Optimizing R2U-Net for Skin Cancer Segmentation with Attention Gates and Hybrid Loss

Sreejita Das

Advisor: **Dr. Kaustuv Nag**

Department of Computer Science and Engineering
Indian Institute of Information Technology Guwahati

This report is submitted for the course of
CS300 : Project-I

IIIT Guwahati March 2025

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this report are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this or any other university. This report is my own work and contains nothing that is the outcome of work done in collaboration with others except as specified in the text and Acknowledgements.

Sreejita Das
Roll: 2201201,
3rd year, Bachelors of Technology,
Department of Computer Science and Engineering,
Indian Institute of Information Technology Guwahati.

Acknowledgements

I express my sincere gratitude to Dr. Kaustuv Nag for their expert guidance and steadfast support throughout this project. I am also thankful to the faculty of the Department of Computer Science and Engineering at the Indian Institute of Information Technology Guwahati for their valuable mentorship. Additionally, I appreciate the support of my peers during this academic endeavor. Lastly, I extend my gratitude to my parents and family for their consistent encouragement and support throughout my studies.

Abstract

This report explores the effects of incorporating additional prompts on image quality in the context of ControlNet, a neural network methodology for text-to-image models. The study examines the influence of two additional prompts: positive sentiment and negative sentiment - alongside the primary image input. The investigation evaluates five pairs of positive and negative prompts, each conveying varying degrees of emphasis. Generated images with and without prompts are analyzed quantitatively using Structural Similarity Index Measure (SSIM) scores. This study provides insights into the impact of prompt variation on image generation outcomes, offering a foundation for future advancements in prompt-based image synthesis methodologies.

Table of Contents

List of Figures

Chapter 1

Introduction

Creating images from textual input has been a difficult challenge in various fields for a long time. However, ControlNet [5] has developed an end-to-end neural network methodology that enhances text-to-image diffusion models. It achieves this by preserving the quality of large pre-trained models by locking their parameters. Additionally, ControlNet introduces a trainable copy of encoding layers while treating the original as a robust backbone for learning diverse conditional controls. By connecting the trainable copy with zero convolution layers, which progressively grow during training, ControlNet can prevent harmful noise introduction. ControlNet has been tested with various conditioning inputs, including Canny edges and segmentation maps, and has demonstrated robustness and scalability. Even in tasks like depth-to-image conditioning, ControlNet has achieved competitive results.

The ControlNet takes an image as input, such as the canny edge of the image, along with three additional inputs: a text prompt that describes the content of the desired output image, a positive prompt, and a negative prompt. For instance, a positive prompt could be *"best quality, extremely detailed,"* while a negative prompt could be *"low quality, bad anatomy."* ControlNet aims to enhance the output image quality by generating images that follow the positive prompt and do not follow the negative prompt. Investigating the effect of these additional prompts in generating quality image output is crucial for two reasons. First, it can help us to determine better additional prompts. Second, it can aid in investigating a method of improving the quality of additional prompts for ControlNet in a latent space for improved performance. Thus, determining the effect of the additional prompts in generating quality image output sets the premise for this investigation. Therefore, in this work, we inspect the effect of these additional prompts in generating quality image output.

1.1 Objective

This study attempts to understand how adding prompts can impact the quality of generated images using ControlNet, a neural network methodology for text-to-image models. Thus, the study has two objectives: first, to identify effective prompts that can improve image generation quality, and second, to explore ways to enhance prompt quality within a latent space. By examining the influence of prompts on image generation outcomes, this research aims to pave the way for future advancements in text-to-image synthesis. The study has made progress towards achieving its first objective, and the report on the same is available. We plan to focus on achieving the second objective moving forward.

1.2 Contribution

Our study examines five pairs of positive and five pairs of negative prompts. Each prompt pertains to the same input but with varying degrees of emphasis. For example, positive prompts like *"good"* and *"excellent"* represent different levels of positivity, with *"excellent"* being stronger. To create additional prompts, we generate two images for an image's input canny edge: one without any prompt and the other with either a positive or negative prompt. Then, we compute the Structural Similarity Index Measure (SSIM) [] between these two images. We repeat this process for ten input canny edge images, each described by a one-word prompt. Finally, we analyze the effect of these additional prompts on ControlNet by examining the average SSIM scores.

Chapter 2

Related Works

Probabilistic models face the major challenge of balancing two conflicting objectives: achieving computational tractability while maintaining model flexibility [4]. Analytically tractable models can be easily evaluated and fit to data, but they fall short in describing complex structures in large datasets. The authors in [4] proposed a new approach that offers extreme flexibility in model structure, enabling easy multiplication with other distributions and cheap evaluation of the model log-likelihood.

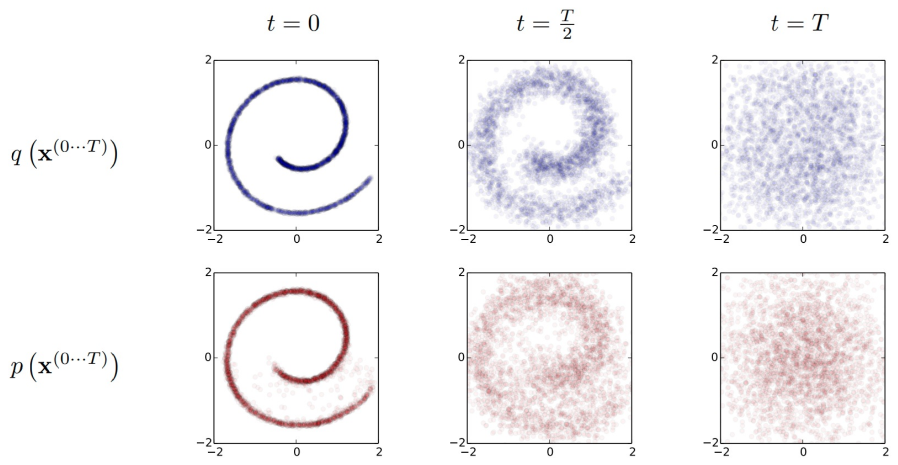

Fig. 2.1 This framework is trained on 2-d Swiss roll data. The top row shows time slices from the forward trajectory $q\left(x^{(0\dots T)}\right)$.. The data distribution (left) undergoes Gaussian diffusion, which gradually transforms it into an identity-covariance Gaussian (right). $p\left(x^{(0\dots T)}\right)$. An identity-covariance Gaussian (right) undergoes a Gaussian diffusion process with learned mean and covariance functions and is gradually transformed back into the data distribution (left).

Let $q(\boldsymbol{x}^{(0)})$ Represent the initial data distribution and $\boldsymbol{y}$ denote the variable i.e. gradually transformed through the diffusion process. We also assume that $(T_\pi(\boldsymbol{y}|\boldsymbol{y}'; \beta)$ is a Markov diffusion kernel responsible for the transformation of $\boldsymbol{y}$. It depends on $\boldsymbol{y}'$ and the diffusion rate β. Let β control the rate of the diffusion process and t represent the current time step during diffusion. We now define a conditional distribution $q(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(t-1)})$ of data at time step t given the data at the previous time step $t-1$ that comes under the forward diffusion process. We further assume that β_t represents the diffusion rate at time step t. This rate is a measure of how the concentration of the substance changes with respect to time. Another distribution $p\left(\boldsymbol{x}^{(T)}\right)$ represents the probability density function (PDF) of a random variable $\boldsymbol{x}$ at a specific time or point in time denoted by T. The associated function $\pi(x(T))$ stands for some function or value associated with the random variable x at time T, potentially representing a prior probability or a specific function. The joint distribution $p(x^(0...T))$ is the probability density function of a sequence of random variables x observed from time 0 up to time T. The time parameter T serves as a parameter representing a specific time point or index. The conditional distribution $p(x^(t-1)|x^(t))$ is the conditional probability density function of $x(t-1)$ given $x(t)$, expressing the probability of $x(t-1)$ given $x(t)$. This notation is employed to symbolize the reverse diffusion process.

2.1 Forward Trajectory

This process converts the data distribution gradually into a well-behaved distribution $\pi(\boldsymbol{y})$ (2.1) by repeated application of a Markov diffusion kernel $T_\pi(\boldsymbol{y}|\boldsymbol{y}_0; \beta)$ for $\pi(\boldsymbol{y})$, where β is the diffusion rate:

$$\pi(\boldsymbol{y}) = \int T_\pi(\boldsymbol{y}|\boldsymbol{y}', \beta)\pi(\boldsymbol{y}')d\boldsymbol{y}' \tag{2.1}$$

$$q\left(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(t-1)}\right) = T_\pi\left(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(t-1)}, \beta_t\right)$$

$$q\left(\boldsymbol{x}^{(0\ldots T)}\right) = q\left(\boldsymbol{x}^{(0)}\right)\prod_{t=1}^{T} q\left(\boldsymbol{x}^{(t)} \mid \boldsymbol{x}^{(t-1)}\right) \tag{2.2}$$

2.2 Reverse Trajectory

The generative distribution will be trained to describe the same trajectory but in reverse eq (2.3):

$$\begin{aligned} p\left(\boldsymbol{x}^{(T)}\right) &= \boldsymbol{\pi}\left(\boldsymbol{x}^{(T)}\right) \\ p\left(\boldsymbol{x}^{(0\dots T)}\right) &= p\left(\boldsymbol{x}^{(T)}\right)\prod_{t=1}^{T} p\left(\boldsymbol{x}^{(t-1)} \mid \boldsymbol{x}^{(t)}\right) \end{aligned} \tag{2.3}$$

Evaluate the relative probability of the forward and reverse trajectories, averaged over forward trajectories:

$$p\left(\boldsymbol{x}^{(0)}\right) = p\left(\boldsymbol{x}^{(T)}\right)\prod_{t=1}^{T} \frac{p\left(\boldsymbol{x}^{(t-1)} \mid \boldsymbol{x}^{(t)}\right)}{q\left(\boldsymbol{x}^{(t)} \mid \boldsymbol{x}^{(t-1)}\right)} \tag{2.4}$$

The primary goal of training is to identify the reverse Markov transitions that maximize the log-likelihood lower bound. This is employed for tasks like denoising and inpainting of natural images. Derive the upper and lower bounds on the conditional entropy of each step in the reverse trajectory:

$$\begin{aligned} & H_q\left(X^{(t)} \mid X^{(t-1)}\right) + H_q\left(X^{(t-1)} \mid X^{(0)}\right) - H_q\left(X^{(t)} \mid X^{(0)}\right) \\ \leq\ & H_q\left(X^{(t-1)} \mid X^{(t)}\right) \\ \leq\ & H_q\left(X^{(t)} \mid X^{(t-1)}\right) \end{aligned} \tag{2.5}$$

This methodology offers transparency and a robust mathematical foundation for understanding and harnessing probabilistic models. It enables rapid learning, sampling, and computation of conditional and posterior probabilities within deep generative models containing millions of time steps. However, while diffusion models are easily defined and efficient for training, there remains a lack of evidence demonstrating their ability to generate high-quality samples. In this study, another paper is examined[2], which introduces a technique for generating high-quality images through the utilization of diffusion models.

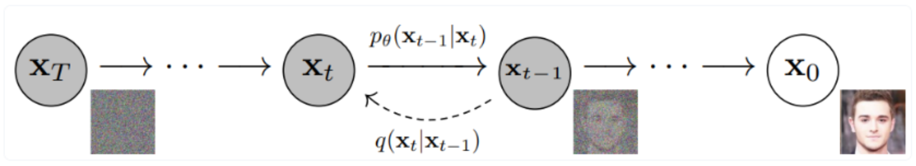
$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$\mathbf{x}_0$

Fig. 2.2 The directed graphical model is considered in this work.

2.3 Working Principle

The diffusion model's basic idea is to create samples that, after a predetermined amount of time, closely resemble the given data. A parameterized Markov chain that was trained using variational inference is used by the model. The transitions in this chain are designed to reverse a diffusion process, in which noise is systematically added to the data by a Markov chain in the reverse direction of sampling until the signal is eliminated. The model achieves simplicity that is consistent with a particular neural network parameterization by using small amounts of Gaussian noise and setting sampling chain transitions to conditional Gaussians. This parameterization is to be equivalent to annealed Langevin dynamics during sampling and denoising score matching across different noise levels during training. This simplified parameterization leads to optimal sample quality.

2.4 Notations

Let $p_\theta\left(\boldsymbol{x}_0\right)$ is the "Probability Density Function" for initial point $\boldsymbol{x}_0$ and $p_\theta\left(\boldsymbol{x}_{0:T}\right)$ is a Joint distribution that is a parameterized Markov chain with learned Gaussian transitions. They take as input the current point x_t and possibly the time t. Neural network for mean $\boldsymbol{\mu}_t$ is $\boldsymbol{\mu}_\theta\left(x_t, t\right)$ and for covariance $\boldsymbol{\Sigma}$ is $\boldsymbol{\Sigma}_\theta\left(x_t, t\right)$. Here $p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \mathbf{0}, \boldsymbol{I})$. Here $\mathcal{N}$ represents the standard normal distribution where the mean is 0 and the standard deviation is $\boldsymbol{I}$. Here approximate Posterior is $[q\left(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0\right)]$ and Covariance Matrix is $[\beta_t I]$. We can represent mean in this way $\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}$ also as it shows the effect of the previous state on the current state.

2.5 Reverse Process

We can train the reverse process mean function approximator μ_θ to predict $\tilde{\mu}_t$, we can train it to predict ϵ. The ϵ-prediction parameterization both resembles

Langevin dynamics and simplifies the diffusion model's variational bound to an objective that resembles denoising score matching. It is just another parameterization of $p_\theta(x_{t-1}|x_t)$, so we verify its effectiveness. We compare predicting ϵ against predicting $\tilde{\mu}_t$. From eq(2.3):

$$p_\theta\left(\boldsymbol{x}_{0:T}\right) = p_\theta\left(\boldsymbol{x}_T\right) \prod_{t=1}^{T} p\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t\right) \tag{2.6}$$

$$p_\theta\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t\right) = \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta\left(x_t, t\right), \Sigma_\theta\left(\boldsymbol{x}_t, t\right)) \tag{2.7}$$

2.6 Forward Process

The forward process variances β_t, regardless of their learnability, L_T is constant and insignificant throughout training. From eq (2.2):

$$q\left(\boldsymbol{x}_{1:T} \mid x_0\right) = \prod_{t=1}^{T} q\left(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}\right) \tag{2.8}$$

$$q\left(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}\right) = \mathcal{N}\left(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \boldsymbol{I}\right) \tag{2.9}$$

Here, training is performed by optimizing the usual variational bound on negative log-likelihood. Using the notation $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ in eq(2.9):

$$q\left(\boldsymbol{x}_t \mid x_0\right) = \mathcal{N}\left(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\boldsymbol{I}\right)$$

Further improvements come from variance reduction by rewriting L:

$$L_0 = -\log p_\theta(\boldsymbol{x}_0 \mid \boldsymbol{x}_1) \tag{2.10}$$

$$L_{t-1} = D_{KL}\left(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)\right) \tag{2.11}$$

$$L_T = D_{KL}\left(q(\boldsymbol{x}_T \mid \boldsymbol{x}_0) \parallel p(x_T)\right) \tag{2.12}$$

2.7 Notations

The negative log-likelihood at the initial step is defined as $L_0 = -\log p_\theta(x_0|x_1)$ in eq(2.10). Subsequently, the divergence between the approximate posterior and the conditional distribution is captured by $L_{t-1} = D_{\mathrm{KL}}(q(x_{t-1}|x_t, x_0) \,||\, p_\theta(x_{t-1}|x_t))$ in eq(2.11). The overall divergence for the entire diffusion process is represented as $L_T = D_{\mathrm{KL}}(q(x_T|x_0) \,||\, p(x_T))$ in eq(2.12).

2.8 Loss function

To represent the mean $\mu_\theta(x_t, t)$, we propose a specific parameterization motivated by the analysis of L_t. With $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I)$, we can write:

$$L_{t-1} = \mathbb{E}\left[\frac{1}{2\sigma_t^2} \parallel \bar{\mu}_t\left(\boldsymbol{x}_t, \boldsymbol{x}_0\right) - \mu_\theta\left(\boldsymbol{x}_t, t\right) \parallel^2\right] + C$$
$$L_{simple} = \mathbb{E}_{t,x_0,\epsilon}\left[\parallel \epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\right) \parallel^2\right]$$

2.9 Notations

The loss at time step $t-1$, denoted as L_{t-1}, is expressed as the expected squared Euclidean distance between $\bar{\mu}_t(x_t, x_0)$ and $\mu_\theta(x_t, t)$, normalized by $2\sigma_t^2$. The expectation is taken over relevant variables. The term C is an additional constant.

The simple loss, denoted as L_{simple}, is defined as the expected squared Euclidean distance between ϵ and $\epsilon_\theta\left(\sqrt{\bar{\alpha}_t x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t}\right)$, where the expectation is taken over x_0, t, and ϵ.

2.10 Training

Algorithm 1 Training Algorithm

```
1: while not converged do
2:     $x_0 \sim q(x_0)$
3:     $t \sim \text{Uniform}(1, \ldots, T)$
4:     $\epsilon \sim \mathcal{N}(0, 1)$
5:     Take gradient descent step on $\nabla_\theta \left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{a}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\right)\right\|^2$
6: end while
```

The algorithm repeatedly samples an initial value, time step, and random noise. Then, it performs a gradient descent step on a function involving the gradient with respect to parameters. This process continues until the convergence criterion is met. The specific details of the convergence criteria, the functions involved, and the meaning of the parameters like α_t and $\bar{\alpha}_t$ would need to be provided in a more detailed context or algorithmic description.

2.11 Sampling

Algorithm 2 Sampling Algorithm

```
1: $x_T \sim \mathcal{N}(0, I)$
2: for $t \leftarrow T$ to 1 do
3:     $z \sim \mathcal{N}(0, I)$ if $t > 1$, else $z = 0$
4:     $\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \boldsymbol{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} \epsilon_\theta(\boldsymbol{x}_t, t) \right) + \sigma_t \boldsymbol{z}$
5: end for
6: return $X$
```

The algorithm initializes with a sample, and then iteratively generates new samples based on a recursive formula that involves noise and previous samples. The parameters α_t and σ_t might be specified or updated at each iteration.

This methodology presents high-quality image samples through diffusion models, revealing connections with variational inference, denoising score matching, annealed Langevin dynamics, autoregressive models, and progressive lossy compression. Although diffusion models show significant inductive biases when applied to image data, there is a curiosity analyzing how effectively they perform with different types of data as well as the way they might be integrated into various generative models and machine learning systems.

In this study, another paper is examined[3], which addresses limitations in the original methodology by enhancing sampling efficiency through learned variances, achieving competitive log-likelihoods, and demonstrating seamless scalability with model capacity and training compute. These refinements collectively contribute to a more robust and versatile probabilistic model.

$$L_{\text{vlb}} = L_0 + L_1 + \ldots + L_{T-1} + L_T \tag{2.13}$$

Sampling arbitrary steps during training in a forward noising process, parameterizing the mean function $\mu_\theta\left(\boldsymbol{x}_t, t\right)$ using neural networks.

$$\mu_\theta(x_t,t) = \frac{1}{\sqrt{\alpha}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t,t)\right)$$
$$L_{simple} = \mathbb{E}_{t,x_0,\epsilon}\left[\| \epsilon - \epsilon_\theta(x_t,t) \|^2\right] \tag{2.14}$$

We define a new hybrid objective L_{hybrid} using (2.13) and (2.14).The hybrid objective achieves better log-likelihoods on the training set. With this important sampled objective, we can achieve our best log-likelihoods by optimizing L_{vlb}

$$L_{\text{hybrid}} = L_{\text{simple}} + \lambda L_{\text{vlb}} \tag{2.15}$$

We use the variational lower-bound (VLB) to modify log-likelihoods and show the competitiveness of Denoising Diffusion Probabilistic Models (DDPMs) on high-diversity datasets such as ImageNet. By using a simple reparameterization and a hybrid learning objective that combines VLB with a more straightforward method, our models outperform log-likelihoods obtained from direct optimization. Interestingly, the latter shows increased noise on gradients during training, which is reduced by applying a straightforward importance sampling strategy to achieve better log-likelihoods.

Our models remarkably attain comparable sample quality with fewer sampling steps upon integrating learned variances. In comparison, DDPM requires a large number of forward passes; this reduces the sampling procedure to as little as 50 passes, improving its practical usefulness. This accomplishment is in line with ongoing initiatives to investigate other strategies for accelerating sample procedures. Although diffusion models are already at the cutting edge, they are still inferior to GANs when it comes to difficult generation datasets. In this study, another paper is examined [1] GANs, with their adversarial architecture, achieve diverse and high-quality samples, but face a trade-off between diversity and fidelity. To bring these advantages to diffusion models, efforts focus on enhancing architecture and devising a scheme for balancing diversity and fidelity.

The referred to architecture improvements considerably increase FID. shows the tunability of a single hyperparameter for the diversity-fidelity trade-off by introducing a strategy that uses classifier gradients to control diffusion model sampling. Significant is the significant rise of the gradient scale factor without adversarial consequences. Our enhanced architecture performs exceptionally well in conditional synthesis and unconditional picture synthesis when classifier guidance is used. Interestingly, FIDs comparable to BigGAN are maintained with as few as 25 forward passes under the classifier direction. We present the combined efficacy for better outcomes on ImageNet at 256x256 and 512x512 resolutions by comparing our models to upsampling stacks.

Emphasizing the empirical superiority of the mean-squared error objective L_{simple} over the variational lower bound L_{vlb}, showcasing its practical effectiveness. This approach employs Langevin dynamics within a denoising model, demonstrating remarkable performance in generating high-quality image samples. The term "diffusion models" is used broadly to encompass both categories of models. To address limitations in scenarios with fewer diffusion steps, we propose a neural network parameterization for adjusting the variance $\Sigma_\theta(\boldsymbol{x}_t, t)$. The neural network output v is then integrated into an interpolation scheme.

$$\Sigma_\theta(\boldsymbol{x}_t, t) = \exp(v \log \beta_t + (1 - v) \log \tilde{\beta}_t) \tag{2.16}$$

2.12 Notations

$\Sigma_\theta(\boldsymbol{x}_t, t)$: Covariance matrix parameterized by θ. v: Balance parameter in the equation. β_t: Variable. $\tilde{\beta}_t$: Determines the final form of the covariance matrix.

2.13 Algorithm

Algorithm 3 Classifier-Guided Diffusion Sampling

```
Require: Diffusion model parameters $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $p_\phi(y|x_t)$, gradient scale $s$
1: function ClassifierGuidedDiffusionSampling($y$, $s$)
2:     $x_T \leftarrow$ sample from $N(0, I)$
3:     for $t \leftarrow T$ to 1 do
4:         $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$
5:         $x_{t-1} \leftarrow$ sample from $N\left(\mu + s\Sigma\nabla_{x_t} \log p_\phi(y|x_t), \Sigma\right)$
6:     end for
7:     return $x_0$
8: end function
```

Algorithm 4 Classifier-Guided DDIM Sampling

```
Require: Diffusion model $\epsilon_\theta(x_t)$, classifier $p_\phi(y|x_t)$, gradient scale $s$
1: function ClassifierGuidedDDIMSampling($y$, $s$)
2:     $x_T \leftarrow$ sample from $N(0, I)$
3:     for $t \leftarrow T$ to 1 do
4:         $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1 - \bar{\alpha}_t}\nabla_{x_t} \log p_\phi(y|x_t)$
5:         $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \left(x_t - \sqrt{\frac{\bar{\alpha}_t}{1-\bar{\alpha}_t}}\hat{\epsilon}\right) + \sqrt{1 - \bar{\alpha}_{t-1}}\hat{\epsilon}$
6:     end for
7:     return $x_0$
8: end function
```

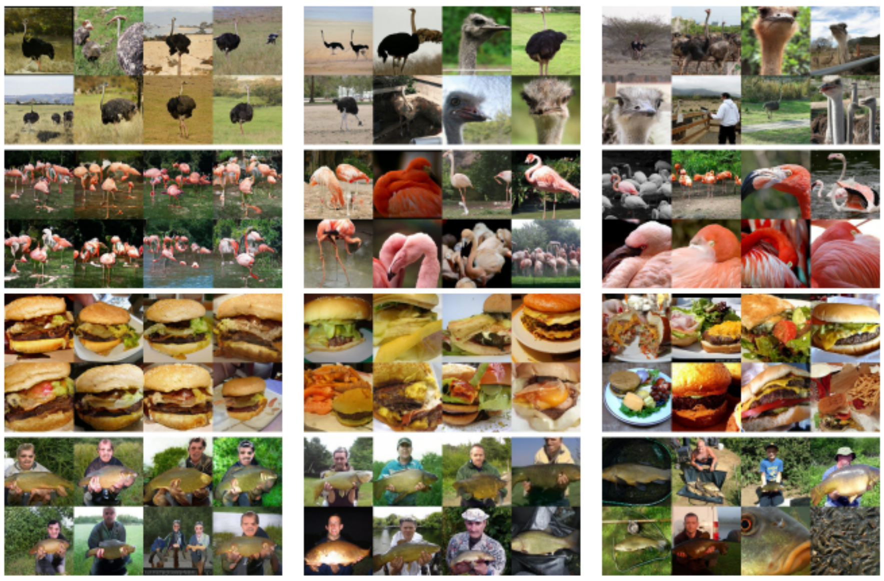

Fig. 2.3 Samples from BigGAN-deep with truncation 1.0 (FID 6.95, left) vs samples from our diffusion model with guidance (FID 4.59, middle) and samples from the training set (right).

Chapter 3

Proposed Model[5]

3.1 Finetuning Neural Networks

The standard approach to enhance a neural network is to train it again using more data. But this simple continuance may result in disastrous forgetting, mode collapse, and overfitting. A great deal of research has gone into creating solutions that are optimized to lessen these problems in order to overcome these concerns.

3.2 HyperNetwork

This mechanism involves training a small recurrent neural network to influence the weights of a larger one. HyperNetworks implementation for Stable Diffusion to alter the artistic style of the output images.

3.3 Adapter

In Natural Language Processing (NLP), methods that use adapters are often used to customize transformer models that have already been trained for different applications by adding new module layers. Adapters are used in incremental learning and domain adaption in computer vision. This method is frequently used in conjunction with CLIP to apply backbone models that have already been trained to a variety of tasks. Notably, adapters have proven successful in ViT adapters and vision transformers. T2IAdapter adjusts Stable Diffusion to external circumstances in parallel.

3.4 Additive Learning

The method uses prunes, hard attention, or learning weight masks to introduce a few new parameters and freezes the original model weights to prevent forgetting. By using a side branch model and combining its outputs with a preset schedule, Side-Tuning learns new capabilities in addition to a frozen model and an additional network.

3.5 Low-Rank Adaptation (LoRA)

This prevents catastrophic forgetting by learning the offset of parameters with low-rank matrices, based on the observation that many over- parameterized models reside in a low intrinsic dimension subspace.

3.6 Zero-Initialized Layers

References

[1] Dhariwal, P. and Nichol, A. (2021). Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794. Curran Associates, Inc. 10

[2] Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. 5

[3] Nichol, A. Q. and Dhariwal, P. (2021). Improved denoising diffusion probabilistic models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 8162–8171. PMLR. 9

[4] Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. 3

[5] Zhang, L., Rao, A., and Agrawala, M. (2023). Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3836–3847. v, 1, 14