanova et al., 2019). In this section, we will delve into their workings, their emergence in deep learning, and their connections with other popular architectures (He et al., 2016). We will also focus on a particular type, Neural Ordinary Differential Equations, discussing their characteristics, implementation, and potential applications (Kidger et al., 2020).

2.2.1 Actually, what are neural differential equations?

In short, a neural differential equation is a differential equation characterised by using a neural network to specify the parameters of the vector field that represent the system's dynamics. One such representative instance is a neural partial differential equation:

$$u(0,x) = u_0(x) \qquad \frac{\partial u}{\partial t}(t,x) = f_\theta(t,x,u(t,x)). \tag{2.10}$$

In the above formulation, θ denotes the set of trained parameters, f_θ is a function parameterised by a neural network and $u : [0,T] \times \mathbb{R}^{d_1\times\cdots\times d_k} \to \mathbb{R}$ is the solution of the equation.

In essence, the differential equation defines a transformation from one state to another over continuous time or space, and the neural network is used to learn the form of this transformation. The principal idea of how this method works is that a differential equation solver is used as part of a learnt differentiable computation graph. This means that the solver is integrated into the backpropagation process, allowing the gradients of the loss function to be propagated through the solver to the parameters of the neural network that defines the dynamics of the differential equation.

2.2.2 Emergence in deep learning

It is possible to naturally arrive at neural differential equations by studying modern deep learning (Kidger, 2022). There is a close relation to many different architectures, for instance, the formulation of a residual neural network is:

$$y_{l+1} = f_\theta(j, y_l) + y_l, \tag{2.11}$$

where $f_\theta(j, \cdot)$ is the l-th residual block parametrised by θ.

On the other hand, the definition of neural ODE is Equation (2.13) which, discretised using the Euler method at times t_l uniformly separated by Δt, yields:

$$\frac{y(t_{l+1} - y(t_l)}{\Delta t} \approx \frac{dy}{dt}(t_l) = f_\theta(t_l, t(t_l)),$$

and rearranging,

$$y(t_{l+1} = y(t_l) + \Delta t f_\theta(t_l, t(t_l)). \tag{2.12}$$

Taking into account that we can absorb Δt into f_θ, we realise that we recover Equation (2.11). Therefore, neural ODEs are the continuous limit of residual networks.

It is interesting to note that we see similar relationships with many modern deep learning architectures. StyleGAN2 (Karras et al., 2020) and score-based diffusion models, for example, can be regarded as discretised SDEs, or as we will see in a later section, diffusion models in the limit of an infinite number of hidden layers are a special case of a neural ordinary differential equation.

2.2.3 Neural Ordinary Differential Equations

In the landscape of generative models, the Neural Ordinary Differential Equation (neural ODE) (Chen et al., 2019) is a prime example of a neural differential equation. It is denoted as:

$$y(0) = y_0, \qquad \frac{dy}{dt}(t) = f_\theta(t, y(t)), \tag{2.13}$$

where $y_0 \in \mathbb{R}^{d_1 \times \cdots \times d_k}$ refers to an input tensor with any dimensions, θ symbolizes a learned parameter vector, and $f_\theta : \mathbb{R} \times \mathbb{R}^{d_1 \times \cdots \times d_k} \to \mathbb{R}^{d_1 \times \cdots \times d_k}$ is a neural network function. Typically, f_θ is parameterized by simple neural architectures, including feedforward or convolutional networks. The selection of the architecture depends on the nature of the data and is subject to efficient training methods, such as the adjoint sensitivity method for backpropagation through the ODE solver. The idea can be summarised pictorially in Figure 2.3.

Understanding the existence and uniqueness of solutions to (2.13) is crucial. Firstly, from the perspective of machine learning, the existence of a solution guarantees that the model is

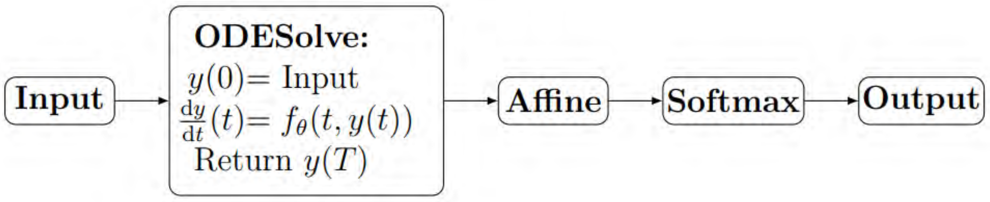


Fig. 2.3 Computation graph for a simple neural ODE (Kidger, 2022).

capable of producing outputs for given inputs, i.e. it can effectively function as a predictive tool. Uniqueness is critical for maintaining the deterministic nature of the model. It is this deterministic nature that facilitates the optimisation process during training, making it predictable and replicable.

As f_θ is parametrised by a neural network, it will usually be a Lipschitz function. This is because the functions represented by neural networks are typically composed of layers each with a linear transformation (which is Lipschitz) followed by an activation function (which is also usually Lipschitz). The Lipschitz property is preserved under function composition and combination, which means that if you stack multiple Lipschitz layers on top of each other, the overall function represented by the network will also be Lipschitz. Why is this important? Well, in the context of neural ODEs, we often need our neural networks to be Lipschitz in order to ensure the existence and uniqueness of solutions to the ODEs they represent, if so, theorem 2.2.1 applies.

Theorem

Theorem 2.2.1 (Picard's Existence Theorem). Let $f : [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$ be continuous in t and uniformly Lipschitz in y. Let $y_0 \in \mathbb{R}^d$. Then there exists a unique differentiable $y : [0,T] \to \mathbb{R}^d$ satisfying 2.13.

Note that uniformly Lipschitz in y and continuous in t means that there exists a constant $C > 0$ such that for all t, y_1, y_2 then $||f_\theta(t, y_1) - f_\theta(t, y_2)|| \leq C||y_1 - y_2||$.

There are a couple of factors to consider when thinking about the evaluation and training of models based on differential equations. One must be able to solve these differential equations numerically and then, also be able to backpropagate through the differential equation to obtain gradients for its parameters θ (Kidger, 2022).

During the training phase, backpropagation through the ODE solver, also referred to as the adjoint sensitivity method, is used. Instead of directly computing gradients of the loss function with respect to the neural network parameters, this method solves an auxiliary reverse-time ODE for the adjoint state, a variable that assists in calculating the gradients. This allows efficient computation of gradients regardless of the number of evaluation points in the ODE solver, making it memory efficient compared to standard backpropagation through time. In the evaluation, given an initial condition, the model employs an ODE solver to produce output at a specified time.

2.2.4 Augmentations in Neural ODEs

Augmentation is a technique that inserts an affine map between the input and initial value, adding additional dimensions to the hidden state before it's processed by the neural ODE. Given some input $x \in \mathbb{R}^d$, the initial value of the ODE is taken to be a mapping of the input through some function $g_\theta(x)$ for some learnt $g_\theta : \mathbb{R}^d \to \mathbb{R}^{d_l}$ to a higher dimensional space $d_l > d$, instead of $y(0) = x$ (Kidger, 2022). The framework therefore becomes:

$$y(0) = g_\theta(x), \qquad \frac{dy}{dt}(t) = f_\theta(t, y(t)), \tag{2.14}$$

There are many popular choices for augmentations. Some examples are:

- **Zero augmentation:** This is the simplest form of augmentation. The original inputs are concatenated with zeros, thus expanding the dimensionality of the inputs without adding any newly learned information. More specifically, $g_\theta(x) = [x, 0]$.
- **Learned augmentation** (Dupont et al. 2019) **:** Here, $g_\theta(x) = [x, g_e\theta(x)]$, where $g_e\theta(x)$ is a learned function of the input x. Unlike zero augmentation, this method attempts to learn a function to map the inputs into a higher-dimensional space.
- **Affine map:** In this technique $g_\theta(x)$ is an affine map, i.e. a function composed of a linear transformation and a translation. This approach allows the model to learn a linear transformation of the inputs that might better expose the data's underlying structure to the neural ODE.

Because these augmentations introduce additional dimensions that can capture complex dependencies in the data that are not strictly Markovian, lifting into a higher-dimensional space may be equivalent to relaxing the Markov property. What this means is that for $s < t$ the output $l_\theta(y(s))$ does not completely determine $l_\theta(y(t))$ even though $y(s)$ does determine $y(t)$.

In the framework we will present in an upcoming section, we heavily make use of the second-order-augmentation. Introduced in Norcliffe et al. (2020), authors set $d_l = 2d$ and set up the corresponding vector field in a way so that the augmented dimensions correspond to velocities. Expressed as a second-order neural ODE:

$$\frac{d^2s}{dt}(t) = f_\theta(t, s(t), \frac{ds}{dt}(t)) \tag{2.15}$$

2.2.5 Connection with Diffusion Models

Neural Differential Equations and Diffusion Models share the fundamental concept of modelling the evolution of systems in continuous time. While Neural Differential Equations focus on deterministic state transitions, Diffusion Models consider probabilistic data distribution transitions.

If we consider a DDPM model in the limit of an infinite number of hidden layers, we are required to start working in a continuous time setting. This has the advantage to allow us to leverage the large existing literature on solvers for ordinary differential equations to enable fast generation. Firstly, consider a diffusion process where the noise level gets rewritten as $\beta_t \Delta t$, where Δt is the step size:

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\mathcal{N}(\mathbf{0},\mathbf{I}) = \sqrt{1-\beta_t\Delta t}x_{t-1} + \sqrt{\beta_t\Delta t}\mathcal{N}(\mathbf{0},\mathbf{I}). \tag{2.16}$$

If the step size is small enough, one could use a Taylor series expansion and get:

$$x_t \approx x_{t-1} - \frac{\beta(t)\Delta t}{2}x_{t-1} + \sqrt{\beta(t)\Delta t}\mathcal{N}(\mathbf{0},\mathbf{I}) \Rightarrow \frac{x_t - x_{t-1}}{\Delta t} \approx -\frac{\beta(t)}{2}x_{t-1} + \frac{\sqrt{\beta(t)}}{\sqrt{\Delta t}}\mathcal{N}(\mathbf{0},\mathbf{I}). \tag{2.17}$$

We can now switch to the continuous time limit and write it as an ODE. Say we sample the initial state and we let it evolve deterministically over time following an ODE:

$$dx = \underbrace{\left[f(x,t) - \frac{1}{2}g(t)^2\Delta_x \log p_t(x)\right]}_{h(x,t)}dt. \tag{2.18}$$

This ODE is known as the probability flow ODE (Boffi and Vanden-Eijnden, 2023) and it can be solved for any t through a solver:

$$x(t) = x(0) + \int_0^t h(x,\tau)d\tau \tag{2.19}$$

We can visualise the deterministic trajectories from this solution in Figure 2.4.

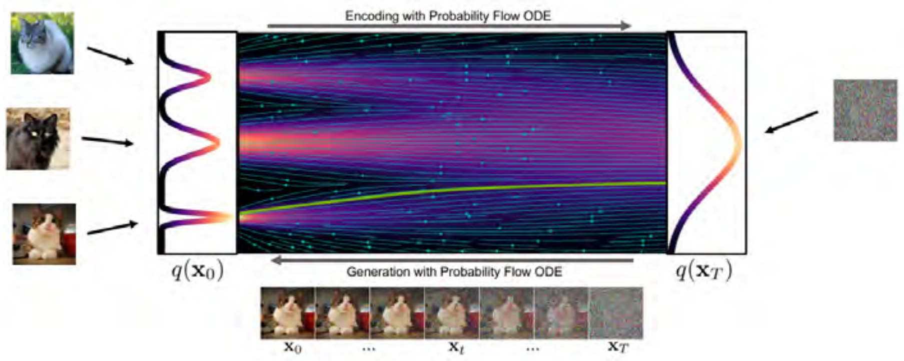


Fig. 2.4 Representation of the probability flow ODE as a neural ODE. It enables the use of advanced ODE solvers and has deterministic encoding and generation processes. (Kreis et al., 2022)

Skipping some steps, it is then possible to derive the probability flow ODE for the reverse diffusion to get:

$$dx_t = \left[f(x,t) - \frac{1}{2} g(t)^2 s_\theta(x_t,t) \right] dt. \tag{2.20}$$

Setting $f(x,t) = -\frac{1}{2}\beta(t)$ and $g(t) = \sqrt{\beta(t)}$, just in like the formulation of DDPMs, we obtain:

$$dx_t = -\frac{1}{2}\beta(t)[x_t + s_\theta(x_t,t)]dt. \tag{2.21}$$

We can refer back to Figure 2.4 to get an illustration of this. This is solvable using, for instance, Euler's method:

$$x_{t-1} = x_t + \frac{1}{2}\beta(t)[x_t + s_\theta(x_t,t)]\Delta t. \tag{2.22}$$

We can see how this model is a special case of a neural ODE - a continuous normalising flow. For this reason, we can derive the exact log marginal likelihood, and then use score matching to fit our model.

2.3 U-Net and Continuous U-Net

In the specific case of image segmentation, U-Net, a type of Convolutional Neural Network (CNN) architecture, has seen widespread usage due to its performance and efficiency (Ronneberger et al., 2015). However, despite its impressive performance, it suffers from several limitations, including its discrete nature, the need to hard code the receptive field size, the absence of robust handling of inherent noise in images, and a lack of mathematical foundation that would facilitate a rigorous understanding and analysis of the architecture. These limitations have motivated the development of new models that address these shortcomings, and one such model is the recently introduced continuous U-Net (Cheng et al., 2023).

This section presents a detailed comparison and analysis of these two models. The first subsection introduces U-Net, its architecture, and its major limitations. The second subsection presents continuous U-Net, a recent advancement that addresses the limitations of U-Net by employing a continuous approach grounded in the theory of Neural ODEs. This is followed by detailed mathematical underpinnings supporting the continuous U-Net model, propositions, theorems, and proofs that facilitate a deeper understanding of the model's performance and the advantages it brings.

2.3.1 U-Net

Ronneberger et al. (2015) present U-Net, a network together with a training strategy that leans on data augmentation, maximising the use of available annotated samples. The architecture encompasses a contracting pathway for understanding context, paired with a symmetrically expanding pathway that facilitates exact localization. Their core idea was to enhance a conventional contracting network with additional layers, substituting pooling operators with upsampling operators, thereby improving the output's resolution. Precise localisation was achieved by fusing the high-resolution features from the contracting pathway with the upsampled output. One of the novelties of the architecture was including a substantial number of feature channels in the upsampling section to enable the network to transmit context information to layers with higher resolution. As a result, the expanding path is symmetrical to the contracting path, giving rise to a U-shaped architecture (Figure 2.5).

Despite the notable success of U-Net in biomedical image segmentation and denoising (Chen et al., 2018a), the model does come with some limitations. One such drawback for some applications is the discrete nature of U-Net's architecture. This manifests in, for example, the use of medical image data, which is continuous rather than discrete [1]. Also, having a discrete architecture means making predictions by discretising the solution layer by layer, leading to a very high computation cost.

[1]Medical image data is considered continuous because it captures a continuous physical property/phenomena, represented digitally via pixels in a discretised form. When we talk about an image being continuous, we mean that any small movement in the physical world corresponds to a potentially infinitesimal change in the image.

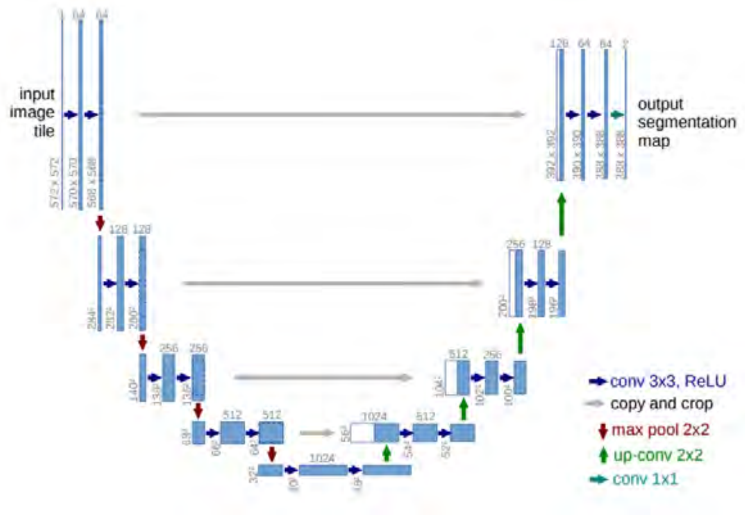


Fig. 2.5 Illustration of the U-Net architecture for a base resolution of 32x32 pixels. Blue boxes denote multi-channel feature maps, with the number of channels displayed above each box. The x-y dimensions are shown at each box's bottom left. Copied feature maps are in white. Arrows represent different operations performed. (Ronneberger et al., 2015)

As well, existing U-Net models do not account for inherent noise that affects predictions. In the context of medical imaging, noise can arise due to various factors like imaging hardware, patient movement, or inherent biological variability. This noise can significantly affect the quality of the image and, subsequently, the performance of the model in tasks like segmentation or anomaly detection. Lastly, U-Net's design lacks the mathematical underpinnings that would allow for a more rigorous analysis and understanding of its functioning. This can prove problematic when attempting to reason about its performance or when trying to improve upon its design in a systematic and principled manner.

2.3.2 Continuous U-Net: Neural ODEs + U-Net

The question addressed by Cheng et al. (2023) is: *can one design a U-type network that overcomes the aforementioned major issues of existing U-Net models?*. Authors propose **continuous U-Net**, a new network for medical image segmentation motivated by works in deep implicit learning and continuous approaches based on Neural ODEs (Chen et al., 2019; Dupont et al., 2019). This novel architecture consists of a continuous deep network whose dynamics are modelled by second-order ordinary differential equations. The idea is to transform the dynamics in the network - previously CNN blocks - into dynamic blocks (Figure 2.6) to get a solution.

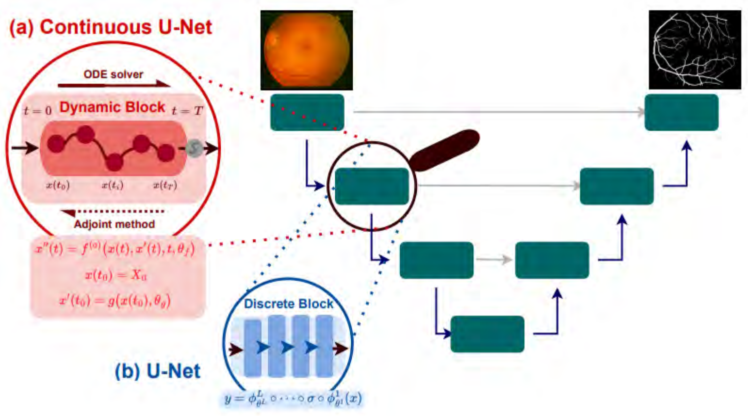

(a) Continuous U-Net
ODE solver
$t=0$ Dynamic Block $t=T$
$x(t_0)$ $x(t_i)$ $x(t_T)$
Adjoint method
$x''(t) = f^{(a)}(x(t), x'(t), t, \theta_f)$
$x(t_0) = X_0$
$x'(t_0) = g(x(t_0), \theta_g)$
Discrete Block
(b) U-Net
$y = \phi^L_{\theta^L} \circ \cdots \circ \sigma \circ \phi^1_{\theta^1}(x)$

Fig. 2.6 Diagram comparing continuous U-Net and existing U-Net variants. The difference in the composition of the blocks is highlighted. While U-Net uses convolutional blocks, continuous U-Net models the dynamics through dynamical blocks. (Cheng et al., 2023)

Continuous U-Net has desirable properties, these include faster convergence, more robustness, it is noiseless, and it comes with underpinning theory. Faster convergence is achieved due to fewer iterations and constant memory cost, stemming from modelling network dynamics in higher dimensions, which offers enhanced trajectory learning flexibility. Secondly, as will be remarked below, the dynamic blocks are proven to be reliably useful providing the model with greater robustness. Also, the original paper shows that continuous U-Net is always bounded by some range whilst CNNs are not. This, together with the fact that it is smoother than existing architectures, leads to better handling of the inherent noise in the data.

As mentioned before, continuous U-Net is modelled with dynamical blocks, hence it is formulated from a dynamical systems perspective. The CNN blocks are transformed into second-order ODE blocks defined as:

$$\begin{cases} x''(t) = f^{(a)}(x(t), x'(t), t, \theta_f) \\ x(t_0) = X_0, \quad x'(t_0) = g(x(t_0), \theta_g), \end{cases} \tag{2.23}$$

here, velocity is given by $f^{(a)}$, a neural network with parameters θ_f and initial position described by the samples in the dataset X_0.

Now, let us present and develop the mathematical underpinnings behind continuous U-Net that highlight the guarantees that make it attractive for our applications. To start, we will present a few propositions and theorems that will come in handy for future reference.

Proposition

Proposition 2.3.1. Any given high-order neural ODE can be transformed into a first-order neural ODE.

Proof. Consider a high-order neural ODE of order m. This can be generally represented as:

$$\frac{d^m x}{dt^m} = f(t, x, x^{(1)}, x^{(2)}, \dots, x^{(m-1)}), \tag{2.24}$$

where $x(t)$ is the state, $x^{(i)}$ represents the ith derivative of x, and f is a neural network function.

We want to transform this into a first-order Neural ODE. We do so by defining new variables $y_i = x^{(i-1)}$ for $i = 1, \dots, m$, with $y_1 = x$, $y_2 = x^{(1)}$, ..., $y_m = x^{(m-1)}$. Now, we can write down a system of first-order ODEs:

$$\frac{dy_1}{dt} = y_2, \quad \frac{dy_2}{dt} = y_3, \quad \dots \quad \frac{dy_{m-1}}{dt} = y_m, \quad \frac{dy_m}{dt} = f(t, y_1, y_2, \dots, y_m), \tag{2.25}$$

which is now a system of first-order ODEs. □

Theorem

Theorem 2.3.1. An m^{th}-order neural ODE with neural network f_θ is well-posed (in the sense of Hadamard) if f_θ is Lipschitz.

Proof. By invoking proposition 2.3.1, we can transform any given high-order ODE into a system of first-order ODEs. Let's define the input and output data: $X_k \mapsto Y_k$, $k \in K$, where K is a linearly-ordered finite subset of $\mathbb{N}$.

The solution to the system of first-order neural ODEs can be approached as an initial value problem (IVP):

$$y(S) = h_y(h_x(x) + \int_0^S f_\theta(\tau, x, z(\tau)) d\tau). \tag{2.26}$$

Given that f_θ is Lipschitz (because it is a neural network function), it follows that the solution to this IVP is well-posed in the sense of Hadamard. □

Theorem

Theorem 2.3.2. The second-order Neural ODE's dynamic blocks can be solved by using the first-order adjoint method.

Proof. The dynamic blocks are second-order neural ODEs. This is equivalent to setting $m = 2$ in proposition 2.3.1, implying: $\mathbf{z}'(t) = f^{(v)}(\mathbf{z}(t), t, \theta_f)$. □

We can see how proposition 2.3.1, allows for Theorem 2.3.1, which at the same time, leads to Theorem 2.3.2. Leveraging the ability to utilize the first-order adjoint method in solving dynamic blocks, continuous U-Net effectively addresses the receptive field size limitation seen in the original U-Net. This approach eliminates the need for storing layers sequentially, instead requiring just a single point to reconstruct the entire trajectory via forward and backward iterations. Consequently, continuous U-Net has constant memory cost, $\mathcal{O}(1)$.

In terms of the robustness and the noiseless characteristics of continuous U-Net we can provide an intuition on why these exist. From Yan et al. (2022), we know that the inherent properties of the integral curves of ODEs make them more robust than CNNs.

Theorem

Theorem 2.3.3. ODE integral curves do not intersect (Coddington and Levinson, 1955). If $z_1(t)$ and $z_2(t)$ are two ODE solutions with different initial values of the same function, then $z_1(t) \neq z_2(t)$ for all $t \in [0, \infty)$ (Cheng et al., 2023). The proof can be found in Coddington and Levinson (1955).

By virtue of Theorem 2.3.3, the solution to dynamic blocks in continuous U-Net is always bounded, in contrast to the unboundedness of traditional CNNs. As well, employing second-order ODEs provides increased smoothness, due to them being - at least - twice continuously differentiable and therefore less sensitive to noise. Unlike U-type networks that learn smooth homeomorphisms, continuous U-Net dynamic blocks, with their additional dimensions from the second-order design, avoid this constraint. These, together with the parameter efficiency of second-order dynamic blocks, with no need for parameters on $x_0'(t_0) = g(x(t_0)), \theta_g$, make second-order neural ODEs more robust and noiseless compared to first-order neural ODEs and other existing U-type architectures.

2.3.3 Do U-Nets fit inside Diffusion Models?

One could argue multiple reasons why U-Nets are an excellent design choice for a denoising network (Krull et al., 2019). For instance, at the core of U-Net's success in denoising is its capability to learn hierarchical features. The encoder progressively extracts high-level semantic information by reducing spatial dimensions and increasing the depth of the feature maps. The decoder on the other hand, gradually recovers the spatial information to reconstruct the image. This structure is especially beneficial in denoising tasks as it allows the model to identify patterns and structures at multiple scales. As well, the skip connections found in U-Nets allow the direct transfer of detailed spatial information from earlier layers to the later ones. In denoising tasks, these skip connections enable the U-Net to reintegrate high-resolution, low-level details that are typically lost during the encoding process, thus ensuring that the denoised output preserves the sharpness and intricacy of the original image. Furthermore, U-Nets inherently combine local and global contexts, making them highly effective for denoising.

In the context of diffusion models, DDPMs construct a Markov chain in a latent space, with each step resembling a denoising task. As such, leveraging the strengths of U-Nets in DDPMs becomes a natural choice. As described in Sohl-Dickstein et al. (2015) and further utilised in the work of Ho et al. (2020), a simple data generation process involving continuous-time diffusion is reversed by iterative denoising the corrupted data. This denoising step serves as the bridge between each Markovian transition, attempting to infer the original distribution of the data from the corrupted one. Moreover, the high capacity and expressiveness of U-Nets allow them to leverage the entire sequence of diffusion steps. This, as explored by Lucic et al. (2019), enables U-Nets to not only understand the denoising task at each timestep but also the evolution of noise over the diffusion process.

While theoretically powerful, the reverse process of DDPMs presents challenges that are strongly linked to the limitations of the chosen denoising network, which in the literature is always a U-Net variant. As the reverse process requires iterative noise-removal steps, the generation of a single sample involves multiple forward passes of a neural network leading to high inference times. Also, the performance is closely tied to the quality of the learned denoising function. It requires a model capable of learning complex structures and dependencies in the data, and, while U-Nets have been relatively successful in this regard, there is still room for improvement in the areas of handling diverse and complex noise patterns.

Improvements in the capabilities of the denoising model would directly enhance the performance of the reverse process and, therefore, the overall performance of the diffusion model (Lucic et al., 2019). This is what we aim to explore, *will the addition of continuous U-Net and other adaptations such as skip connections, time embeddings, and attention mechanisms, be able to translate the inherent benefits of deep implicit layers to the diffusion model to which it was applied?*

Chapter 3

Methodology

In this chapter, we outline the technical approach and methods utilised for integrating continuous U-Net within Diffusion Denoising Probabilistic Models (DDPMs) for image denoising. The discussion includes architectural modifications, configurations, and the training procedure.

3.1 Overview

In standard DDPMs, the reverse process involves reconstructing the original data from noisy observations through a series of discrete steps using variants of a U-Net architecture. In contrast, our approach (Figure 3.1) employs a continuous U-Net architecture to model the reverse process in a *locally continuous-time setting*[1].

Unlike previous work on continuous U-Nets, focusing on segmentation (Cheng et al., 2023), we adapt the architecture to carry out denoising within the reverse process of DDPMs, marking the introduction of the first continuous U-Net-based denoising network.

The most evident adaptation was adjusting the output channels of the network. Instead of predicting segmentation classes, our modified continuous U-Net is configured to produce output with dimensions equal to the number of channels in the input image. Additionally, in terms of the optimization process, we change the loss function from using a categorical cross-entropy loss to a reconstruction-based loss[2] which measures the pixel-wise deviation between the denoised image and the ground truth.

[1]The term "locally continuous-time setting" refers to a hybrid approach in which the overall training procedure adheres to a discretized framework; however, within each discrete step, the latent representation of the image is subject to a continuous-time modelling paradigm, steered by the dynamics encapsulated within a neural ordinary differential equation.

[2]See Section 3.6 for more details.

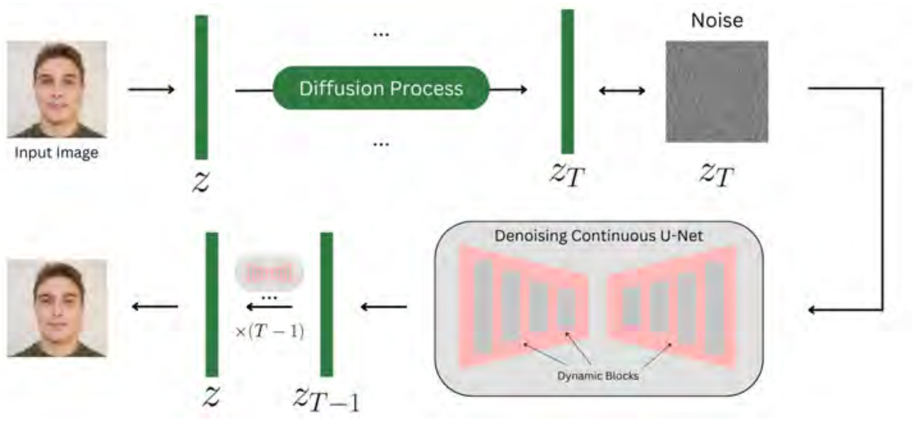


Fig. 3.1 We used a modified continuous U-Net architecture tailored for denoising in the reverse process of a diffusion model to help reconstruct the original data from a noise-corrupted version.

In denoising tasks, maintaining spatial resolution is critical (Xu et al., 2020), contrasting with segmentation tasks where resolution reduction is often acceptable. To this end, our modified continuous U-Net uses tweaked stride values to ensure minimal loss of spatial resolution. Lastly, given that denoising inherently involves handling noisy inputs, the dynamic blocks are fine-tuned for more robust noise handling and removal capabilities in the new downstream task.

A pivotal change we present is the addition of time embeddings within our denoising network. Diffusion models rely on an accurate understanding of the dynamics as the diffusion process evolves across discrete time steps. The temporal sequence is a critical parameter that informs the forward and reverse dynamics of the process. Our custom implementation of the time embeddings within our continuous U-Net-based model allows the simultaneous modelling of both the forward corruption process and the reverse recovery process. Moreover, time embeddings enable adaptive processing strategies tailored to the specific stage of the diffusion process. By recognizing the current time step, the continuous U-Net within the model can dynamically adapt to the particular characteristics of that stage, whether it's the varying noise levels or more complex latent patterns. In Section 3.4, we provide the details about our inclusion of these embeddings inside our dynamical blocks.

Our modified continuous U-Net plays a fundamental role in modelling the intricate relationships between latent variables in the diffusion model. Hence, we require a large and complex enough architecture to encapsulate the rich hierarchical structures inherent in the data and to address the challenges of modelling long-range dependencies while facilitating efficient training. Deeper networks are capable of learning higher-order and more complex representations, as well, residual connections (He et al., 2015) have been shown to enable conventional U-Nets to efficiently learn hierarchical patterns by allowing the gradient to flow freely through multiple layers thereby mitigating vanishing or exploding gradients. In addition, attention mechanisms (Vaswani et al., 2023) have proven to be effective at capturing long-range dependencies and improving a global understanding of the data. For these reasons, and in order to be able to produce high-quality samples, we are the first paper in exploring the scalability of continuous U-Net and its compatibility with the previously mentioned components (i.e. attention, residual connections, etc). More details about our architectures are shared in section 3.3.

3.2 Dynamic Blocks for Diffusion

In our pursuit to establish a foundation for a continuous U-Net model, we have modified and designed a series of dynamic blocks tailored to cater to the underlying needs of a diffusion process. This section elucidates the conceptual essence of these foundational blocks, detailing their design and showing the revised mathematical formulation of the reverse process.

3.2.1 Foundational Blocks

Firstly, because we are working with a continuous model based on a second-order ODE, we make use of an initial velocity block that determines the initial conditions for our model. We leverage instance normalisation to ensure stable feature scale across the network, necessary for tasks that demand spatial invariance. Following this, we include sequential convolution operations to process the input data and capture detailed spatial features. The first convolution transitions the input data into an intermediate representation, then, further convolutions refine and expand the feature channels, ensuring a comprehensive representation of the input. Among these operations, we include ReLU activation layers to enable the modelling of non-linear relationships as a standard practice due to its performance (Agarap, 2019).

Moreover, we designed a block that encapsulates a neural network as a function approximator to be used as the function f that defines the derivative in an ODE of the form $\frac{dz}{dt} = f(t,z)$ (Figure 3.2). This function represents how the hidden state z changes with respect to the continuous-time variable t. Here, we use group normalization layers instead to ensure feature scaling, while subsequent convolutional operations handle spatial feature extraction and processing. In order to fit the use-case of diffusion models, we implement the new addition of time embeddings through the use of multi-layer perceptrons which aim to modify the convolutional outputs by scaling and shifting operations. The resultant transformation, augmented by our own custom residual connection, provides the derivative of the hidden state with respect to time.

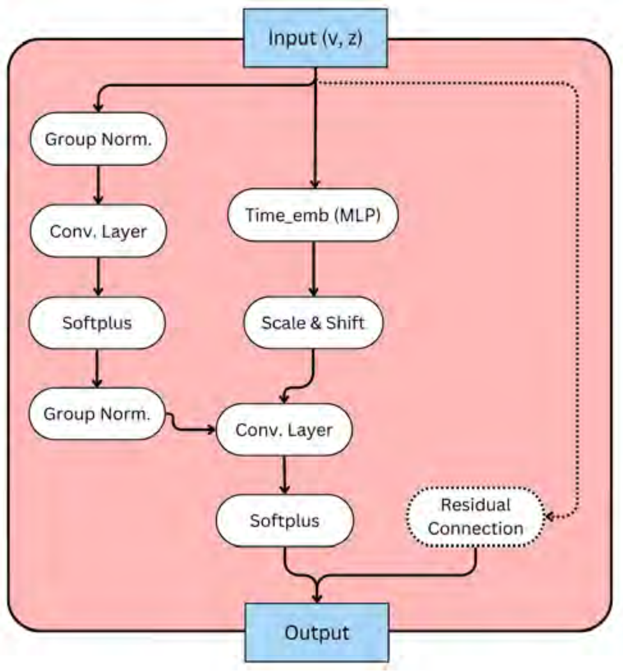


Fig. 3.2 Schematic of our custom block for approximating the derivative in an ODE. This design integrates group normalization, convolutional operations, time embeddings via multi-layer perceptrons, and custom residual connections.

Lastly, we use an ODE block that encapsulates continuous-time dynamics (Figure 3.3). By utilising an ODE function and some initial conditions both given by our previous blocks, this ODE block defines the evolutionary trajectory of the data. Depending on the use case, we offer flexibility to select the ODE solvers.

In summary, our continuous U-Net's foundation lies in its ability to interconnect these blocks. Upon initialization, the initial velocity block sets the model's starting conditions. From here, the dynamics of the model evolve, governed by the ODE block. The function approximator block, being an integral component of the ODE block, provides the specific rules for this evolution, translating spatial features and temporal information into changes over continuous time. In essence, the model's latent state undergoes transformations determined by the function approximator, while the ODE block steers its trajectory based on the dynamics outlined.

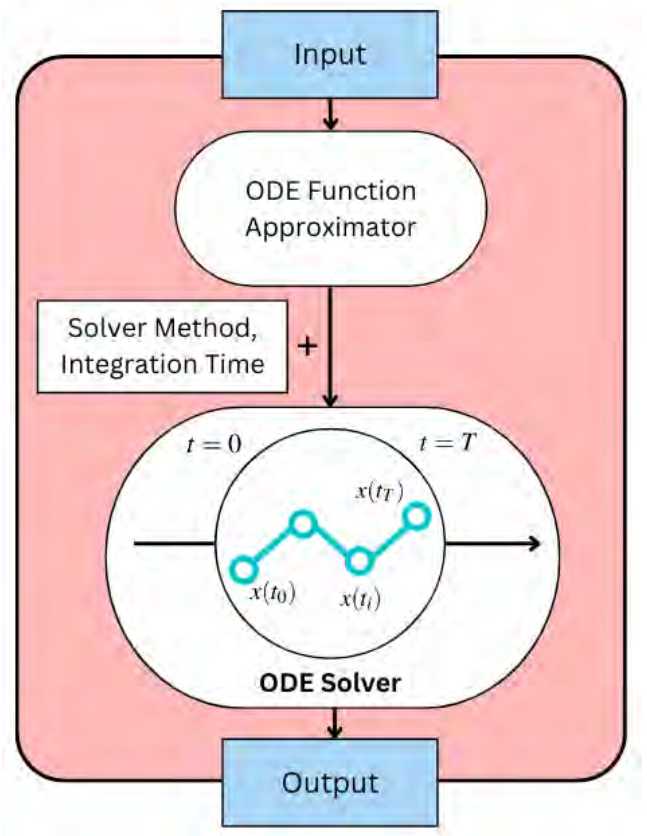


Fig. 3.3 Illustration of our custom ODE block capturing continuous-time dynamics. The block processes outputs from the derivative function approximator and uses them to delineate the data's evolutionary trajectory. It is equipped to work with various ODE solvers.

3.2.2 Mathematical Formulation

In terms of how the mathematical formulation of the reverse process of DDPMs changes, let us remind ourselves that the goal is to approximate the transition probability $q_t(x_{t-1})$ using a neural network. Denote the output of the continuous U-Net by $\tilde{U}(x,t;\theta)$, where x is the input, t is the time variable, and θ represents the parameters of the network including θ_f from the dynamic blocks built into the architecture. We can now rewrite the reverse process of DDPMs with the continuous U-Net integrated:

$$x_t = \sqrt{1-\alpha_t} \times x_{t-1} + \sqrt{\alpha_t} \times \varepsilon_t + \sqrt{\alpha_t} \times \tilde{U}(x_{t-1},t;\theta). \tag{3.1}$$

As reflected in the equation above, the transition probability $q_t(x_{t-1})$ is approximated by the continuous U-Net $\tilde{U}(x,t;\theta)$. This reformulation enables modelling the transition probability using the continuous-time dynamics encapsulated in the continuous U-Net, as opposed to traditional discrete U-Net architectures.

Going further, we can represent the continuous U-Net function in terms of the dynamical blocks:

$$\tilde{U}(x_{t-1},t;\theta) = x(t_1), \tag{3.2}$$

where,

$$\begin{cases} x''(t) = f^{(a)}(x(t),x'(t),t,\theta_f) \\ x(t_0) = X_0, \ \ x'(t_0) = g(x(t_0),\theta_g) \\ t_1 = t_0 + 1. \end{cases} \tag{3.3}$$

Here, $x''(t)$ represents the second-order derivative of the state with respect to time (acceleration), $f(a)(\cdot,\cdot,\cdot,\theta_f)$ is the neural network parameterising the acceleration and dynamics of the system, and $x(t_0)$ and $x'(t_0)$ are the initial state and velocity. The final state $x(t_1)$ at time $t_1 = t_0 + 1$ serves as the output of the continuous U-Net for the reverse process.

Note that for simplicity, in this formulation we are assuming that our continuous U-Net architecture is based on one unique dynamical block. However, we can extend this as $\tilde{U}(x_{t-1},t;\theta)$ can be considered a composite function that encompasses multiple dynamical blocks, each governed by a neural ODE. If we have K dynamical blocks within our network, we can represent the output after the i-th dynamical block as $x_i(t)$, where $1 \leq i \leq K$. We can represent $\tilde{U}$ as a composition of these blocks:

$$\tilde{U}(x_{t-1},t;\theta) = x_K(f_K(x_{K-1}(f_{K-1}(...x_1(f_1(x_{t-1},t;\theta_1),t;\theta_{K-1})...),t;\theta_K)) \tag{3.4}$$

where f_i represents the function governing the dynamics within the i-th dynamical block, and θ_i denotes the parameters of the i-th dynamical block.

3.3 Architecture Overview

In this section, we present an overview of the baseline architecture (conventional U-Net) and our proposed architecture (continuous U-Net variant) highlighting the differences and the benefits of the latter.

3.3.1 Baseline: U-Net

Our baseline U-Net architecture comprises an initial convolutional layer followed by four downsampling blocks, a bottleneck, four upsampling blocks, and a final mapping to the output channels. The initial convolutional layer has a kernel size of 7×7 and padding of 3, producing feature maps whose initial number of channels is 16. Downsampling blocks are constructed with two ResNet blocks (He et al., 2015) each containing eight groups. Time embeddings are incorporated into each of the ResNet blocks. Following these, in the 16×16 resolution, we include a full attention block. The downsampling itself is achieved using 2D convolution except in the last block. At the core of the architecture, between the downsampling and upsampling sequences, lies the bottleneck which is composed of a ResNet Block followed by an attention block and another ResNet block.

In the upsampling sequence, each block begins by concatenating the output of the previous layer with high-resolution features from the corresponding downsampling block. This is followed by two ResNet blocks similar to those in the downsampling sequence, and an attention block in the 16×16 resolution. Upsampling is implemented using 2D convolutions, analogous to the downsampling blocks. After the upsampling sequence, the feature maps are concatenated with the output of the initial convolutional layer. These concatenated features are passed through an additional ResNet block. Lastly, a 1x1 convolution maps the features to the desired output channels

3.3.2 Proposal: Modified Continuous U-Net

Our continuous U-Net design was inspired by our baseline U-Net. It starts with an initial layer that calculates the initial velocity of the input, followed by four downsampling blocks, a bottleneck, and four upsampling blocks leading to a final mapping to the output channels.

The downsampling consists of ODE blocks, each comprising a set of second-order neural ODE blocks. These ODE functions model the continuous transformation of feature maps across layers. The blocks are followed by convolutional layers that double the number of channels in the feature maps consequently. To reduce the spatial dimensions, a bilinear interpolation technique is utilised. Within the downsampling phase, an attention mechanism is incorporated in the latter part (16×16 resolution) for recalibration of the feature maps. The bottleneck is a single ODE block to process the high-dimensional features.

In the upsampling phase, each block begins by upscaling the feature maps using bilinear interpolation. These upscaled feature maps are concatenated with the corresponding feature maps from the downsampling phase. This concatenation is followed by a convolutional layer and a corresponding ODE block analogous to the downsampling phase. Here too, an attention mechanism is employed in the initial part of the upsampling phase (again, 16×16 resolution).

As we will discuss further in the upcoming section, the architecture incorporates time embeddings passed through the Transformer sinusoidal embedding function (Vaswani et al., 2023). These embeddings are utilized as additional input to the ODE blocks, providing a richer representation and allowing the model to capture the time-dependent dynamics of diffusion. Lastly, the concatenated feature maps are passed through a convolutional layer that maps the features to the desired output channels.

As for the ODE block parameters, we use a tolerance for the ODE solver of 10^{-3}, a maximum number of steps to solve the ODE of 1000, we do not use the adjoint method to compute the gradients, and we set the non-linearity to softplus.

Both our proposed model and the baseline are structurally analogous, with the distinctions being the continuous nature of the continuous U-Net and the lack of ResNet block which are substituted by our custom ODE blocks. This makes the comparison between them fair and well-grounded. For a diffusion model framework, the faster convergence of the continuous U-Net is advantageous for time-dependent simulations, as it allows for rapid and efficient approximations of the diffusion process with constant memory cost.

3.4 Residual Connections and Time Embeddings

In our continuous denoising model, the function approximator f that defines the derivative in an ODE of the form $\frac{dz}{dt} = f(t,z)$ is constructed as a convolutional block with neural ODE characteristics (i.e. time dependence, continuous dynamics, etc). Here, z represents the hidden state, and t is the continuous-time variable. As the original block in Cheng et al. (2023), ours is also designed to express how the hidden state z evolves with respect to the continuous-time variable t. In addition, we incorporate two main novelties: time embeddings and residual connections (Figure 3.2).

Given the temporal nature of diffusion processes, the inclusion of time embeddings is necessary for our model to effectively capture the progression of the transformations across time steps. The incorporation of the time variable t within our block has several advantages which led to our decision to embed time within the convolutional operations as opposed to using separate ResNet blocks like in the original DDPM implementation. First, this integration supports the parameter efficiency nature of our model as it facilitates the sharing of parameters across different time steps, resulting in a compact model that can still capture complex temporal dependencies. Secondly, by embedding time information directly within the block, we reduce redundancy in computation which can arise from having separate modules to handle time embeddings.

We use a sinusoidal embedding layer as the first processing step of the raw value of t:

$$\begin{cases} \sin(t/10000^{2d/D}) & \text{if } d \text{ is even} \\ \cos(t/10000^{2d/D}) & \text{if } d \text{ is odd} \end{cases} \tag{3.5}$$

where d is the dimension index varying from 0 to $D-1$ (with D being the dimensionality of the embedding), and t is the raw time value being embedded. The terms $10000^{2d/D}$ are frequency terms that increase geometrically, allowing the model to capture information at different timescales. With this embedding, the block employs a multilayer perceptron that transforms the time embedding through a small feedforward network, producing a scale and a shift that modulates the output. This transformation can be expressed as:

$$\begin{aligned} \text{scale, shift} &= \text{MLP}(t) \\ \text{out} &= \text{out} \times (\text{scale} + 1) + \text{shift} \end{aligned} \tag{3.6}$$

In our framework, it was noted that the training process encountered issues with vanishing or exploding gradients across different configurations. Given the significance of gradient information in capturing complex dynamics within Neural ODEs, we aimed to incorporate residual connections into the neural ODE blocks. Residual connections allow for more efficient gradient propagation through the network, helping to alleviate the vanishing and exploding gradient problems. Additionally, they enable the model to learn identity functions. Furthermore, the inclusion of residual connections within Neural ODE blocks is consistent with the mathematical representation of ODEs, where the evolution of the hidden state is expressed as a deviation from its current state.

We implemented the residual connections by integrating them directly into the block that acts as the function f used within the neural ODE solver. This design ensures that the input to the block is combined with the output, allowing for the direct flow of information. Mathematically, this can be represented as:

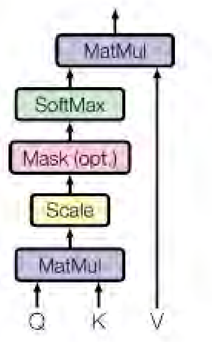
$$z_{\text{out}} = f(t, z_{\text{in}}; \theta) + R_c(z_{\text{in}}), \tag{3.7}$$

where z_{out} is the output state, z_{in} is the input state, f is the function approximation (representing the derivative in the ODE), and R_c is the residual connection that combines input with the transformation.

3.5 Attention Mechanism

The attention mechanism has been instrumental in various deep learning architectures (Vaswani et al., 2023), particularly in dealing with sequences and structured data. In our proposed architecture, we incorporate attention to enhance the ability to selectively focus on the parts of the image where the noise is significant and needs to be removed.

In particular, we use a self-attention (Vaswani et al., 2023), a mechanism wherein the queries, keys and values are derived from the same feature map x (Figure 3.4). In self-attention, the model is capable of focusing on different parts of the input for every element of the output, essentially allowing it to weigh the importance of different parts of the input.



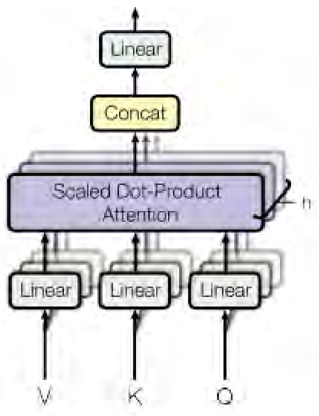


Fig. 3.4 Given queries Q, keys K, and values V, the attention mechanism computes a weighted sum of values based on the similarity (dot-product) between the queries and keys. The similarity scores are scaled by a factor and then passed through a softmax to produce the attention weights (left). Multi-head attention (right) computes attention multiple times in parallel (Vaswani et al., 2023).

We employ scaled dot-product attention to calculate the attention scores through computations that involve the query, Q, the key, K, and the value, V, representations. These are generated through linear transformations of the input tensor x via a (1×1) 2-dimensional convolution operation, resulting in three tensors of dimensions batch $\times$ channels $\times$ height $\times$ width. More specifically, attention scores are computed as follows:

$$\text{Attention}(Q,K,V) = \text{Softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}} \times V\right), \tag{3.8}$$

where d_k represents the dimensions of the key vectors. The softmax function was used to ensure that the attention scores are normalised, allowing them to be interpreted as weights. Dropout (Srivastava et al., 2014) is used on these attention scores as a common regularisation.

We integrate attention blocks between the ODE blocks at the 16×16 resolution, these refer to the fourth downscaling ODE block, and the first upscaling ODE block. The attention block during the downscaling aims to focus on important features and contexts at this scale. This is thought for encoding high-level semantic information after the spatial dimensions of the feature maps are significantly reduced due to the downscaling blocks previously applied. Secondly, integrating attention after the first upscaling ODE block in the decoder allows the model to selectively focus on features necessary for reconstruction.

In summary, we develop the first attention-based continuous U-Net model expecting this mechanism to enable our model to concentrate on global features across the entire input, as opposed to solely local features, which is beneficial for tasks that require understanding the overall content and context of an image (Vaswani et al., 2023). This can support the previously discussed residual connections to mitigate the vanishing gradient problems observed earlier.

3.6 Training Procedure

In our work, we train the DDPM using a continuous U-Net in the reverse process. The core training procedure retains the fundamental algorithmic setup from the original DDPM, but with some necessary modifications and optimizations. We fit our model by maximising the evidence lower bound (ELBO), similar to how VAEs (Kingma and Welling, 2022) are trained. In particular, for each data example $\mathbf{x}_0$, we have:

$$\log p_\theta(\mathbf{x}_0) = \log \left[\int d\mathbf{x}_{1:T} q(\mathbf{x}_{1:T}|\mathbf{x}_0) \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right]$$

$$\geq \int d\mathbf{x}_{1:T} q(\mathbf{x}_{1:T}|\mathbf{x}_0) \log \left(p(\mathbf{x}_T) \prod_{t=1}^{T} \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right)$$

$$= \mathbb{E}_q \left[\log p(\mathbf{x}_T) + \sum_{t=1}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] \mathrel{\hat=} Ł(\mathbf{x}_0) \tag{3.9}$$

By invoking the Markov property, we have $q(x_t|x_{t-1}) = q(x_t|x_{t-1}, x_0)$. Applying Bayes' rule yields:

$$q(x_t|x_{t-1}, x_0) = \frac{q(x_{t-1}|x_t, x_0) q(x_t|x_0)}{q(x_{t-1}|x_0)} \tag{3.10}$$

Substituting this into the ELBO, we get:

$$\mathscr{L}(x_0) = \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log p(x_T) + \sum_{t=2}^{T} \frac{\log p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)} + \sum_{t=2}^{T} \frac{\log q(x_{t-1}|x_0)}{q(x_t|x_0)} + \frac{\log p_\theta(x_0|x_1)}{q(x_1|x_0)}\right] \tag{3.11}$$

Note that the third term inside this expectation can be simplified as follows:

$$\sum_{t=2}^{T} \frac{\log q(x_{t-1}|x_0)}{q(x_t|x_0)} = -\log q(x_T|x_0) + \log q(x_1|x_0) \tag{3.12}$$

Consequently, the negative ELBO (the variational upper bound) becomes:

$$\mathscr{L}(x_0) = -\mathbb{E}_{q(x_{1:T}|x_0)}\left[\frac{\log p(x_T)}{q(x_T|x_0)} + \sum_{t=2}^{T} \frac{\log p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)} + \frac{\log p_\theta(x_0|x_1)}{q(x_1|x_0)}\right]$$

$$= \underbrace{D_{KL}\left(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)\right)}_{=L_T(\mathbf{x}_0)} + \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} D_{KL}\left(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)\right)}_{=L_{t-1}(\mathbf{x}_0)} - \underbrace{\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{=L_0(\mathbf{x}_0)} \tag{3.13}$$

Each KL-divergence term in L_{VLB}, can be calculated in closed form as all the distributions are Gaussian. L_T is a constant, so it can be disregarded during the training phase because it lacks modifiable parameters and $\mathbf{x}_T$ is subject to Gaussian noise.

In the following, we shift our focus to the L_{t-1} term. Given that $\mathbf{x}_t = \sqrt{\alpha_t \mathbf{x}_0} + \sqrt{1-\alpha_t}\varepsilon$, we can rewrite the term as follows:

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon\right) \tag{3.14}$$

Rather than teaching the model to predict the mean of the denoised version of $\mathbf{x}_{t-1}$ from its noisy counterpart $\mathbf{x}_t$, we can instruct the model to predict the noise. Consequently, we can compute the mean as:

$$\mu_\theta(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(\mathbf{x}_t, t)\right) \tag{3.15}$$

The loss function, averaged over the dataset, can be expressed as:

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0 \sim q_0(\mathbf{x}_0), \varepsilon \sim \mathcal{N}(0,I)} \left[\underbrace{\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1-\bar{\alpha}_t)}}_{=\lambda_t} ||\varepsilon - \varepsilon_\theta \left(\underbrace{\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon}_{=\mathbf{x}_t}, t \right) ||^2 \right] \tag{3.16}$$

The term outside the $|| \cdot ||^2$ norm, i.e. the time-varying weight λ_t, guarantees that the training objective corresponds to maximum likelihood training, assuming the variational bound is tight. However, empirically, Ho et al. (2020) shows that the model works best when this weighting term is trivially equal to 1. The resulting simplified loss, which is also averaged over the time steps t in the model, is given by:

$$L_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0 \sim q_0(\mathbf{x}_0), \varepsilon \sim \mathcal{N}(0,I), t \sim \text{Unif}(1,T)} \left[||\varepsilon - \varepsilon_\theta \left(\underbrace{\sqrt{\alpha_t \mathbf{x}_0} + \sqrt{1-\alpha_t}\varepsilon}_{=\mathbf{x}_t}, t \right) ||^2 \right] \tag{3.17}$$

where t is a uniform random variable between 1 and T. This definition discards the weighting in Eq. (12) which leads to a reweighting that improves the quality of the samples generated by the model.

The continuous U-Net model predicts the noise $\varepsilon_\theta(x_t, t)$ given x_t and t. As a denoiser, it provides us with the advantage of leveraging the continuous structure to more flexibly sample from different noise levels during the reverse process, allowing us to better explore the latent space and improve the generation performance of the model. Algorithmically, the training procedure is presented in Algorithm 1.

Algorithm 1 Training procedure for the DDPM model.

```
1: **repeat**
2:     Sample initial data point $x_0 \sim q(x_0)$
3:     Sample time step $t \sim \text{Uniform}(\{1, \ldots, T\})$
4:     Sample noise $\varepsilon \sim \mathcal{N}(0, I)$
5:     Take gradient descent step on $\nabla_\theta \left|\left|\varepsilon - \varepsilon_\theta\left(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon, t\right)\right|\right|^2$
6: **until** convergence
```

Chapter 4

Evaluation and Results

This chapter delves into a comprehensive evaluation and analysis of our proposed model in comparison to discrete U-Nets, other deep models, and traditional methods.

The main objectives of this chapter are to examine the performance of the proposed model in terms of image synthesis, denoising, and parameter efficiency while discussing its current limitations. We will show how our method yields comparable results in the task of image synthesis and perceptually outperforms other methods when it comes to denoising, and all of this with only approximately a fourth of the number of parameters.

We start this chapter by presenting an in-depth analysis of the model's performance on image synthesis and denoising tasks. These sections will provide quantitative assessments based on chosen metrics, supplemented by qualitative visual comparisons to provide an intuitive understanding of the results. Subsequently, we offer an evaluation of the model's parameter efficiency. We demonstrate how the Continuous U-Net model's continuous nature leads to the efficient utilization of parameters and contributes to the model's overall usability. Lastly, we confront the limitations of our proposed model. We aim to provide a balanced view of its capabilities and discuss areas where the model can further improve.

4.1 Experimental Setup

In Section 3.3, we detailed our intent to compare models with structurally similar architectures. Relying on findings from Cheng et al. (2023), we understand that the continuous U-Net does not require significant scaling to match or surpass other U-type architectures. Given this advantage, our chosen baseline for the proposed framework is a more compact continuous U-Net compared to the conventional U-Net.

The nature of our work is to compare diffusion models with the two types of U-Nets as denoising networks in the reverse process. However, for the denoising downstream task, we believe it is a useful addition to compare these models with other traditional end-to-end trained denoisers. This will not only highlight the capacity of the reverse process of each of the models but will also prove their ability to be used, not only for synthesis but also as potential denoisers. For this reason, we also implement a classical (but powerful) technique called BM3D (Dabov et al., 2007), and two deep learning solutions - a convolutional autoencoder and a DnCNN model (Zhang et al., 2017).

Finally, and as a disclaimer, to ensure a fair comparison of inference times between models, we have made the decision to measure the times based on our models running on CPUs. This approach is primarily due to current implementation constraints associated with ODE solvers in Python. A deeper discussion on these limitations and their implications on our work can be found in Section 4.6. We believe that this approach offers a transparent and consistent metric for model comparison in the context of our research.

Additionally, in the following sections, we explain in detail, the particularities of our dataset preprocessing, the metrics that we choose for evaluation, and the specific hyperparameters used for training.

4.1.1 Dataset Description

We employ three distinct datasets. The first, the *MNIST* dataset (LeCun et al., 2010), is a benchmark in the domain of machine learning and consists of grayscale images representing handwritten digits (Figure 4.1), spanning from 0 to 9. It boasts a collection of 60,000 images, each with a resolution of 28x28 pixels. To augment our training process and enhance the generalizability of the model, we introduce random horizontal flips as a transformation on this dataset. We also incorporate the *CelebA* dataset (Liu et al., 2015), a renowned resource in facial attributes studies. This dataset is composed of celebrity portraits (Figure 4.2), emphasizing facial features with substantial variance in terms of age, ethnicity, and other facial attributes. We utilize a subset of 30,000 images from CelebA, each resized to a resolution of 64x64 pixels for consistency and computational efficiency. To further diversify the training data and induce robustness, we apply random horizontal flips and center crops as our chosen data transformations. Lastly, we integrate the *LSUN Church* dataset into our training regimen. LSUN Church is part of the larger LSUN suite and is characterized by its images of churches (Figure 4.3), capturing the intricacies of architectural designs. This dataset provides a unique challenge, with its 126,227 images of churches, each standardized to a resolution of 64x64 pixels for our experiments. Given the inherent diversity and richness of the dataset, we opt for no additional transformations, preserving its original visual structure and information.

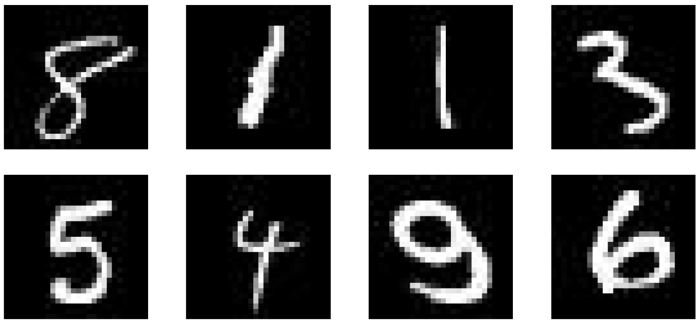


Fig. 4.1 Sample images from the MNIST dataset: Grayscale representations of handwritten digits ranging from 0 to 9. Each image is of size 28x28 pixels.

Fig. 4.2 Selected images from the CelebA dataset: Portraits of celebrities showcasing diversity in facial attributes such as age, ethnicity, and features. Images are resized to 64x64 pixels for our experiments.

4.1.2 Evaluation Protocol

Our evaluation protocol involves a standard and comprehensive set of metrics to assess the performance of our models across image synthesis, denoising, and efficiency. To evaluate the quality of the samples generated by our models, we use the Fréchet Inception Distance (FID) (Heusel et al., 2018). This is a popular metric used to evaluate the quality of samples generated by generative models and is designed to measure the similarity between two datasets of images. It does this by comparing the distributions of the Inception network's

Fig. 4.3 Example images from the LSUN Church dataset: Various churches depicting architectural intricacies and designs. Images are standardized to 64x64 pixels for analysis.

(Szegedy et al., 2014) activations for each dataset. This means that the Inception network is used as a feature extractor, and the mean and covariance of these features are then computed for each of the datasets being compared - the training dataset and one made up of generations of the trained generative model. For example, the Fréchet distance between two multivariate Gaussians is calculated as:

$$\mathrm{FID}(x,g) = ||\mu_x - \mu_g||^2 + \mathrm{tr}\left(C_x + C_g - 2\sqrt{C_x C_g}\right), \tag{4.1}$$

where $||\mu_x - \mu_g||^2$ is the L_2 distance between the mean of the real and generated samples to account for aspects of the quality of the individual generated samples through the mean term. On the other hand, $\mathrm{tr}\left(C_x + C_g - 2\sqrt{C_x C_g}\right)$ is the trace of the covariance matrix of the real and generated samples. This last term measures the similarity between the covariance matrices of the two samples, hence capturing the diversity of the generated samples. In general, a low FID means that the images are more similar to the real ones, which means that the generated model is producing better-quality images. On the other hand, a high FID score means that the samples have a feature distribution more distant from that of the real images.

For denoising, we evaluate the quality of denoised samples by considering mainly LPIPS and SSIM. The Structural Similarity Index (SSIM) is a distortion metric quantifying the visual similarity between two images. It evaluates structural information by comparing local patterns of pixel intensities, with a value of 1 indicating identical structural information between the compared images. Learned Perceptual Image Patch Similarity (LPIPS) (Zhang et al., 2018) is a distance metric that utilizes deep neural networks to compute perceptual differences between images. It measures the distance between embeddings produced by a pre-trained network, capturing high-level perceptual and semantic differences between the images. In the context of generative modelling and denoising, LPIPS is generally favoured over other commonly used metrics such as SSIM and PSNR due to its ability to capture perceptual dissimilarities that are consistent with human judgment (Figure 4.4).

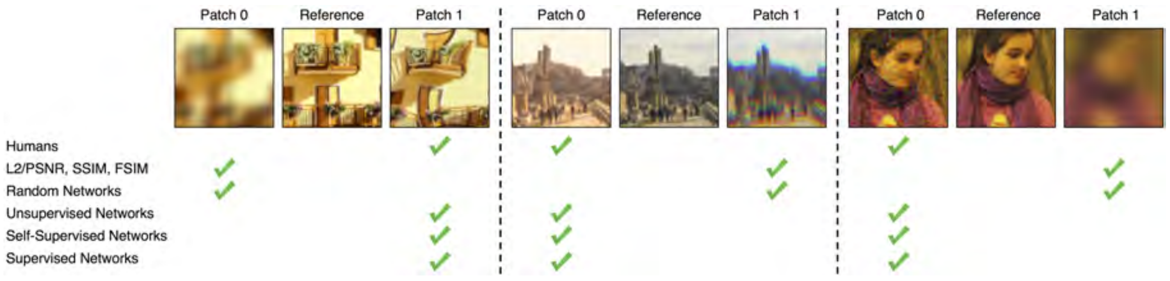


Fig. 4.4 Image borrowed from Zhang et al. (2018) showcasing the effect of perceptual versus distortion metrics. Traditional metrics (L2/PSNR, SSIM) consistently diverge from human evaluations. In contrast, deep network models, irrespective of their architecture or type of supervision, yield embeddings that align remarkably well with human judgments.

Finally, we compare the efficiency of our benchmarked models in terms of their number of parameters, and their FLOP counts. Firstly, the number of parameters in a model offers a direct measure of its size. Models with fewer parameters are generally more memory-efficient, requiring less storage space when saved and potentially leading to faster inference times. Meanwhile, FLOPs (Floating Point Operations Per Second) provide an estimate of the model's computational complexity during inference. A lower FLOP count suggests that the model can process data more swiftly, leading to quicker response times, which is especially crucial in real-time processing scenarios. By considering both the number of parameters and FLOPs, we aim to gauge not just the performance but also the operational practicality of our models.

4.1.3 Training Scheme

With the theoretical underpinnings and the derived loss function laid out in section 3.6, it is important to delve into the practical aspects of implementing and training our model. For this reason, we will proceed to present the training setup, including hardware details, hyper-parameters, training data, and regularization techniques, which are designed to maximize performance and efficiency.

The hardware used for the training of both our proposal and baseline models consists of NVIDIA Ampere A100 with 40GB of RAM. The model underwent approximately 500,000 training steps for the CelebA and the LSUN datasets, and significantly lesser for 100,000 steps in the MNIST dataset.

For the experiments with the LSUN Church and the CelebA datasets, with images resized to 64x64 pixels, we employed a learning rate of 2×10^{-4}, with a batch size of 128. For the MNIST dataset, we utilized a learning rate of 2^{-5} and a batch size of 256. For all these setups, the optimiser used was Adam. Hyperparameters specific to the continuous U-Net included a maximum of 1000 steps for the ODE solver, a tolerance set to 10^{-3}, and time-embedding column tensors of dimension 256. Additionally, we used the rk4 ODE solver with the adjoint flag set to False. Inside the block designed to be the function f, which defines the derivative in the ODE of the form $\frac{dz}{dt} = f(t,z)$, we used softplus activation functions rather than ReLU to mathematically ensure a solution (Gholami et al., 2019), leaving the parameters in the ODE block as default.

As mentioned before, we used three main datasets for our different experiments: MNIST, LSUN Church, and CelebA (64x64). We employed the same data augmentations as those used in Ho et al. (2020), including horizontal flips, random crops, and random translations for the 64x64 dataset. For MNIST, these augmentations consisted of random translations and rotations. For regularization purposes, we incorporated dropout and RMSNorm within the attention blocks. In addition, group normalisation was used inside the derivative function approximator block. These techniques aid in avoiding overfitting. Outside of that, we prioritised keeping our hyperparameter choices consistent with the ones utilized in Ho et al. (2020). This choice was primarily to ensure a fair comparison between the models.

Lastly, let us describe the training setup we chose for the deep learning models we implemented for the denoising benchmarking. Our DnCNN architecture is tailored for image denoising, operating on default 3-channel (RGB) images and comprising 20 convolutional layers. The initial layer uses 64 filters followed by ReLU activation. Subsequent layers consist of convolutions with 64 filters paired with batch normalization and ReLU. The final layer projects back to the image's original channel size, aiding in noise prediction. During inference, the predicted noise is subtracted from the input through residual learning. In terms of our convolutional autoencoder, the encoder sequentially applies four convolutional layers with ReLU activations, reducing the spatial dimensions from 64x64 to 4x4 while expanding the channels from 3 to 512. The decoder mirrors this process, using transposed convolutions to upsample the spatial dimensions back to 64x64 and compressing the channels down to 3, followed by a sigmoid activation. Both models were trained using mean squared error loss, epochs ranged from 10-50 depending on the noise level and we used a batch size of 128, employing the Adam optimizer at a learning rate of 0.001.

4.2 Image Synthesis

We know that the generation process in a DDPM is modelled as a Markov chain, starting from a Gaussian distribution and gradually diffusing it to match the ground truth data distribution. This is a process that is implemented through a series of diffusion steps, each one slightly altering the distribution towards the target. It can be mathematically expressed as a stochastic differential equation:

$$dx(t) = \sqrt{\beta(t)}dB(t) - \frac{1}{2}\beta(t)x(t)dt \tag{4.2}$$

This is an Ornstein-Uhlenbeck process which is often used to model systems with a tendency to revert to a long-term mean, i.e. it is a simple model of Brownian motion with a drift term that pulls the process back towards the mean. In our context, the process $x(t)$ represents the state of the system (e.g. an image) at time t, and the Brownian motion $B(t)$ represents the random noise added at each step. The deterministic part of the equation would then represent the denoising operation trying to revert the system back to its original state.

To carry out image synthesis, we can reverse this diffusion process, starting with a sample from a simple Gaussian and then applying the reverse diffusion for a number of steps. Here is where the relevance of our denoising models comes into play. We use these networks to approximate a learned denoising function f that attempts to reconstruct $x(t)$ given $x(t+1)$. In this way, we can draw a sample z from a simple Gaussian distribution and then apply the learned denoising function f for a fixed number of timesteps.

For a qualitative assessment of the images produced, we randomly selected samples from both the baseline and our proposed models, adjusting the sampling timesteps based on each model and dataset. In order to determine the sampling timesteps, we compared the convergence performance of the models by analyzing the FID against the number of sampling timesteps (Figure 4.5). Our diffusion model typically achieved optimal FID values with fewer timesteps than the U-Net-based model, as shown in Table 4.1, allowing for the production of higher-quality samples with fewer timesteps. In terms of the quantitative results, we evaluate the results using their FID score (Table 4.1). To do this, we generated two datasets, each containing 30,000 generated samples from each of the models, in the same way as we generated the images shown in the figures above. These new datasets are then directly used for the FID score computation with a batch size of 512 for the feature extraction. We also note that we use the 2048-dimensional layer of the Inception network for feature extraction as this is a common choice to capture higher-level features.

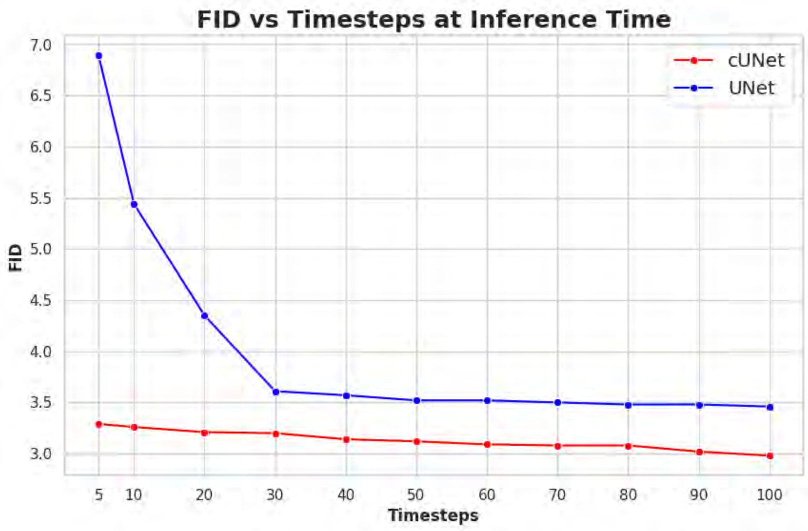


Fig. 4.5 Comparison of FID values as a function of sampling timesteps for both our proposed diffusion model and the U-Net-based model. The graph demonstrates the efficiency of our model in achieving optimal FID scores with fewer timesteps.

Table 4.1 Performance metrics across datasets: FID scores, sampling timesteps (Steps), and average generation time for both the U-Net and continuous U-Net (cU-Net) models. The table highlights the efficiency and effectiveness of the proposed cU-Net in image generation.

Network	**MNIST**			**CelebA**			**LSUN Church**		
	FID	**Steps**	**Time (s)**	**FID**	**Steps**	**Time (s)**	**FID**	**Steps**	**Time (s)**
U-Net	3.61	30	3.56	**19.75**	100	12.48	12.28	100	12.14
cU-Net	**2.98**	**5**	**0.54**	21.44	**80**	**7.36**	**12.14**	**90**	**8.33**

From Table 4.1 we also see that both models achieve similar performance with our continuous U-Net-based reverse process slightly leading the table in the MNIST and LSUN datasets, and demonstrating a small disadvantage for CelebA. Figures 4.6 and 4.7 show samples from the U-Net-based and continuous U-Net-based models trained on MNIST. Likewise, Figures 4.8 and 4.9 present results from models trained on the CelebA 64×64 dataset, and Figures 4.10 and 4.11 for the LSUN Church dataset. Overall, our key qualitative takeaways from visually inspecting the samples are the following:

1. For MNIST, both models seem to produce excellent samples. However, we observe that it is a common behaviour of the U-Net DDPM to generate odd-looking digits that are hard to distinguish. As well, we spotted several samples that still contained some level of noise even after the 256 sampling steps. We did not find either of these problems in the samples generated by our proposed model.

2. For the CelebA models, we observe a clear instability issue with the background of the samples generated by our continuous U-Net-based model. This is not the case for the baseline model in which the background of the generated samples is usually single-coloured and smooth. We discuss this further, together with potential solutions in Section 4.6. It is worth noting that even if this happens with the background of the images, both models achieve to be consistent with face features to a good degree of quality as appreciated in Figures 4.8 and 4.9.

3. For the LSUN Church, we found many cases in which the generations by the U-Net-based model suffered from an evident colour saturation, similar to mode collapse in other generative models. Again, this issue was not encountered during the analysis of the continuous U-Net samples.

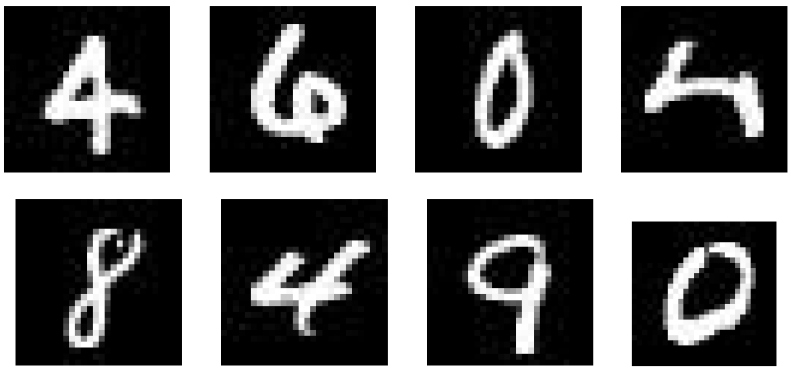


Fig. 4.6 Randomly selected generated samples from our baseline diffusion model.

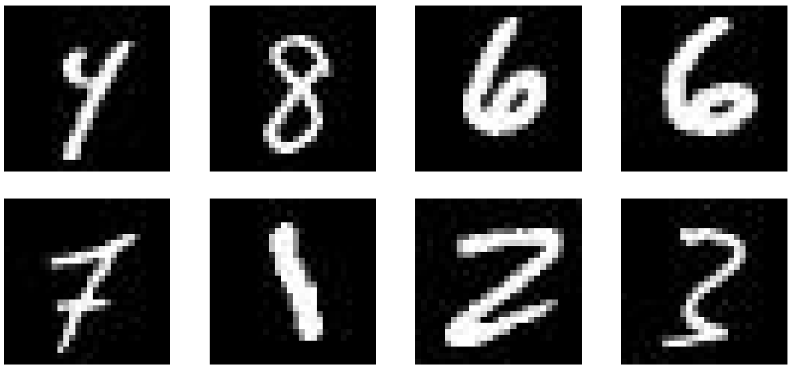


Fig. 4.7 Randomly selected generated samples from our continuous U-Net-based model.

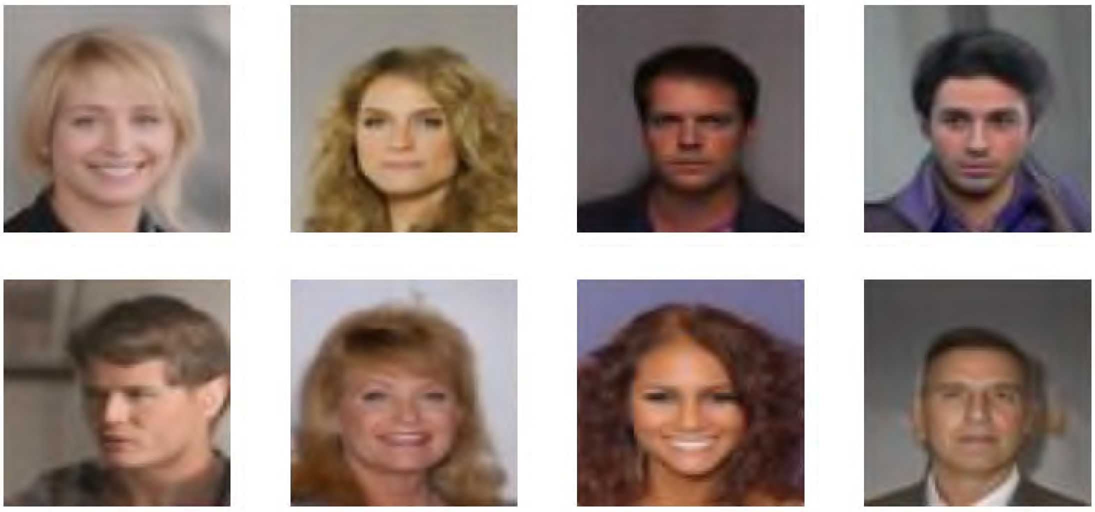


Fig. 4.8 Randomly selected generated samples from our baseline diffusion model.

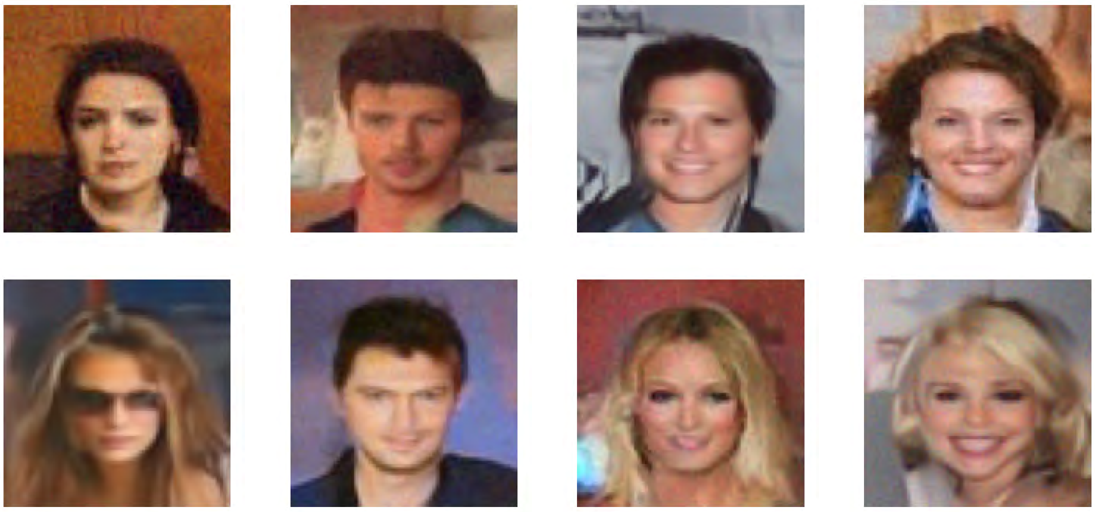


Fig. 4.9 Randomly selected generated samples from our continuous U-Net-based model.

4.3 Denoising

Denoising forms a crucial part of diffusion models because it approximates the reverse of the Markov chain formed by the forward process, starting with a sample from the noise distribution and progressively denoising it until it resembles a sample from the data distribution.

Improving the denoising mechanism can thus directly lead to improvements in the reverse process of the diffusion model. The reason for this is that the reverse process relies on accurately estimating the conditional distribution of the data given the corrupted samples at each timestep. If the denoising mechanism can more accurately estimate this conditional

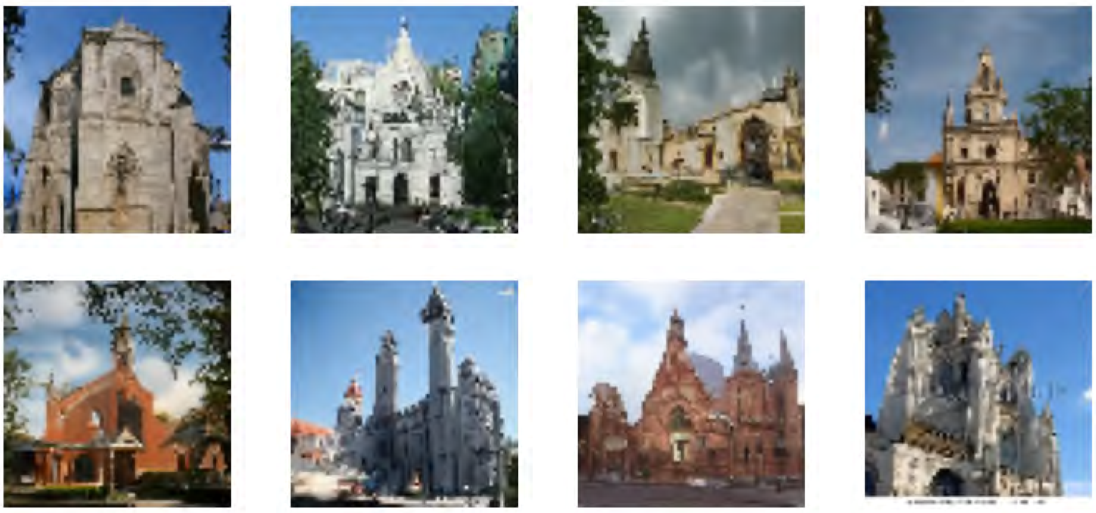


Fig. 4.10 Randomly selected generated samples from our baseline diffusion model.

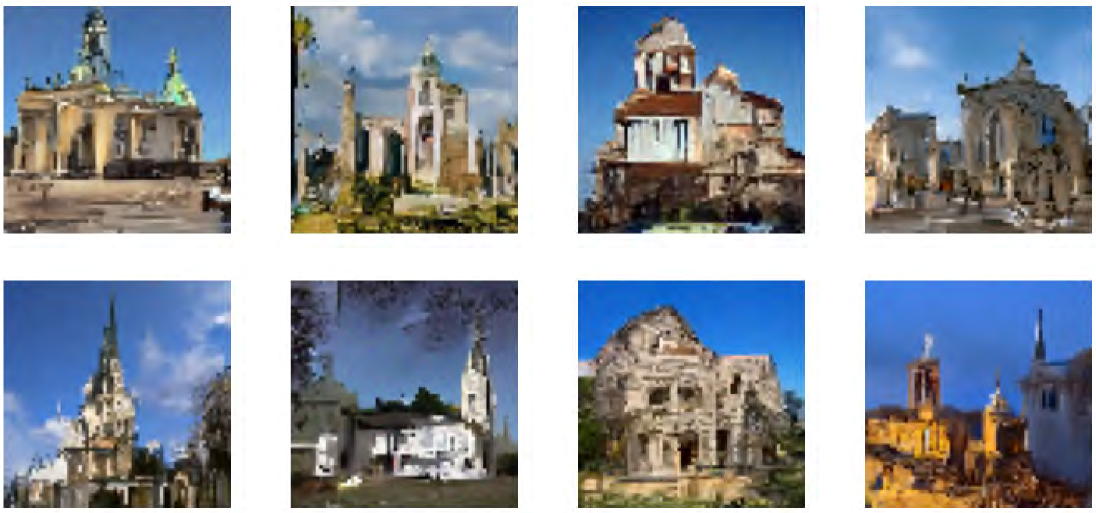


Fig. 4.11 Randomly selected generated samples from our continuous U-Net-based model.

distribution, the model will make more accurate steps during the reverse process, leading to generated samples that are closer to the data distribution. Furthermore, an effective denoising mechanism can allow the diffusion model to make more significant transformations at each timestep. This can potentially make the reverse process more efficient by allowing it to reach the data distribution in fewer steps, hence reducing computational costs.

The process of noising images in our experiments is closely linked to the role of the denoising networks in the reverse process. Here, the networks use the timesteps, representing the progression of the diffusion process, to determine the anticipated noise level of the input image at a given point in time. Such a relationship arises because the model exploits time embeddings to gauge the expected noise magnitude for specific timesteps. To determine these noise levels accurately, we employ the forward process up to the designated timesteps. During this forward phase, the image undergoes a gradual transition, accumulating noise with each timestep. This is clearly illustrated in Figure 4.12, where an increased timestep corresponds to higher noise levels. Thus, the noise level can effectively be seen as a function of the timesteps of the forward process.

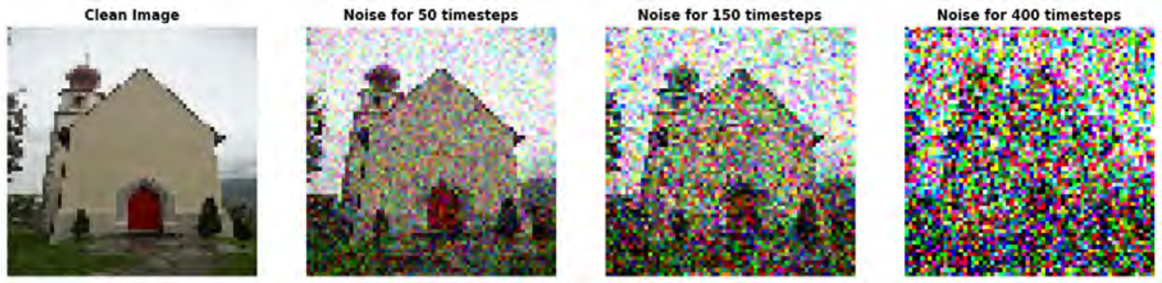


Fig. 4.12 Visualization of noise accumulation in images over increasing timesteps. As timesteps advance, the images exhibit higher levels of noise, showcasing the correlation between timesteps and noise intensity. The progression highlights the effectiveness of time embeddings in predicting noise magnitude at specific stages of the diffusion process.

For our main denoising experiment, a test dataset consisting of 300 images was utilised to assess the average performance of our models over varying noise levels. We monitored SSIM and LPIPS across an extensive range of timesteps to understand the distortion and perceptual differences in the model outputs. As shown in Table 4.2, there exists an evident disparity in the strengths of each of the models. The conventional U-Net predominantly attains higher SSIM scores, whereas our proposed models excel in LPIPS evaluations. Despite SSIM being considered as a metric that measures perceived quality, it has been observed to have a strong correlation with simpler measures like PSNR (Horé and Ziou, 2010) due to being a distortion measure. Notably, PSNR tends to favour over-smoothed samples, which suggests that a high SSIM score may not always correspond to visually appealing results but rather to an over-smoothed image. This correlation underscores the importance of using diverse metrics like LPIPS to get a more comprehensive view of denoising performance.

The results yielded by the U-Net are symptomatic of a well-known issue in supervised learning methods employed for denoising. When models are trained using paired clean and noisy images through a distance-based loss function, they tend to output denoised solutions that are excessively smooth. This is because the underlying approach frames the denoising task as a deterministic mapping from a noisy image y to its clean counterpart x. From a Bayesian viewpoint, when conditioned on x, y follows a posterior distribution:

Table 4.2 Comparative average denoising performance between U-Net (left values) and cU-Net (right values) for different noise levels over the test dataset. While U-Net predominantly achieves higher SSIM scores, cU-Net often outperforms in LPIPS evaluations, indicating differences in the nature of their denoising approaches.

Noising Timesteps	Best SSIM Value	Best LPIPS Value
50	0.88 / **0.90**	0.025 / **0.019**
100	**0.85** / 0.83	0.044 / **0.038**
150	**0.79** / 0.78	0.063 / **0.050**
200	**0.74** / 0.71	0.079 / **0.069**
250	**0.72** / 0.64	0.104 / **0.084**
400	**0.58** / 0.44	0.184 / **0.146**
600	**0.44** / 0.26	0.316 / **0.238**
800	**0.32** / 0.18	0.419 / **0.315**

$$q(x|y) = \frac{q(y|x)q(x)}{q(y)}. \tag{4.3}$$

If the distance of the loss function is the L2 norm, the resulting model effectively estimates $\mathbb{E}[x|y]$, which translates to the posterior mean. This gives insight into the observed over-smoothing in traditional supervised learning outcomes. We do not only claim that this higher SSIM might be related to over-smoothed images, but we also provide examples to showcase this issue. In Appendix A, we show images at different noise levels and the denoised outputs by each of our models (Figure A.4). One can observe our model achieving consistent results with more fine-grained details. In fact, at higher noise levels where neither of the models is capable of recovering details (in this case from the face), our model attempts to predict the features of the image instead of prioritising the smoothness of the texture (Figure A.3). We have several hypotheses for why this could be happening, for instance, as our continuous U-Net processes information over a continuum of scales, it may become better suited to refine and predict intricate details. Also, continuous U-Net might have learned a more "aggressive" denoising strategy, where it's more willing to introduce features that it "believes" should be present, based on the training data. However, determining the exact reason for this phenomenon remains part of our future work.

Figures 4.13 and 4.14 are a perfect example of what is known as the *Perception-Distortion tradeoff*. This tradeoff exists for any distortion measure and is mathematically proven using rate-distortion style arguments Blau and Michaeli (2018). Intuitively, averaging and blurring reduce distortion, but make images look unnatural. Perceptual quality is quantified using the total variation (TV) distance between the distribution of reconstructed images $p_{\hat{X}}$ and the distribution of natural images p_X. The TV distance is given by:

$$d_{\text{TV}}(p_{\hat{X}}, p_X) = \frac{1}{2}\int |p_{\hat{X}}(x) - p_X(x)|\, dx. \tag{4.4}$$

The metric evaluates the divergence between the two distributions. As d_{TV} diminishes, $p_{\hat{X}}$ converges to p_X, leading to enhanced perceptual quality. One can define a perception-distortion function $P(D)$ which represents the minimal TV distance, which translates to the optimal perceptual quality for a given distortion bound D:

$$P(D) = \min_{p_{\hat{X}|Y}} d_{\text{TV}}(p_{\hat{X}}, p_X) \quad \text{s.t.} \quad \mathbb{E}[\Delta(X, \hat{X})] \leq D. \tag{4.5}$$

In the above equation, the minimization spans over estimators $p_{\hat{X}|Y}$, and $\Delta(X, \hat{X})$ characterizes the distortion metric. An important observation is that $P(D)$ exhibits monotonic decrement and is convex in nature. On a fundamental level, raising the permissible distortion D broadens the feasible domain, thereby enabling estimators with improved perceptual quality, causing $P(D)$ to diminish. The convex nature of the function can be illuminated by considering two operating points $(D_1, P(D_1))$ and $(D_2, P(D_2))$ alongside their weighted mean. The convexity of the TV distance then ensures:

$$\lambda P(D_1) + (1-\lambda)P(D_2) \geq P(\lambda D_1 + (1-\lambda)D_2), \tag{4.6}$$

where λ is a scalar weight that is used to take a convex combination of two operating points. This convexity underlines a rigorous trade-off at lower D values. Diminishing the distortion beneath a specific threshold demands a significant compromise in perceptual quality.

Moreover, a key component of our experiment involved monitoring the specific timestep at which each model reached its peak performance in terms of both SSIM and LPIPS. Concurrently, we measured the elapsed time taken by each model to reach this optimal timestep. Encouragingly, our proposed model consistently outperformed in this aspect, delivering superior inference speeds and requiring fewer timesteps to converge. These promising results are compiled and can be viewed in Table 4.3.

In our last experiment, we compared the reverse processes of our two diffusion models with other denoising methods. Referencing Section 4.1.3, we tested against the DnCNN, a convolutional Autoencoder, and BM3D - a state-of-the-art classical technique. From Table 4.4, when timesteps are low, most models perform well, with our proposed model leading in both metrics. For higher timesteps, here again, the standard DDPM with a typical U-Net shows good SSIM results, while our continuous U-Net model performs best in perceptual quality. In all cases, we show that both U-Nets, without the need to be retrained can handle denoising to a high degree of quality for a wider range of noise levels as the other deep

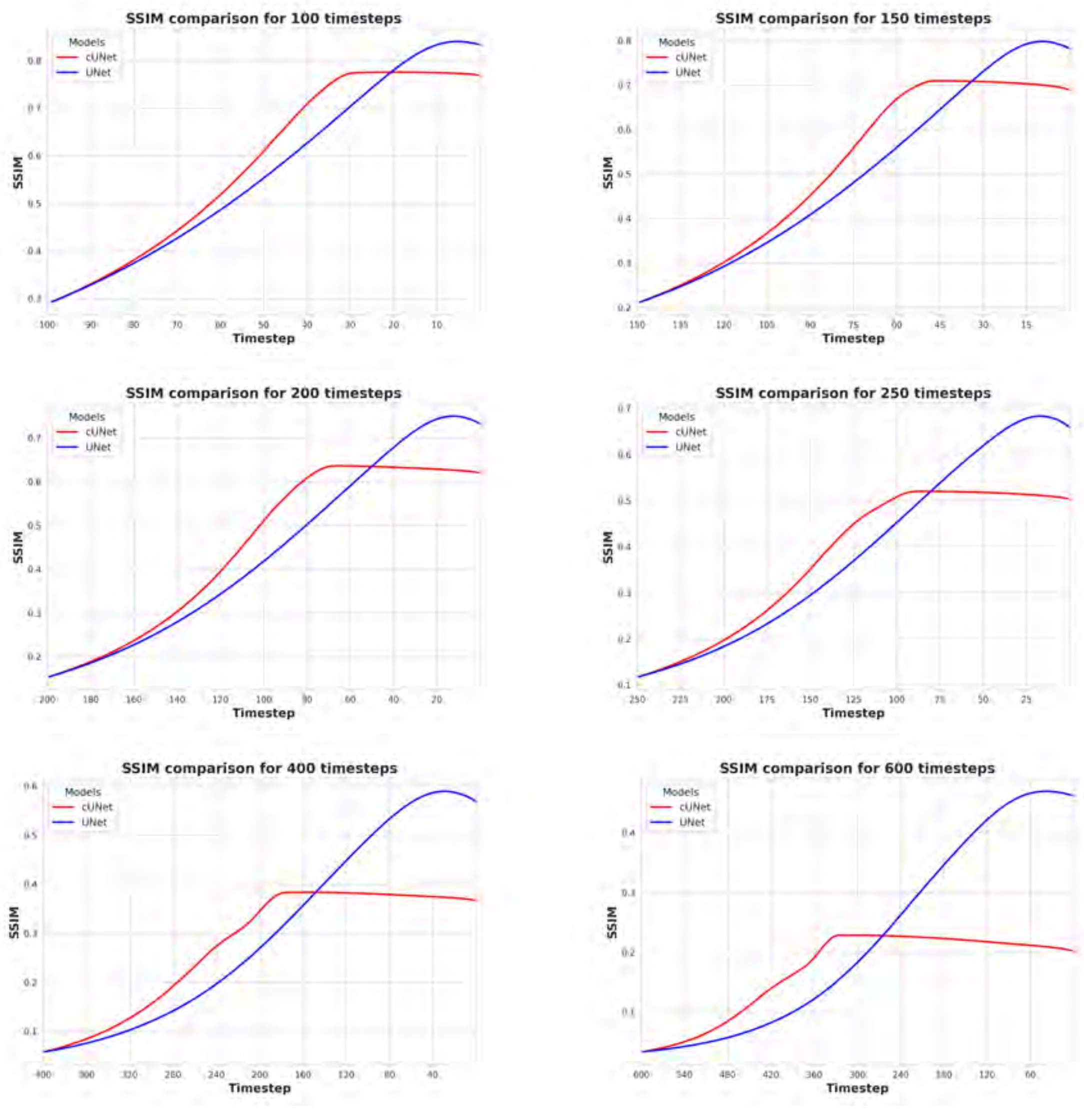


Fig. 4.13 SSIM scores plotted against diffusion steps for varying noise levels for one image. The graph underscores the consistently superior performance of U-Net over cU-Net in terms of SSIM, particularly at high noise levels. This dominance in SSIM may be misleading due to the inherent Distortion-Perception tradeoff and the tendency of our model to predict features instead of distorting the images.

learning techniques were specifically trained for each of the noise levels. This is a benefit of using the pre-trained diffusion models, their learned probability distribution is much wider and they can generalise well across many more noise levels.

In summary, throughout our experiments, we observed a distinctive tradeoff between perception and distortion. The conventional U-Net generally scored higher in SSIM, indicating less distortion. On the other hand, the continuous U-Net performed better in perceptual metrics.

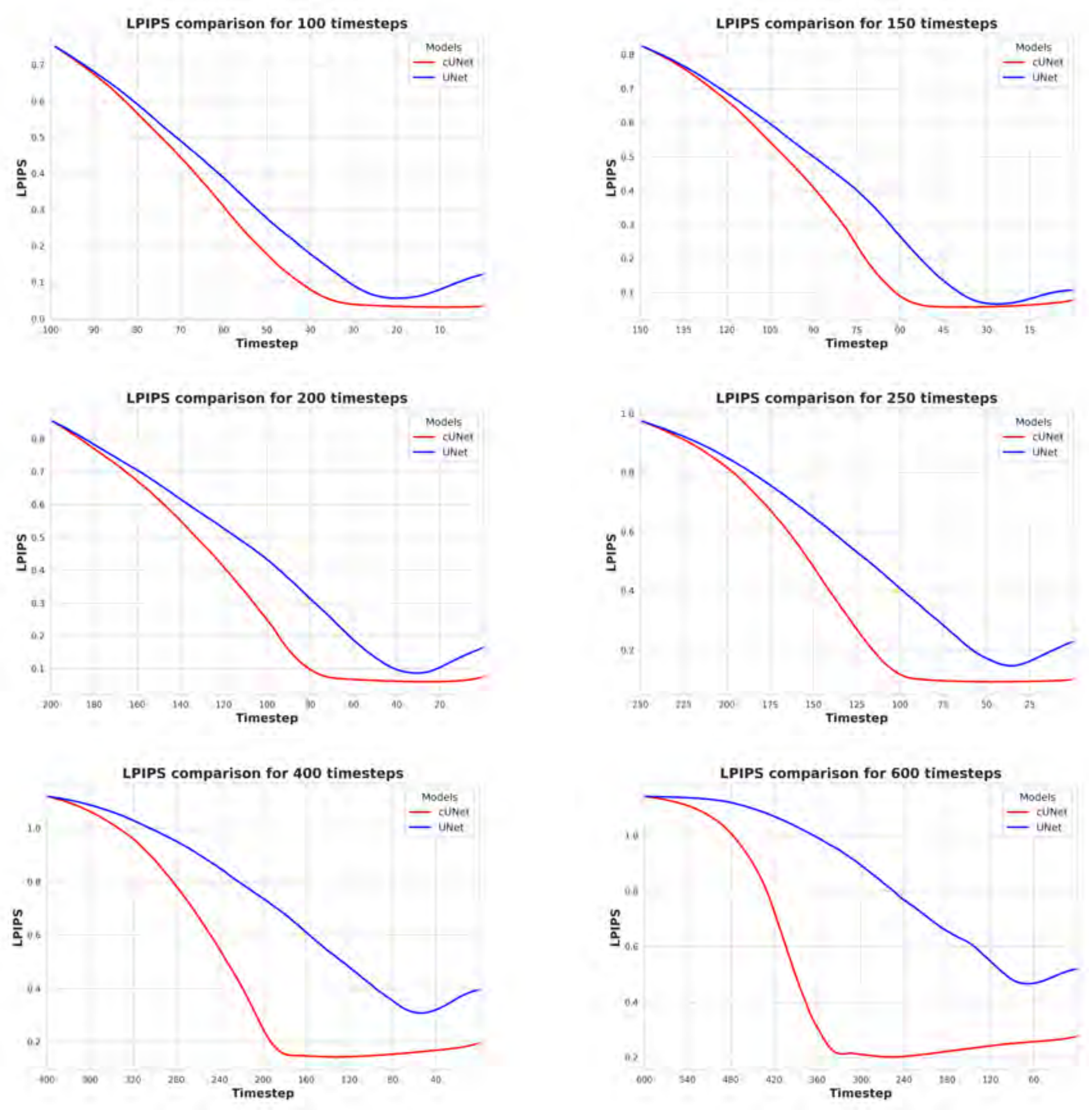


Fig. 4.14 LPIPS score versus diffusion - or denoising timesteps for one image. We can observe how our continuous U-Net consistently achieves better LPIPS score and does not suffer from such a significant *elbow effect* observed in the U-Net model in which the quality of the predictions starts deteriorating after the model achieves peak performance. The discontinuous lines indicate the timestep at which the peak LPIPS was achieved. Here, we see that, especially for a large number of timesteps, continuous U-Net seems to converge to a better solution in fewer steps. This is because of the ability it has of predicting facial features rather than simply settling for over-smoothed results (Figure A.3).

We argue that generative models strive to produce outputs that not only follow the statistical properties of the training data but are also perceptually similar. As well, in denoising, the objective is to remove noise while preserving the perceptual quality of the image. Here, the

Table 4.3 Comparison of average performance for U-Net (left) and cU-Net (right) at different noise levels in terms of the specific timestep at which peak performance was attained and time taken. These results are average across all the samples in our test set.

Noise Steps	Best SSIM Step	Time SSIM (s)	Best LPIPS Step	Time LPIPS (s)
50	47 / **39**	5.45 / **4.40**	41 / **39**	4.71 / **4.40**
100	93 / **73**	19.72 / **9.89**	78 / **72**	16.54 / **9.69**
150	140 / **103**	29.69 / **14.27**	119 / **102**	25.18 / **13.88**
200	186 / **130**	39.51 / **18.16**	161 / **128**	34.09 / **17.82**
250	232 / **154**	49.14 / **21.59**	203 / **152**	43.15 / **21.22**
400	368 / **217**	77.33 / **29.60**	332 / **212**	69.77 / **29.19**
600	548 / **265**	114.90 / **35.75**	507 / **263**	106.42 / **35.49**
800	731 / **284**	153.38 / **39.11**	668 / **284**	140.26 / **39.05**

Table 4.4 Comparative average performance of various denoising methods at select noise levels across the test set. Results demonstrate the capability of diffusion-based models (Diff U-Net and Diff cU-Net) in handling a broad spectrum of noise levels without retraining.

	50 Timesteps		**150 Timesteps**		**400 Timesteps**	
Method	**SSIM**	**LPIPS**	**SSIM**	**LPIPS**	**SSIM**	**LPIPS**
BM3D	0.74	0.062	0.26	0.624	0.06	0.977
Conv AE	0.89	0.030	0.80	0.072	0.52	0.204
DnCNN	0.89	0.026	**0.81**	0.051	0.53	0.227
Diff U-Net	0.88	0.025	0.79	0.063	**0.58**	0.184
Diff cU-Net	**0.90**	**0.019**	0.78	**0.050**	0.44	**0.146**

hand-crafted and relatively simplistic nature of SSIM and PSNR can fail to capture subtle aspects of perceptual quality.

4.4 Efficiency

One of the main challenges in the development of deep learning models is the excessive need for computational resources, primarily due to the large number of parameters involved. This issue becomes especially pronounced in the context of high-dimensional. In the specific case of diffusion models, we can see how in, for instance, Stable Diffusion (Rombach et al., 2022) - a state-of-the-art text-to-image model based on a diffusion model - the denoising U-Net used in its reverse process takes up $\sim 90\%$ of the total parameters (i.e. 860M out of 983M parameters). Such a substantial number of parameters often restricts the training and usability of these models to high-end servers or cloud environments, thereby limiting their accessibility and practical utility.

Our proposed continuous U-Net (cUNet) based framework significantly mitigates this problem by offering a much more parameter-efficient alternative. As highlighted in Figure 4.15, the cUNet model uses only around 8.8M parameters - approximately one-fourth of the total parameters of an equivalent conventional UNet. To make this comparison fair, we ensured to use equivalent architectures for each of the experiments. This means that the number of downsampling and upsampling blocks, as well as the number of linear layers for the time embeddings, were kept the same for both architectures. What makes this even more compelling is that our model achieves this reduction without any notable compromise in the model's performance, as reflected in previous subsections. On the contrary, it provides comparable results in image synthesis and arguably outperforms the U-Net in denoising tasks.

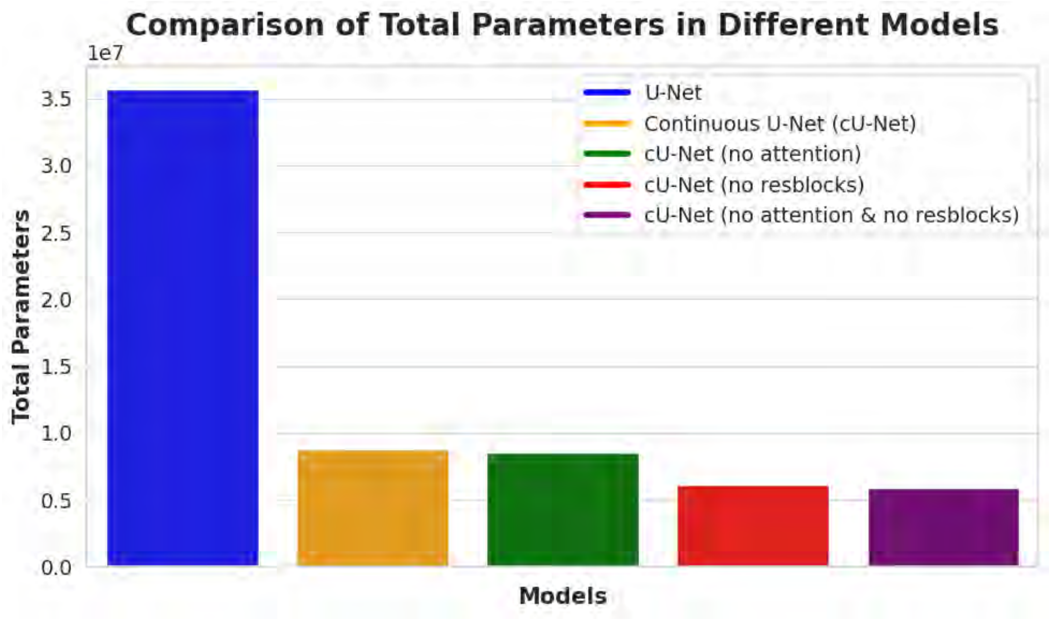


Fig. 4.15 Bar chart illustrating the total number of parameters in different variations of U-Net and Continuous U-Net (cU-Net) models. The models include the original U-Net, the cU-Net, and three variations of the cU-Net with different components removed: attention, resblocks, and both attention and resblocks. Note that the y-axis is on the 10^7 scale.

It is important to highlight that, while the focus of our work has been on DDPMs, the modular nature of our continuous U-Net framework opens the possibility for its extension to more sophisticated diffusion models. For instance, Score-Based Generative Models (Song and Ermon, 2020), which extend the ideas of DDPMs by estimating score functions directly, may leverage our approach. Similarly, the Noise Conditional Score Networks (NCSNs) (Song et al., 2021), which enhance score-based generative models by considering the variance of the noise in the score estimation, should also benefit from the increased efficiency and performance of our continuous architecture. The applicability of our approach to such models is premised on the fact that these more advanced diffusion models still utilize U-Net-type architectures for approximating the transition probability in the reverse process. The introduction of the continuous U-Net framework, therefore, could provide an opportunity to further improve the performance of these models while preserving, and possibly reducing the computational complexity. Furthermore, as demonstrated for DDPMs, the continuous nature of our framework facilitates a more accurate and smooth approximation of the transition probability, which could potentially result in enhanced image synthesis and denoising results across a wider range of diffusion models.

This superior parameter efficiency of the cUNet model coupled with the low FLOPs and reduced storage in memory (Table 4.5), have profound implications. For instance, it drastically reduces the computational requirements of the model. This allows our frameworks to be deployed in environments with more limited resources, such as personal computers or low-cost cloud environments. Secondly, the reduced parameter count significantly impacts the model's training time in some devices. By requiring fewer parameters, the model can potentially be trained more rapidly, thus allowing more efficient iterations during the development phase.

Table 4.5 Number of GigaFLOPS (GFLOPS) and Megabytes in Memory (MB) for Different Models.

Model	**GFLOPS**	**MB**
UNet	7.21	545.5
Continuous UNet (cUNet)	**2.90**	**137.9**
cUNet (no attention)	2.81	128.7
cUNet (no resblocks)	1.71	92.0
cUNet (no attention & no resblocks)	1.62	88.4

4.5 Ablation Studies

In this section, we conduct ablation studies to analyse the performance contributions of our model's components. Specifically, we assess the impact of our custom ODE blocks with residual connections, time embeddings, and attention blocks. Our goal is to understand how each element affects the results we observed in the previous sections. We do this by training our final model architecture each time without a specific element, this leads to

3 additional models: continuous U-Net (cUNet) without attention (-attn), cUNet without residual connections (-res), and a basic version of cUNet which does not include either.

In the context of image synthesis, the methodology described in Section 4.2 was kept consistent. A dataset of 30,000 generated samples was prepared for each model variant and used to compute the FID (see Table 4.6). This methodology allowed for a clear determination of the optimal sampling steps for each model, parallel to the approach presented in Figure 4.5. Moreover, the average generation time for each model over 100 samples was also evaluated across the respective datasets.

Table 4.6 Ablation study examining the impact of various components on model performance. The table provides FID scores, optimal sampling steps, and average generation time over 100 samples for each model configuration. There is a moderate impact of the attention mechanism and the pivotal role of residual connections in the generation quality.

Network	**MNIST**			**CelebA**			**LSUN Church**		
	FID	**Steps**	**Time (s)**	**FID**	**Steps**	**Time (s)**	**FID**	**Steps**	**Time (s)**
U-Net	3.61	30	3.56	19.75	100	12.48	12.28	100	12.14
cUNet	**2.98**	**5**	0.54	**21.44**	**80**	7.36	**12.14**	90	8.33
-attn	3.25	5	0.52	23.10	80	7.29	13.10	90	8.24
-res	3.40	5	0.41	34.20	90	7.04	33.12	**80**	7.21
basic	6.24	15	0.33	75.00	100	6.88	95.31	100	6.67

From Table 4.6, we see that, when attention is removed from our cUNet, there is a slight deterioration in the FID score, and generation times remain comparable. This indicates that, while the attention mechanism helps produce higher-quality samples, its absence does not drastically impair the model's generation performance. On the other hand, the omission of residual connections has more pronounced effects, resulting in a substantial increase in FID, especially evident in the more complex datasets. This suggests that residual connections play an important role in optimising the model's performance. From the results, it is also possible to see that the basic version of our model, which lacks most of the enhancements, considerably underperforms. Given the poor performance of the basic cUNet, we decided to not proceed with it in further experiments. We observe elevated FID scores across datasets indicating a drop in image quality. We have included representative examples of the generations of each of the models in Appendix B.

In our denoising experiments, we once again relied on the same test dataset and employed varying noise levels linked to the different timesteps applied by the forward process. Table 4.7 shows that although there are minimal differences in the metrics among the models across noise levels, our model consistently achieves superior results in terms of quality metrics.

This trend is further confirmed by Tables 4.8 and 4.9, which demonstrate that our full model converges in fewer timesteps across all noise levels. Just as a note, on some occasions, the time taken to converge is slightly lower for the cUNet without residual connections simply due to its reduced complexity, this, as usual with deep learning is a tradeoff for the quality of the samples.

Table 4.7 Comparison of average denoising performance between U-Net and cU-Net variants at different noise levels across the test set.

Noising Timesteps	Best SSIM Value	Best LPIPS Value
50	**0.90** / 0.88 / 0.87	**0.019** / 0.025 / 0.031
100	**0.83** / 0.83 / 0.82 /	**0.038** / 0.039 / 0.038
150	**0.78** / 0.76 / 0.73 /	**0.050** / 0.052 / 0.053
200	**0.71** / 0.69 / 0.58 /	**0.069** / 0.071 / 0.072
250	**0.64** / 0.63 / 0.56 /	**0.084** / 0.086 / 0.087
400	**0.44** / 0.44 / 0.44 /	**0.146** / 0.149 / 0.155
600	**0.26** / 0.24 / 0.23 /	**0.238** / 0.266 / 0.275
800	**0.18** / 0.12 / 0.10 /	**0.315** / 0.396 / 0.436

Table 4.8 Average denoising performance of the cUNet, cUNet without attention, and cUNet without residual connections based on SSIM metric in the test dataset.

Noising Timesteps	Best SSIM Step	Time SSIM (s)
50	**39** / 42 / 48	4.40 / 4.50 / **4.35**
100	**73** / 74 / 78	9.89 / 10.00 / **9.85**
150	**103** / 106 / 113	14.27 / 14.40 / **14.20**
200	**130** / 135 / 145	18.16 / 18.30 / **18.10**
250	**154** / 161 / 173	21.59 / 21.75 / **21.50**
400	**217** / 222 / 232	29.60 / 29.80 / **29.55**
600	**265** / 267 / 339	**35.75** / 35.90 / 36.05
800	**284** / 291 / 452	**39.11** / 39.30 / 39.45

Results suggest that our full model, augmented by attention and residual connections, exhibits superior performance. However, the small difference in denoising capabilities across different network configurations indicates the foundational strength of the continuous U-Net architecture itself. By representing dynamics in higher dimensions, the continuous U-Net inherently understands trajectories, ensuring a more robust model. As discussed in Section 2.3.2, in contrast to convolutional networks, which exhibit stability concerns due to their unconstrained operation, continuous U-Net is bounded by the constraint of the ODE integral curves not intersecting, ensuring its robustness. This stability, paired with noiseless operation and in-built robustness, is likely the dominant factor in its overall better performance

Table 4.9 Average denoising performance of the cUNet, cUNet without attention, and cUNet without residual connections based on the LPIPS metric in the test dataset.

Noising Timesteps	Best LPIPS Step	Time LPIPS (s)
50	**39** / 42 / 48	4.40 / 4.50 / **4.35**
100	**72** / 72 / 76	9.69 / 9.80 / **9.65**
150	**102** / 104 / 109	13.88 / 14.00 / **13.85**
200	**128** / 133 / 140	17.82 / 18.00 / **17.75**
250	**152** / 159 / 165	21.22 / 21.40 / **21.15**
400	**212** / 219 / 227	29.19 / 29.40 / **29.15**
600	**263** / 265 / 297	**35.49** / 35.65 / 35.80
800	**284** / 290 / 381	**39.05** / 39.25 / 39.40

than other discrete U-Net alternatives. It is essential to note that standard diffusion models already integrate features like attention and residual connections. Therefore, while these mechanisms enhance our proposed architecture, they should not be misunderstood as the primary drivers. The architecture's efficacy emerges from the combination of its core design and these embedded features.

4.6 Limitations

In order to make a fair comparison, we carry out the denoising experiments at inference time running on CPUs. This is connected to the inherent issue of parallelisation of ODE solvers. Convolutional operations are inherently parallelisable since the computations for different filters and different positions in the input are independent of each other. Modern GPUs are optimised for these types of operations, and at the same time, deep learning frameworks have heavily optimised CUDA implementations for convolution operations.

On the other hand, the challenge with ODE solvers arises from their iterative nature. Each step in some solvers can depend on the result from the previous step, which inherently makes certain ODE-solving methods sequential. However, it is worth noting that while individual solver steps might be sequential, multiple ODEs or multiple initial conditions can be solved in parallel. There is ongoing research to develop parallel-in-time methods for ODEs and PDEs. These methods aim to exploit parallelism across the temporal dimension of the problem, breaking the inherent sequential nature of some solvers. An already existing example is the work by Lienen and Günnemann (2023), which provides *torchode*, a package that tracks each ODE's progress separately and is carefully optimised for GPUs and compatibility with PyTorch's Just-In-Time (JIT) compiler. They guarantee a practical acceleration by allowing for parallel solving, and JIT compilation, and by avoiding expensive operations such as conditionals evaluated on the host that require a CPU-GPU synchronisation as much as possible and seek to minimise the number of PyTorch kernels launched.

Therefore, due to the not so spread parallelisation of ODE solvers, we conducted our inference time comparisons on CPUs. Although continuous U-Net has theoretical speed benefits, its performance is limited by current implementations. However, as indicated in our future work section, we have clear pathways to addressing this. Using the *torchode* package, which is optimized for GPU, we aim to significantly improve computation speeds. Based on Lienen and Günnemann (2023), we anticipate up to a 4x reduction in computation times, better aligning continuous U-Net's practical performance with its theoretical promise.

Lastly, the other limitation we encountered was instability issues inherent to neural ODEs. Specifically, when training on the CelebA 64x64 dataset, occasional artifacts were observed in the background of the generated images. These anomalies are suspected to have influenced the FID score, potentially leading to a few points increase, which indicates a decrease in image quality. Potential solutions to mitigate this instability include employing regularisation techniques tailored for neural ODEs such as Jacobian or depth regularisation, adjusting the architecture to be more robust to such perturbations, or exploring advanced solver methods that are better equipped to handle the intricacies of this particular problem domain.

Chapter 5

Conclusion

In this work, we have presented the first study into the scalability of continuous U-Net architectures, shedding light on its potential and limitations. We successfully introduced the first-ever continuous U-Net-based model that benefits from the incorporation of attention mechanisms, residual connections, and time embeddings specifically designed to monitor diffusion timesteps. Through our section on ablation studies, we demonstrated the benefits of these new components, in terms of denoising performance and image generation capabilities.

We propose and prove the viability of a new framework for denoising diffusion probabilistic models in which we fundamentally replace the undisputed U-Net denoiser in the reverse process with a continuous U-Net alternative. This modification is not only theoretically motivated (Section 2.3.2) but is substantiated by empirical comparison. We compared the two frameworks on image synthesis, to analyse their expressivity and capacity to learn complex distributions, and denoising in order to get insights into what happens during the reverse process at inference and training. For image synthesis, while both models achieved similar FID scores, the inherent convergence properties of our continuous U-Net allowed the model to achieve optimal FID scores with fewer sampling timesteps for generation than the U-Net alternative. Furthermore, denoising experiments revealed that while U-Net edged our model on metrics like SSIM, visually, our model produced more realistic, less distorted and not over-smoothed samples. As well, we benchmarked the denoising capabilities of the two models to other commonly used baselines for denoising proving their ability to handle a wide range of noise levels with good performance and without retraining.

Additionally, we highlighted the substantial efficiency gains of our framework. With far fewer parameters and floating point operations, and lower memory footprint and inference times, our model promises greatly improved scalability over conventional diffusion models. The substantial efficiency gains of our framework have profound implications for the democratisation and deployment of diffusion models. The reduced computational and memory demands of our model potentially unlock deployment on smaller, resource-constrained devices. This not only broadens accessibility but also provides the ability to train these models on less powerful hardware, promoting inclusive research and reducing the environmental toll.

Lastly, certain limitations remain to be addressed, the most important of these being the parallelisation of the ODE solver. We discuss them in detail in Section 4.6 and we propose potential solutions that remain part of our ideas for future work.

5.1 Future Work

While the findings of our research are promising and have opened up new possibilities, we acknowledge that there are several avenues that remain unexplored and are subjects of our upcoming investigations. Firstly, despite the inherent time cost of neural Stochastic Differential Equations (SDEs), we intend to explore their role to replace neural ODEs in continuous U-Net in order to enable uncertainty-guided sampling, wherein the rejection of high-uncertainty solutions may optimise model convergence in terms of the required timesteps.

Next, as mentioned before, in light of the recent advancements in ODE solver packages that feature Just-In-Time (JIT) compatibility and GPU batch parallelization, we plan to transition from our existing differential equations solver to maximise time performance during training and inference. This could better exploit our framework's efficiency, particularly for large-scale models.

Additionally, we intend to explore further scaling of our current models to reveal further gains from our approach as model scale increases. By adjusting our models to equate not architecture, but memory footprint or parameter count with counterparts, we aim to assess if the merits of our framework amplify and potentially overwhelmingly outperform current U-Net models.

Lastly, our assertions regarding the orthogonal nature of our solution in relation to other performance enhancers pave the way for additional explorations. Specifically, we are keen on investigating the incorporation of sampling optimisation techniques and other acceleration methods mentioned in Chapter 1 into our own model. This should allow compound performance improvements and advance state-of-the-art efficiency in diffusion models.

References

Agarap, A. F. (2019). Deep learning using rectified linear units (relu).

Bao, F., Li, C., Zhu, J., and Zhang, B. (2022). Analytic-dpm: an analytic estimate of the optimal reverse variance in diffusion probabilistic models.

Blau, Y. and Michaeli, T. (2018). The perception-distortion tradeoff. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE.

Boffi, N. M. and Vanden-Eijnden, E. (2023). Probability flow solution of the fokker-planck equation.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018a). Encoder-decoder with atrous separable convolution for semantic image segmentation.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018b). Neural ordinary differential equations. *Advances in Neural Information Processing Systems*.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2019). Neural ordinary differential equations.

Cheng, C.-W., Runkel, C., Liu, L., Chan, R. H., Schönlieb, C.-B., and Aviles-Rivero, A. I. (2023). Continuous u-net: Faster, greater and noiseless.

Cheng, W., Darnell, G., Ramachandran, S., and Crawford, L. (2020). Generalizing variational autoencoders with hierarchical empirical bayes.

Chung, H., Sim, B., Ryu, D., and Ye, J. C. (2022). Improving diffusion models for inverse problems using manifold constraints.

Coddington, A. and Levinson, N. (1955). *Theory of Ordinary Differential Equations*. International series in pure and applied mathematics. McGraw-Hill.

Dabov, K., Foi, A., Katkovnik, V., and Egiazarian, K. (2007). Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095.

Das, A. (2021). An introduction to diffusion probabilistic models. *Personal Blog*.

Dhariwal, P. and Nichol, A. (2021). Diffusion models beat gans on image synthesis.

Dupont, E., Doucet, A., and Teh, Y. W. (2019). Augmented neural odes.

Gholami, A., Keutzer, K., and Biros, G. (2019). Anode: Unconditionally accurate memory-efficient gradients for neural odes.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks.

Güngör, A., Dar, S. U., Şaban Öztürk, Korkmaz, Y., Elmas, G., Özbey, M., and Çukur, T. (2022). Adaptive diffusion priors for accelerated mri reconstruction.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2018). Gans trained by a two time-scale update rule converge to a local nash equilibrium.

Ho, J., Chan, W., Saharia, C., Whang, J., Gao, R., Gritsenko, A., Kingma, D. P., Poole, B., Norouzi, M., Fleet, D. J., and Salimans, T. (2022). Imagen video: High definition video generation with diffusion models.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models.

Ho, J., Saharia, C., Chan, W., Fleet, D. J., Norouzi, M., and Salimans, T. (2021). Cascaded diffusion models for high fidelity image generation.

Horé, A. and Ziou, D. (2010). Image quality metrics: Psnr vs. ssim. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2020). Analyzing and improving the image quality of stylegan.

Kidger, P. (2022). On neural differential equations.

Kidger, P., Morrill, J., Foster, J., and Lyons, T. (2020). Neural controlled differential equations for irregular time series.

Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions.

Kingma, D. P. and Welling, M. (2022). Auto-encoding variational bayes.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. (2021). Diffwave: A versatile diffusion model for audio synthesis.

Kreis, K., Gao, R., and Vahdat, A. (2022). Denoising diffusion-based generative modeling: Foundations and applications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Tutorial*.

Krull, A., Buchholz, T.-O., and Jug, F. (2019). Noise2void - learning denoising from single noisy images.

LeCun, Y., Cortes, C., and Burges, C. J. (2010). Mnist handwritten digit database. http://yann.lecun.com/exdb/mnist/.

Lienen, M. and Günnemann, S. (2023). torchode: A parallel ode solver for pytorch.

Liu, J., Li, C., Ren, Y., Chen, F., and Zhao, Z. (2022). Diffsinger: Singing voice synthesis via shallow diffusion mechanism.

Liu, X., Xiao, T., Si, S., Cao, Q., Kumar, S., and Hsieh, C.-J. (2019). Neural sde: Stabilizing neural ode networks with stochastic noise.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

Lucic, M., Tschannen, M., Ritter, M., Zhai, X., Bachem, O., and Gelly, S. (2019). High-fidelity image generation with fewer labels.

Lyu, Z., XU, X., Yang, C., Lin, D., and Dai, B. (2022). Accelerating diffusion models via early stop of the diffusion process.

Murphy, K. P. (2023). *Probabilistic Machine Learning: Advanced Topics*. MIT Press.

Nichol, A. and Dhariwal, P. (2021). Improved denoising diffusion probabilistic models.

Norcliffe, A., Bodnar, C., Day, B., Simidjievski, N., and Liò, P. (2020). On second order behaviour in augmented neural odes.

Preechakul, K., Chatthee, N., Wizadwongsa, S., and Suwajanakorn, S. (2022). Diffusion autoencoders: Toward a meaningful and decodable representation. pages 10609–10619.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. (2022). Hierarchical text-conditional image generation with clip latents.

Rezende, D. J. and Mohamed, S. (2015). Variational inference with normalizing flows.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation.

Rubanova, Y., Chen, R. T. Q., and Duvenaud, D. (2019). Latent odes for irregularly-sampled time series.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., Salimans, T., Ho, J., Fleet, D. J., and Norouzi, M. (2022). Photorealistic text-to-image diffusion models with deep language understanding.

Salimans, T. and Ho, J. (2022). Progressive distillation for fast sampling of diffusion models.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics.

Song, J., Meng, C., and Ermon, S. (2022). Denoising diffusion implicit models.

Song, Y. and Ermon, S. (2020). Generative modeling by estimating gradients of the data distribution.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021). Score-based generative modeling through stochastic differential equations.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going deeper with convolutions.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2023). Attention is all you need.

Weng, L. (2021). What are diffusion models? *lilianweng.github.io*.

Xu, X., Li, M., Sun, W., and Yang, M.-H. (2020). Learning spatial and spatio-temporal pixel aggregations for image and video denoising. *IEEE Transactions on Image Processing*, 29:7153–7165.

Yan, H., Du, J., Tan, V. Y. F., and Feng, J. (2022). On robustness of neural ordinary differential equations.

Zhang, K., Zuo, W., Chen, Y., Meng, D., and Zhang, L. (2017). Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric.

Appendix A

This appendix serves as the space where we present more in detail visual results of the denoising process for both the baseline model and our proposal.

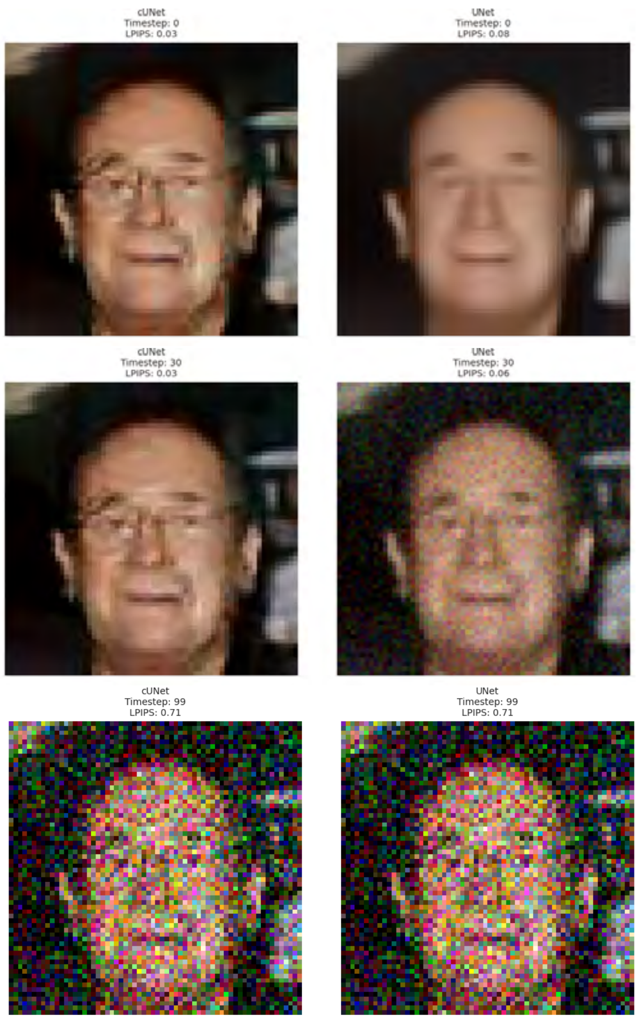


Fig. A.1 Tracking intermediate model denoising predictions. The images on the left are the outputs of our continuous U-Net, and the ones on the right are from the conventional U-Net.

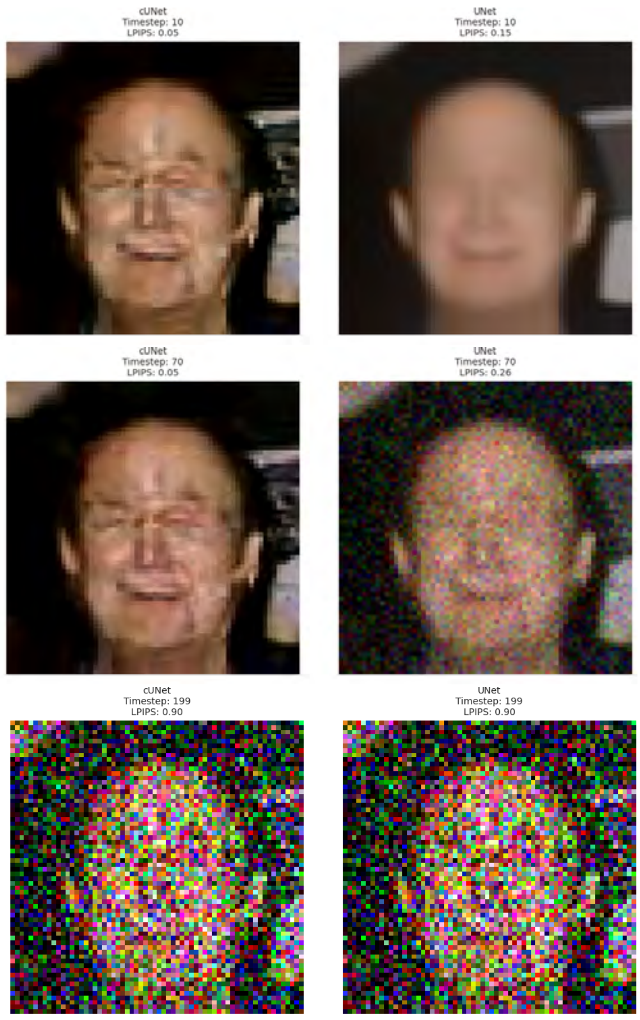


Fig. A.2 Tracking intermediate model denoising predictions: The images on the left depict the outputs of our continuous U-Net, which successfully removes noise in fewer steps compared to its counterpart (middle images) and maintains more fine-grained detail. The images on the right represent outputs from the conventional U-Net.

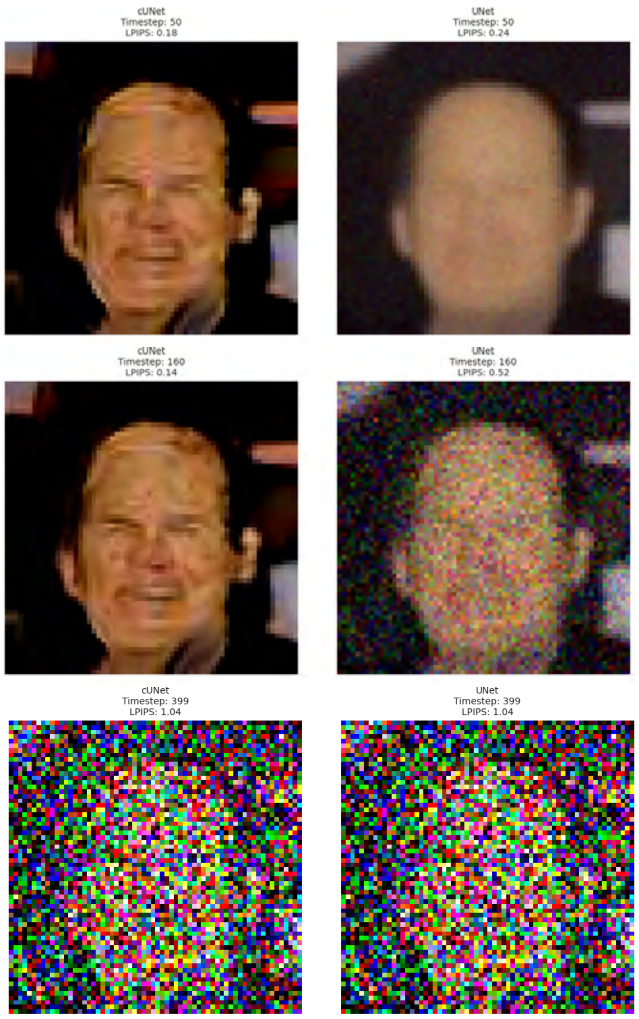


Fig. A.3 Tracking intermediate model denoising predictions: The images on the left depict the outputs of our continuous U-Net, which attempts to predict the facial features amidst the noise. In contrast, the images on the right, from the conventional U-Net, struggle to recognise the face, showcasing its limitations in detailed feature reconstruction.

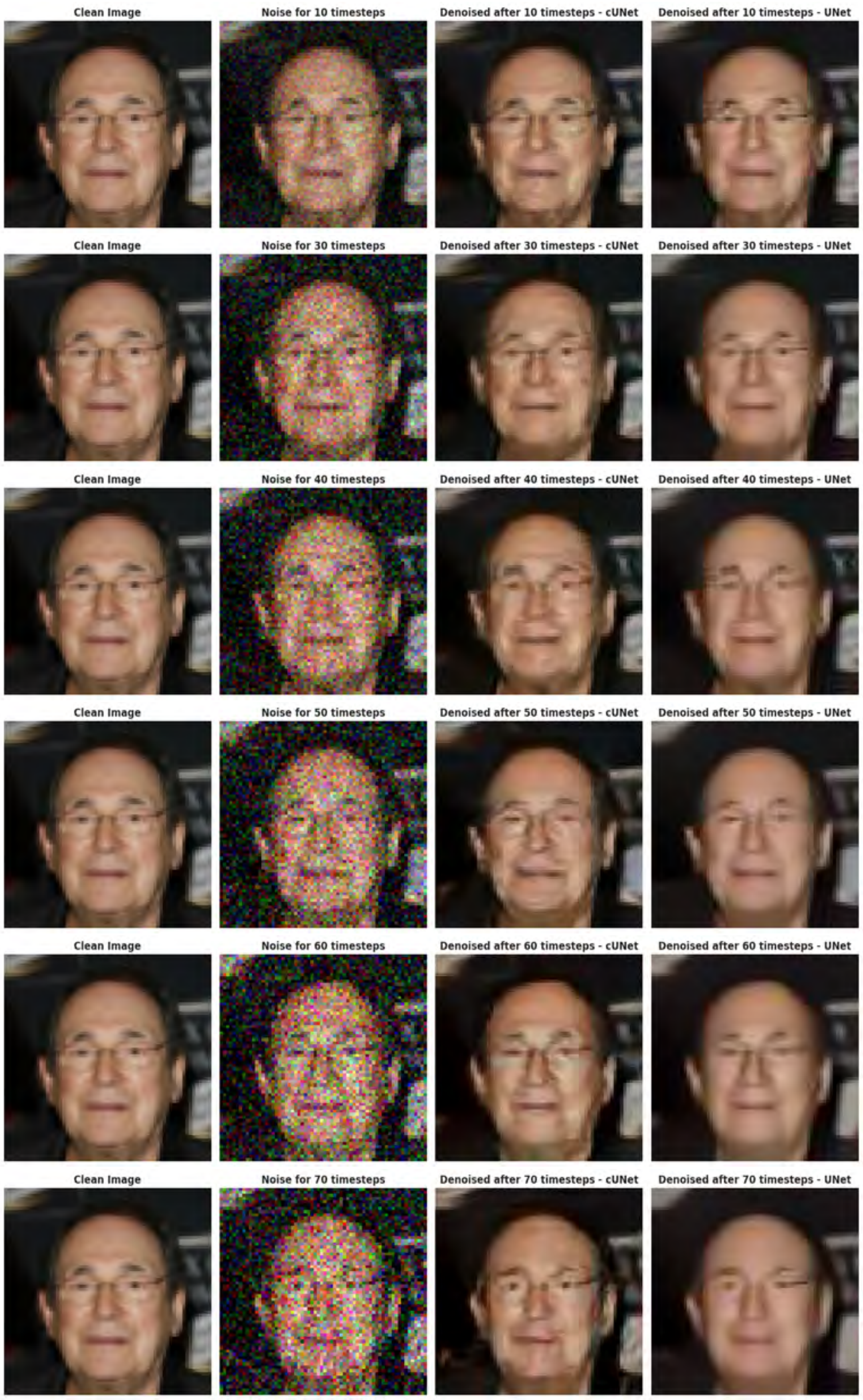

Clean Image
Noise for 10 timesteps
Denoised after 10 timesteps - cUNet
Denoised after 10 timesteps - UNet
Clean Image
Noise for 30 timesteps
Denoised after 30 timesteps - cUNet
Denoised after 30 timesteps - UNet
Clean Image
Noise for 40 timesteps
Denoised after 40 timesteps - cUNet
Denoised after 40 timesteps - UNet
Clean Image
Noise for 50 timesteps
Denoised after 50 timesteps - cUNet
Denoised after 50 timesteps - UNet
Clean Image
Noise for 60 timesteps
Denoised after 60 timesteps - cUNet
Denoised after 60 timesteps - UNet
Clean Image
Noise for 70 timesteps
Denoised after 70 timesteps - cUNet
Denoised after 70 timesteps - UNet

Fig. A.4 Original image (left), original image after adding Gaussian noise (second), and denoised image after conditioning our continuous U-Net model on the noisy image (third and fourth). We can easily appreciate how our continuous U-Net is able to recover the details of the image. As we increase the noise, U-Net is unable to recover the glasses.

Appendix B

In this short appendix, we showcase images generated for our ablation studies (Section 4.5).

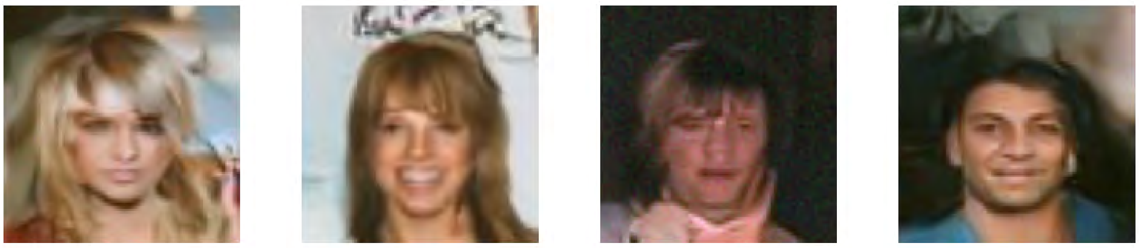


Fig. B.1 Representative samples from the version of our model trained without attention mechanism. The decrease in quality can often be appreciated in the images generated.

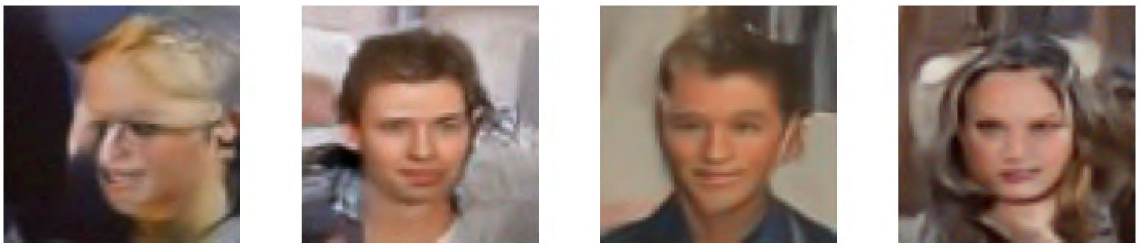


Fig. B.2 Representative samples from the version of our model trained without residual connections within our ODE block. We can see artifacts and inconsistencies frequently.

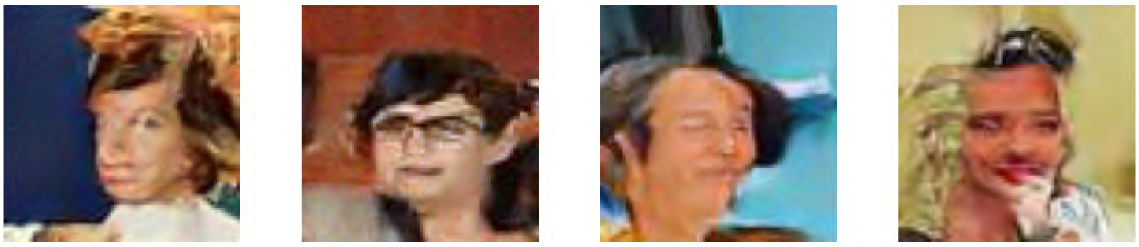


Fig. B.3 Representative samples from the basic version of our model which only includes time embeddings. As one can appreciate, sample quality suffers considerably.