IMPROVED IMAGE GENERATION IN NORMALIZING FLOWS THROUGH A MULTI-SCALE ARCHITECTURE AND VARIATIONAL TRAINING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

DENİZ SAYIN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

AUGUST 2022

Approval of the thesis:

IMPROVED IMAGE GENERATION IN NORMALIZING FLOWS THROUGH A MULTI-SCALE ARCHITECTURE AND VARIATIONAL TRAINING

submitted by **DENİZ SAYIN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** ________________

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** ________________

Assist. Prof. Dr. Ramazan Gökberk Cinbiş
Supervisor, **Computer Engineering, METU** ________________

Examining Committee Members:

Assoc. Prof. Dr. Sinan Kalkan
Computer Engineering, METU ________________

Assist. Prof. Dr. Ramazan Gökberk Cinbiş
Computer Engineering, METU ________________

Prof. Dr. Selim Aksoy
Computer Engineering, Bilkent University ________________

Date: 31.08.2022

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: DENİZ SAYIN

Signature :

ABSTRACT

IMPROVED IMAGE GENERATION IN NORMALIZING FLOWS THROUGH A MULTI-SCALE ARCHITECTURE AND VARIATIONAL TRAINING

SAYIN, DENİZ
M.S., Department of Computer Engineering
Supervisor: Assist. Prof. Dr. Ramazan Gökberk Cinbiş

August 2022, 44 pages

Generative models have been shown to be able to produce very high fidelity samples in natural image generation tasks in recent years, especially using generative adverserial network and denoising diffusion model based approaches. Normalizing flow models are another class of generative models, which are based on learning invertible mappings between the latent space and the image space. Normalizing flow models possess desirable features such as the ability to perform exact density estimation and simple maximum likelihood based training, which can offer theoretical guarantees. While the state-of-the-art normalizing flow models are able to produce high fidelity images on specific simple image generation tasks such as faces and bedrooms, they typically fail to produce sensible results in difficult natural image datasets containing a multitude of underlying classes. We propose an approach focused on improving natural image generation using a new normalizing flow model, in which we start by generating a small natural image and refine it step by step with conditional normalizing flow models performing 2x super-resolution. We also propose a new augmentation method at the feature level for conditional encodings to make the intermediate models in our cascade more robust against noise and artifacts coming previous levels of the cascade. This augmentation method has its roots in variational inference. We perform experiments on the CelebA and CIFAR-10 datasets, show our qualitative results and compare our generations with state-of-the-art approaches using the FID metric.

Keywords: natural image generation, normalizing flows, variational inference, generative models

ÖZ

NORMALLEŞTİRİCİ AKIM MODELLERİNDE ÇOK-ÖLÇEKLİ MİMARİ VE DEĞİŞİMSEL EĞİTİM İLE GELİŞTİRİLMİŞ RESİM ÜRETİMİ

SAYIN, DENİZ
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Ramazan Gökberk Cinbiş

Ağustos 2022 , 44 sayfa

Son yıllarda üretici modellerin doğal resim üretme görevlerinde yüksek kaliteli örnekler üretebildiği, özellikle üretici çekişmeli ağlar ve de yayınım modelleri kullanılarak gösterilmiştir. Normalleştirici akım modelleri diğer bir üretici model sınıfıdır ve resim uzayı ile saklı uzay arasında tersi olan bir fonksiyon öğrenmek üzerine kuruludurlar. Normalleştirici akım modellerinin kesin yoğunluk tahmini yapabilme ve teorik garantiler sağlayan basit en büyük olabilirlik temelli bir eğitime sahip olma gibi istenen özellikleri vardır. Fakat güncel olan en iyi normalleştirici akım modelleri insan yüzü ve yataklar gibi spesifik ve basit resimler içeren veri kümelerinde kaliteli üretici sonuçlar elde edebilmekle birlikte, tipik olarak birden fazla sınıf içeren karışık doğal resim içerikli veri kümelerinde makul sonuçlar üretememektedir. Bu tezde doğal resim içerikli veri kümelerinde üzerinde daha yüksek kaliteli örnekler üretmeyi amaçlayan yeni bir normalleştirici akım modeli öneriyoruz. Bu modelde ilk olarak basit bir normalleştirici akım modeli ile çok küçük çözünürlüklü bir resim üretip, ardından bu resmi 2x süper-çözünürlük uygulayan koşullu normalleştirici akım modelleri ile adım adım iyileştiriyoruz. Ayrıca modelimizdeki ara modelleri alt modellerden gelen resimlerde oluşabilecek hata ve gürültüye daha dayanıklı hale getirmek için öznitelik seviyesinde yeni bir veri artırma yöntemi öneriyoruz. Önerdiğimiz veri artırma yöntemi teorik köklerini değişimsel çıkarsamadan almaktadır. CelebA ve CIFAR-10 veri kümelerinde deneyler yapıp, nitel sonuçlarımızı gösteriyor ve bunları FID metriği ile güncel olan en iyi yöntemlerle karşılaştırıyoruz.

Anahtar Kelimeler: doğal resim üretme, normalleştirici akım, değişimsel çıkarımsama, üretken modeller

To my recently enlarged family

ACKNOWLEDGMENTS

I would first and foremost like to thank my supervisor Asst. Prof. Dr. Ramazan Gökberk Cinbiş for his unending supervision, support and insight which he graciously provided through ups and downs for the full span of my M.Sc. I would have been lost without his exceptional guidance and contributions and would not trade them for anything.

I would then like to thank Prof. Dr. Vittorio Ferrari for the regular advice and feedback he provided on our work through its different phases that culminated in this thesis. Some of his valuable views and guidelines will stay with me for all of my research career.

I would also like to thank my fiancée Ekin for being there for me during the most stressful parts of the writing process. Her joyful presence and continuous support made a huge difference bettering the highs and smoothing out the lows.

Finally, I would like to thank my family for always having my back and encouraging me to move forward no matter the circumstances.

This thesis was supported in part by the TUBITAK Grant 119E597. The numerical calculations reported in this paper were fully/partially performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

NF	Normalizing Flow
VAE	Variational Auto-Encoder
CV-VAE	Constant-Variance Variational Auto-Encoder
GAN	Generative Adversarial Network
IS	Inception Score
FID	Fréchet Inception Distance
KID	Kernel Inception Distance
ELBO	Evidence Lower Bound
ODE	Ordinary Differential Equation
SR	Super-Resolution
BPD	Bit-Per-Dimension
NLL	Negative Log-Likelihood
LR	Low-Resolution
HR	High-Resolution

CHAPTER 1

INTRODUCTION

Generative modeling is an important task in computer vision, where the goal is to model the underlying distribution of a dataset given only samples from it. Applications include a wide variety of tasks: image generation, text-to-image translation, super-resolutio and even artistic tasks such as style transfer.

A well-known class of generative models is Generative Adversarial Networks [1] (GANs), which are trained as a pair of competing deep neural networks using a game theory based objective. The *generator* maps latent noise vectors to image samples and is trained to fool the *discriminator*, which tries to distinguish between *real* examples from the dataset and *fake* examples created by the generator. These models are able to achieve high fidelity in a variety of image generation tasks.

Despite their sample quality, GANs suffer from a range of issues [2]. Their training is unstable, and the properties of the latent vectors are ill-defined unless guided by additional models or objective functions [3]. It is also not possible to analytically calculate the generating latent vector given a data sample.

Normalizing flows [4] (NFs) are another class of generative models, and offer strong theoretical properties. These models are bijections from the data space to the latent space, and are trained with a maximum likelihood objective aiming to transform the data distribution to a simpler prior distribution (usually Gaussian) in the latent space. This training regime is more stable compared to GANs and allows for the model to work both ways once training is complete: I. Samples can be drawn from the simple distribution in the latent space and mapped back into the data space thanks to the invertibility of the model to generate samples. II. A given real data sample can be mapped into the latent space, and its exact probability density can be estimated due to the assumption that it should come from the simple distribution in the latent space.

Despite their theoretical strengths, normalizing flows suffer from several limitations as well. Every building block of the model has to preserve the dimensionality of the input for the sake of invertibility, and cannot compress or decompress the input like standard neural networks. Also, the Jacobian of each transformation in the model needs to be tractable to be able to quickly compute the maximum likelihood objective function [5, 6]. These limit the expressivity of the model, and the fidelity of samples drawn from NF models trained on high-resolution image datasets is a far cry from the quality of samples drawn from GAN models and is open to improvement [7, 8].

1.1 Contributions

In this thesis, we propose a new normalizing flow model focused on improving the fidelity of generated samples when trained on natural images. We aim to improve upon state of the art image generation results qualitatively in the domain of normalizing flows. In particular, we use a cascaded stack of conditional super-resolution normalizing flow models optimized on different scales to iteratively refine an initial low quality image produced by an unconditional normalizing flow model. To reduce error and noise amplification in the conditional model cascade, we also propose an augmentation at the feature level for conditional inputs that has theoretical roots in variational training.

1.2 Outline

We now outline the rest of the thesis. In Chapter 2, we present an overview of image generation approaches and focus on similarities to our own work, while also presenting necessary background information from the work we base our model on. In Chapter 3, we explain the details of our own approach and the reasoning behind it based on the background we provide. In Chapter 4 we provide our basic experimental results and compare them with previous work and also perform ablation studies and analyses to draw conclusions from. In Chapter 5 we summarise the thesis and point to possible directions in which future work could be done.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we first cover related work in the literature by going over the various generative modeling approaches, specifically focusing on normalizing flows as they are of particular interest. We then cover necessary mathematical and architectural details from approaches we base our work on as background for the following chapters.

2.1 Related Work

Image generation tasks have been tackled by a variety of model classes, each having their own advantages and disadvantages relative to each other.

The most well known model class is without a doubt generative adversarial networks (GANs), which was introduced by Goodfellow et al. [1]. These models work by pairing a generator network that attempts to generate image samples from noise and a discriminator network that attempts to distinguish real data samples from generated samples. These two networks improve each other through competition and attempt to reach Nash equilibrium. GANs can generate very high fidelity images [9], but are plagued by various failure modes during training such as non-convergence [2] and mode collapse (only ever generating a single image) [10] and may fail to capture the full distribution of the data due to the lack of a likelihood objective [11]. Inception Score (IS) [10] and Fréchet Inception Distance (FID) [12] metrics that are used for measuring image generation performance have also been introduced in the context of GANs. A recent large scale survey of GANs can be found in Bermano et al. [13].

Another class of generative models are auto-regressive models such as PixelRNN [14], PixelCNN [15] and WaveNet [16]. These models generate samples piece by piece and are trained to model the probability distribution of the next piece of a sample based on the pieces that have been generated so far. Unlike GANs, they can also be used for density estimation, but usually have a slow generation process since each piece of the sample has to be generated in sequence. Auto-regressive in a broad sense can also imply that a model takes is own output as feedback, such as in the case of transformers [17]. As a result, not all models called *auto-regressive* will be generative models.

The next class of variational auto-encoders (VAEs), whose basic formulation we cover in 2.2.2, were first introduced by Kingma & Welling [18]. These models have probabilistic encoders and decoders and are optimized on a lower bound of the maximum likelihood, the evidence lower bound (ELBO). This lower bound can also be used for density estimation for comparative purposes, and their convergence properties are better than GANs. However, a unit-variance normal distribution assumption on the decoder leads to an MSE loss on the sample images, which can cause blurry samples [19].

Normalizing flows, which constitute the core of the architecture used in our work, were first introduced along with the term by Tabak & Turner [4]. They were then used by Rezende & Mohamed [20] as flexible posterior distributions for VAEs, a powerful alternative to the initially used spherical normal distributions. For image generation, Dinh et al. [5] introduced NICE with the coupling layer and successfully modeled simple image datasets MNIST [21] and TFD [22]. This was then improved upon by RealNVP [6], a multiscale architecture based on affine coupling layers, split-squeeze blocks and masked convolutions. Glow[7] further increased the scale of the model and further refined it by adding invertible 1x1 convolutions, achieving high fidelity results on the CelebA HQ dataset [23]. NSF [24] added parameterized quadratic-spline based flows to the same architecture. Flow++ [25] improved the state of the art with variational dequantization, mixture based couplings and self-attention. Sukthanker et al. [26] further added a generic invertible attention mechanism and improved generative performance.

Different styles of normalizing flow models have also been proposed. Autoregressive approaches were developed [27, 28], with similar approaches first being introduced to once again improve VAE posteriors [29, 30]. Residual flows based on inverting constrained residual transformations [31, 8] achieved results highly competitive with the state of the art. Continuous normalizing flows with infinitely many steps were also proposed and formulated as ODEs to be solved by numeric solvers [32].

Conditional normalizing flows for modeling conditional distributions are implemented as straightforward modifications of standard normalizing flows that work by embedding conditional data into the computation and are used for various vision tasks such as segmentation, denoising and super-resolution [33, 34, 35, 36]. SRFlow [36], which was the state of the art in super-resolution on its publication is of particular interest as the backbone of our own multiscale model, and we cover it in 2.2.4. We also cover Glow [7] in 2.2.3 because it is the underlying model used by SRFlow. Further work [37] generalizes SRFlow for both upscaling and downscaling and adds other losses to obtain once again state of the art SR results and [38] examines the use of NF based loss functions in improving super-resolution model performance; but these are not related to our work. A derivative work [39] improves qualitative super-resolution results by augmenting training with paired noisy high and low resolution samples, which is different from our approach in which we only inject noise at the encoding level and focus on generative modeling.

There are also various approaches combining different generative model classes with normalizing flows: [40] proposes modeling a VAE's decoder output distribution as the latent distribution of a normalizing flow to improve generative modeling performance. The converse is done in [41], with a VAE being used to model the latent distribution of a normalizing flow instead of real images, supported by adversarial training. Pires & Figueiredo [42] propose using a mixture of K normalizing flow models trained under a variational inference framework to better model multi-modal datasets.

A few recent works in normalizing flows have particular similarities to our approach: Wavelet Flow [43] and MRCNF [44] use multiple normalizing flow models at different resolutions, but in different contexts: Wavelet Flow models Haar wavelets instead of images, and MRCNF uses continuous normalizing flows instead of discrete. Yüksel et al. [45] explore the addition of noise in the latent space of normalizing flows, but without a multi-resolution model or a variational inference framework. Instead, noise addition is done adversarially to provide useful augmentations for image classifier training.

Finally, denoising diffusion models [46] have recently received increased attention after proving that they can also generate very high fidelity images [47, 48], with their main drawback being slow sampling due to the denoising process containing a large number of steps. Ho et al. [49] is similar to our work in that a cascaded set of super-resolution models are used to refine a simpler initial image, and that various conditional augmentation methods are used. These augmentations include directly adding noise at the image level and generating corruption based on the diffusion model training process. However, our approach is different due to the fact that it uses normalizing flows instead of diffusion models and adds noise at the latent feature level, corresponding to variational training.

2.2 Background

In this section, we first discuss normalizing flows and variational auto-encoders. We then move on to conditional normalizing flows, which we base our multi-scale approach on, which is explained in the next chapter.

2.2.1 Normalizing Flows

Let $\mathcal{X}$ be the space of a given high-dimensional dataset $\mathcal{D}$; having the underlying distribution $p(\mathbf{x})$. A normalizing flow f is a bijection that aims to invertibly transform the space $\mathcal{X}$ to a new space $\mathcal{Z}$, in which points from space $\mathcal{X}$ will conform to a simpler distribution $q(\mathbf{z})$, usually a spherical multivariate Gaussian. In summary, flow models aim to transform an arbitrarily complex high-dimensional distribution to a distribution that is easy to evaluate likelihoods in and sample from.

Given samples $\mathbf{x} \sim \mathcal{D}$, the optimization process attempts to maximize their log-likelihood $\log p(\mathbf{x})$. This is equivalent to minimizing the negative log-likelihood and performed over the whole dataset:

$$\mathcal{L}(\mathcal{D}) = -\sum_{\mathbf{x}\sim\mathcal{D}} \log p(\mathbf{x}) \tag{2.1}$$

As $\mathbf{z} = f(\mathbf{x})$, the likelihood $p(\mathbf{x})$ can be computed from the simpler distribution $q(\mathbf{z})$ using a change of variables ($\mathbf{J}_f$ is the Jacobian of f):

$$\log p(\mathbf{x}) = \log q(\mathbf{z}) + \log|\det \mathbf{J}_f| \tag{2.2}$$

Complete normalizing flow models usually consist of many small cascaded invertible blocks: $f = f_0 \circ f_1 \circ \cdots \circ f_k$, and the final Jacobian $\mathbf{J_f} = \mathbf{J}_{f_0}\mathbf{J}_{f_1}\ldots\mathbf{J}_{f_k}$. This leads us to the final, expanded formula for the loss:

$$\mathcal{L}(\mathcal{D}) = -\sum_{\mathbf{x}\sim\mathcal{D}} \left(\log q(f(\mathbf{x})) + \sum_{i=0}^{k} \log|\det \mathbf{J}_{f_i}|\right) \tag{2.3}$$

Simply put, for each sample $\mathbf{x}$ in the dataset, we transform it, find the log-likelihood of the result, and add the determinant of the Jacobian of each flow block. The latter is a significant issue in the design of normalizing flows. Since the determinant of the Jacobian of each block needs to be computed for every sample, they need to be easy to compute, and most transformations are designed to have triangular Jacobians.

2.2.2 Variational Auto-Encoders

In a standard auto-encoder model, an encoder maps an input data sample $\mathbf{x}$ to a usually lower-dimensional *latent vector* $\mathbf{z}$. This vector can then mapped back into the data space via a second decoder network. The aim is discovering a latent representation for each input, which can be used for compression or comparison of samples via simpler metrics.

The variational auto-encoder [18, 50], casts this into the probabilistic domain. Instead of producing a single latent vector for an input, inputs induce a joint distribution over the latent space. Then, samples from this space are mapped back to the input space via a probabilistic decoder. Matching the latent distribution to a known simple prior also allows for data generation through the decoding of samples taken from the latent prior.

The objective function used in optimization consists of two terms, the log-likelihood of the input data over the decoder's output distribution, to be maximized, and the KL divergence between the latent distribution and its prior, to be minimized:

$$\mathcal{L}(\mathbf{x};\theta,\phi) = -\,\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x} \mid \mathbf{z};\theta)\right] + D_{\mathrm{KL}}(q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})) \tag{2.4}$$

In the above equation, $\mathbf{x}$ represents the input data, $\mathbf{z}$ represents the induced latent vector, θ represents the encoder weights, ϕ represents the decoder weights, q represents the conditional distribution at the encoder output and p represents the latent prior.

Usually, the prior for each latent attribute is assumed to be a Gaussian, and the encoder outputs a series of means and variances (which means that the latent vector distribution is a multivariate Gaussian having diagonal covariance). The decoder is also assumed to output independently distributed Gaussians having identical variance. This follows from the assumption that the output of the decoder is not just a singular reconstruction, but rather the *center* of a spherical Gaussian distribution that the decoder induces. Alternative formulations such as using the Bernoulli distribution are also possible. This is used by Kingma & Welling [18] on the MNIST dataset, with grayscale values in the [0, 1] range being the probability of that pixel being white, as opposed to black.

A tutorial detailing most of the properties of basic VAEs can be found in Kingma et al. [19].

2.2.3 Glow

Glow [7] is a normalizing flow model engineered for training on images, instead of tabular or low-dimensional data. It contains many convolutional operations and its building blocks are a series of cascaded *actnorm*, *invertible 1x1 convolution (invconv)* and *affine coupling* blocks. There are also a few *split* and *squeeze* blocks in the model for reducing dimensionality and increasing the number of channels. All of these are explained below.

2.2.3.1 Actnorm

The actnorm block is an alternative to batch normalization for very small batch sizes. Instead of computing scale and shift values from the given batch at every step, scale and shift values are initialized using the mean and standard deviation of the first training batch. They are then set as trainable parameters that can change during gradient descent. The transformation itself is the same as batch norm:

$$\text{actnorm}(\mathbf{x}) = \frac{\mathbf{x} - \boldsymbol{\mu}_{\text{learnable}}}{\boldsymbol{\sigma}_{\text{learnable}}} \tag{2.5}$$

Its inverse is also easy to compute:

$$\text{actnorm}^{-1}(\mathbf{z}) = \boldsymbol{\sigma}_{\text{learnable}} \odot \mathbf{z} + \boldsymbol{\mu}_{\text{learnable}} \tag{2.6}$$

The Jacobian is a diagonal matrix having the trainable scales as its entries; the output dimensions only depend one input dimension. The log-determinant is then the sum of the logarithms of the trainable positive scale values:

$$\log|\det \mathbf{J}_{\text{actnorm}}| = \sum_i \log \boldsymbol{\sigma}_{\text{learnable}_i} \tag{2.7}$$

2.2.3.2 Invertible 1x1 Convolution

A 1x1 convolution can be formulated as a multiplication by a $c \times c$ square matrix $\mathbf{W}$, where c is the number of channels in the input, essentially *mixing* the channels. The inverse and determinant are also computed as-is:

$$\text{invconv}(\mathbf{x}) = \mathbf{W}\mathbf{x} \tag{2.8}$$

$$\text{invconv}^{-1}(\mathbf{z}) = \mathbf{W}^{-1}\mathbf{z} \tag{2.9}$$

$$\log|\det \mathbf{J}_{\text{invconv}}| = \log|\det \mathbf{W}| \tag{2.10}$$

An alternative formulation with easier determinant computation based on LU decomposition is also proposed in Kingma & Welling [18], but we do not use it in our work.

2.2.3.3 Affine Coupling Block

Originally introduced in Dinh et al. [6], affine coupling blocks split the input into two equal parts in the channel dimension, which we define as the split$(\cdot)$ operation: The first part passes through unchanged, but is used to compute scale and shift values ($\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, respectively) by being passed through a non-invertible neural network g. The second part is then scaled and shifted by the produced values. The modified second part then recombined with the unmodified first part again in the channel dimension, which we define as concatenate$(\cdot, \cdot)$. To avoid negative, zero, and large scale values, the neural network output scales γ are exponentiated and passed through the sigmoid function σ in Glow:

$$\mathbf{x}_1, \mathbf{x}_2 = \text{split}(\mathbf{x}) \tag{2.11}$$

$$\boldsymbol{\gamma}, \boldsymbol{\beta} = \text{split}(g(\mathbf{x}_1)) \tag{2.12}$$

$$\boldsymbol{\alpha} = \sigma(\exp(\boldsymbol{\gamma}) + 2) \tag{2.13}$$

$$\mathbf{y}_1 = \mathbf{y}_1 \tag{2.14}$$

$$\mathbf{y}_2 = \boldsymbol{\alpha} \odot \mathbf{x}_2 + \boldsymbol{\beta} \tag{2.15}$$

$$\mathbf{y} = \text{concatenate}(\mathbf{y}_1, \mathbf{y}_2) \tag{2.16}$$

The addition of 2 in Eq. 2.13 is not explained in Kingma & Dhariwal [7] but present in the original source code. We suspect it was added to help the scale values $\boldsymbol{\alpha}$ be further from zero, because scale values close to zero adversely affect the stability of the inverse transformation, which is a division by the scale values.

The inverse process of the block can be easily derived by following the same steps: the scale and shift values are can be recomputed from $\mathbf{x}_1$ in the inverse direction as it remains unchanged.

Since the transformation of one half depends only on the other half, half of the Jacobian of the transformation is a triangular matrix, and the other half is an identity matrix since it remains unchanged. The log-determinant is once again the sum of the produced scale values:

$$\log|\det \mathbf{J}_{\text{affine}}| = \sum_i \log \boldsymbol{\alpha}_i \tag{2.17}$$

An alternative but weaker transformation is the *additive coupling* block that only shifts the other half of the input and avoids scaling it. We do not use it in our work.

2.2.3.4 Split and Squeeze

The split block simply removes half of the input channels in its input: they immediately become part of the latent vector $\mathbf{z}$ and their log-likelihood based on the chosen simple prior distribution is added to the loss. The other half of the channels continue to the following blocks.

The squeeze block combines groups of four pixels into a single channel, reducing the height and width of its inputs by a factor of two and increasing the number of channels by a factor of four. This *mixes* the pixels together, allowing channel-based normalising flow blocks like the actnorm and invconv to operate on a wider context.

Together, these allow the Glow architecture to be multi-scale, with each level separated by split-squeeze blocks to work on coarser and coarser details. Both were introduced with RealNVP [6].

2.2.3.5 Performance

The generative performance of Glow is evaluated only qualitatively [7], and metrics focus on comparing negative log-likelihood (NLL) on the test with previous normalising flows.

The model is able to produce visually pleasing high resolution samples when trained on the face dataset CelebA [23]. However, the samples drawn from more multimodal datasets like CIFAR-10 [51] are blobby and lack structure.

2.2.4 SRFlow

We now shortly introduce conditioning in normalizing flows before explaining how SRFlow [36] modifies various Glow [7].

2.2.4.1 Conditioning Normalizing Flows

A straightforward modification of normalizing flows is making them model a conditional distribution $p(\mathbf{x}|\mathbf{y})$. This is achieved by embedding the conditional data or its features into the computation of normalizing flow blocks. These models lose the ability to unconditionally estimate the density $p(\mathbf{x})$ of given data samples, but can be used for other purposes, such as class-conditional generation.

We are particularly interested in SRFlow [36], a conditional normalizing flow model based on Glow, optimized for the task of super-resolution. It is conditioned on low resolution images, and produces up to 8x higher resolution versions of the condition image when sampled. Instead of embedding the low resolution image into the flow directly, an RRDB-based [52] super-resolution model is pre-trained using L_1 loss, and its intermediate features are embedded into the flow model during optimization. The SRFlow model itself is trained via the previously explained log-likelihood objective.

To embed conditional features into the Glow architecture, *conditional affine coupling* and *affine injector* blocks are used.

2.2.4.2 Conditional Affine Coupling

Modifying the affine coupling to be conditional simply involves adding the conditional embeddings $\mathbf{c}$ as extra input channels to the neural network producing scale and shift values. This is a simple modification on Eq. 2.12:

$$\boldsymbol{\gamma}, \boldsymbol{\beta} = \text{split}(g(\text{concatenate}(\mathbf{x}_1, \mathbf{c}))). \tag{2.18}$$

2.2.4.3 Affine Injector

The affine injector block is also similar to the coupling block, and uses the conditional embeddings to generate scale and shift values for the whole of the input using another neural network h:

$$\boldsymbol{\alpha}, \boldsymbol{\beta} = \text{split}(h(\mathbf{c})) \tag{2.19}$$

$$\mathbf{z} = \boldsymbol{\alpha} \odot \mathbf{x} + \boldsymbol{\beta} \tag{2.20}$$

The inversion and calculation of the Jacobian determinants is straightforward and the same as the actnorm block's.

CHAPTER 3

METHOD

In this chapter we detail our approach and describe its properties, based on the background introduced in the previous chapter.

3.1 CSRFLOW- Cascaded SRFlow

Our primary aim is addressing the problem of normalizing flows being unable to generate high fidelity image samples when trained on multimodal datasets. We postulate that flow models are good at generating smooth local structure in a limited scope such as face images, but fail when there is significant global structure, such as different classes of objects having very different orientations, colors and shapes.

To produce improved, higher fidelity image samples using flow models, we propose a cascade model composed of multiple flows. As the first step, a single unconditional normalising flow model f_0 generates a low resolution image sample $\mathbf{x_0}$. This sample then becomes the condition for a 2x super-resolving SRFlow model f_1, which can then generate $\mathbf{x_1}$ having twice the resolution. $\mathbf{x_1}$ is then fed as a condition to another SRFlow model f_2, which super-resolves it to produce $\mathbf{x_2}$. This process continues through further SRFlow models until the desired resolution is reached, with the final model f_k and sample $\mathbf{x_k}$. This approach limits the task of each flow model to a much simpler generation or super-resolution problem compared to reconstructing high resolution global structure from scratch.

Although the generation process is sequential, because $\mathbf{x_{i-1}}$ needs to be generated before $\mathbf{x_i}$ can be, the training process of the models remains the same and is parallel.

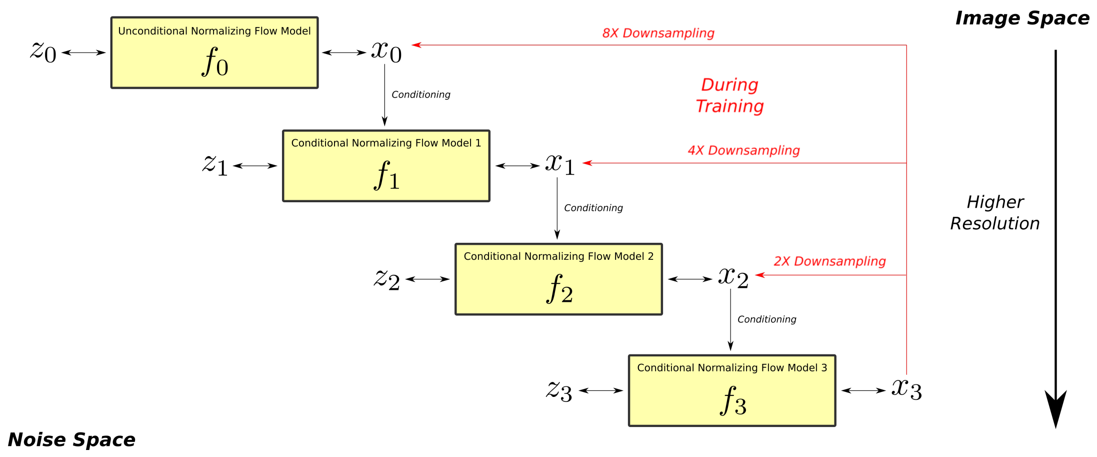


Figure 3.1: Overview of a CSRFlow Model with $k = 3$

Each SRFlow model in the cascade can be optimized on different resolution versions of the same dataset independently, along with the first unconditional Glow model.

An overview of a cascade having $k = 3$ and a super-resolution factor of 2x between models is shown in Figure 3.1.

3.1.1 Exact Density Estimation

A core advantage of normalizing flow models is the fact that they can be used for exact density estimation, and our approach preserves this property even though conditional models are involved. Our goal is estimating the density of a given high resolution data sample, $p(\mathbf{x_k})$. Let us consider the conditional probability modeled by f_i, and rewrite it using Bayes' Rule:

$$p(\mathbf{x_i} \mid \mathbf{x_{i-1}}) = \frac{p(\mathbf{x_{i-1}} \mid \mathbf{x_i})p(\mathbf{x_i})}{p(\mathbf{x_{i-1}})} \tag{3.1}$$

The result in Eq. 3.1 can be simplified by observing that $\mathbf{x_{i-1}}$ is produced from $\mathbf{x_i}$ via a deterministic downsampling operation, such as bicubic downsampling. A given $\mathbf{x_i}$ will always produce the same $\mathbf{x_{i-1}}$. Thus, $p(\mathbf{x_{i-1}} \mid \mathbf{x_i}) = 1$, which simplifies the result:

$$p(\mathbf{x_i} \mid \mathbf{x_{i-1}}) = \frac{p(\mathbf{x_i})}{p(\mathbf{x_{i-1}})} \tag{3.2}$$

We then calculate the product of the conditional densities modeled by each intermediate model, and obtain the following result:

$$\prod_{i=1}^{k} p(\mathbf{x_i} \mid \mathbf{x_{i-1}}) = \prod_{i=1}^{k} \frac{p(\mathbf{x_i})}{p(\mathbf{x_{i-1}})} \tag{3.3}$$

$$= \frac{p(\mathbf{x_k})}{p(\mathbf{x_{k-1}})} \frac{p(\mathbf{x_{k-1}})}{p(\mathbf{x_{k-2}})} \cdots \frac{p(\mathbf{x_1})}{p(\mathbf{x_0})} \tag{3.4}$$

$$= \frac{p(\mathbf{x_k})}{p(\mathbf{x_0})} \tag{3.5}$$

Because the first unconditional model f_0 models $p(\mathbf{x_0})$ directly, we can multiply the result in Eq. 3.5 with it to finally obtain the exact density estimation for a given sample $p(\mathbf{x_k})$.

In summary, we can estimate the exact likelihood of a given high resolution sample $\mathbf{x_k}$ by first downsampling it to obtain conditions and inputs $\mathbf{x_{k-1}} \dots \mathbf{x_1}, \mathbf{x_0}$ for the intermediate models, and then multiply the likelihoods produced by each model to obtain $p(\mathbf{x_k})$. The log-likelihood will similarly be the sum of the log-likelihood estimated by each model.

3.2 Variational Training

A problem arising with any cascaded architecture is error and noise amplification. If one of the earlier models in the cascade produces a noisy or slightly incorrect sample, the error will be amplified further and further by each following super-resolution model, resulting in a highly distorted final sample.

To alleviate this problem, we propose to *augment* the low resolution conditioning data for the intermediate SRFlow models. In SRFlow's optimization process, only one low resolution input is paired with the high resolution image: its perfectly downsampled version, containing no added artifacts or noise. Instead, we want the current model to be able to super-resolve its input even when the input is an imperfect generation produced by the previous level's model. To perform this in a principled way, we turn to variational inference with the aim of maximizing a lower bound on the log-likelihood of our current level $\log p(\mathbf{x}_i)$ through the use of the previous level's latent variable $\mathbf{z}_{i-1}$:

$$\log p(\mathbf{x}_i) = \int p(\mathbf{x}_i, \mathbf{z}_{i-1}) d\mathbf{z}_{i-1} \tag{3.6}$$

$$\geq \mathbb{E}_{q(\mathbf{z}_{i-1}|\mathbf{x}_i)} \left[\log p(\mathbf{x}_i \mid \mathbf{z}_{i-1})\right] - D_{\mathrm{KL}}(q(\mathbf{z}_{i-1} \mid \mathbf{x}_i) \parallel p(\mathbf{z}_{i-1})) \tag{3.7}$$

Examining Eq. 3.7, we can make an observation specific to our case: Because $\mathbf{x}_{i-1} = f_{i-1}^{-1}(\mathbf{z}_{i-1})$, $p(\mathbf{x}_i \mid \mathbf{z}_{i-1}) = p(\mathbf{x}_i \mid \mathbf{x}_{i-1})$, which is directly the probability density modeled by our current conditional super-resolution model. Next, we need to decide on the posterior approximation $q(\mathbf{z}_{i-1} \mid \mathbf{x}_i)$. We choose to use a normal distribution with a constant variance σ centered around $\mathbf{z}_{i-1}$ with $q(\mathbf{z}_{i-1} \mid \mathbf{x}_i) = \mathcal{N}(\mathbf{z}_{i-1}, \sigma\mathbf{I}) = \mathcal{N}(f_{i-1}(\mathbf{x}_{i-1}), \sigma\mathbf{I})$. Training f_i's likelihood based on samples drawn from this chosen approximate posterior, which is essentially disturbing $\mathbf{x}_{i-1}$ at the feature level $\mathbf{z}_{i-1}$ with Gaussian noise, maximizes part of a lower bound on $\log p(\mathbf{x}_i)$, with the tightness of this bound being dependent on how close our chosen approximate is to the true posterior. We believe our choice constitutes a good approximation to the true posterior based on the fact that f_{i-1} has already been trained to model a complex distribution for $\mathbf{x}_{i-1}$.

As another boon, the KL divergence term $D_{\mathrm{KL}}(q(\mathbf{z}_{i-1} \mid \mathbf{x}_i) \parallel r(\mathbf{z}_{i-1}))$ turns out to be equivalent to f_{i-1}'s standard training objective based on maximum likelihood. First, let us derive the maximum likelihood based objective for f_{i-1}: The well-known probability density function of an arbitrary multivariate k-dimensional Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the following. We use $\mathbf{z}$ as our variable:

$$p(\mathbf{z}) = \frac{\exp(-\frac{1}{2}(\mathbf{z} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{z} - \boldsymbol{\mu}))}{\sqrt{(2\pi^k)|\boldsymbol{\Sigma}|}} \tag{3.8}$$

The log-probability can be obtained by taking the logarithm of both sides:

$$\log p(\mathbf{z}) = -\frac{1}{2}\left[(\mathbf{z} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{z} - \boldsymbol{\mu}) + k \log 2\pi + \log |\boldsymbol{\Sigma}|\right] \tag{3.9}$$

In our normalising flow case, we are trying to match $\mathbf{z}_{i-1}$ to the standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Since $\mathbf{I}^{-1} = \mathbf{I}$ and $|\mathbf{I}| = 1$, the expression in Eq. 3.9 simplifies to the following:

$$\log p(\mathbf{z}_{i-1}) = -\frac{1}{2}(\mathbf{z}_{i-1}^T \mathbf{z}_{i-1} + k \log 2\pi) \tag{3.10}$$

Because k is constant, maximizing the log-likelihood in Eq. 3.10 is equivalent to minimizing the dot product $\mathbf{z}_{i-1}^T\mathbf{z}_{i-1}$.

Now, we focus on the KL-divergence. The KL-divergence between two k-dimensional multivariate normal distributions $\mathcal{N}_0(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ and $\mathcal{N}_1(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ is given by the following expression:

$$D_{\mathrm{KL}}(\mathcal{N}_0 \parallel \mathcal{N}_1) = \frac{1}{2}\left(\operatorname{tr}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0) - k + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) + \log\frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_0|}\right) \tag{3.11}$$

Our choice of the induced distribution is $\mathcal{N}(\mathbf{z}_{i-1}, \sigma\mathbf{I})$ with constant σ, which we are trying to match to the standard normal $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Note that $(\sigma\mathbf{I})^{-1} = \frac{1}{\sigma}\mathbf{I}$ and $|\sigma\mathbf{I}| = \sigma^k$. This leads to the following expression:

$$\begin{aligned}
D_{\mathrm{KL}}(\mathcal{N}(\mathbf{z}_{i-1}, \sigma\mathbf{I}) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I})) &= \frac{1}{2}\left(\operatorname{tr}(\sigma\mathbf{I}) - k - \mathbf{z}_{i-1}^T\mathbf{z}_{i-1} + \log\frac{1}{\sigma^k}\right) && (3.12)\\
&= \frac{1}{2}(k\sigma - k - \mathbf{z}_{i-1}^T\mathbf{z}_{i-1} - k\log\sigma) && (3.13)\\
&= \frac{1}{2}\left[\mathbf{z}_{i-1}^T\mathbf{z}_{i-1} + k(\sigma - 1 - \log\sigma)\right] && (3.14)
\end{aligned}$$

Since σ is a hyper-parameter that remains constant during optimization, the objective in Eq. 3.14 is also equivalent to minimizing $\mathbf{z}_{i-1}^T\mathbf{z}_{i-1}$, just like Eq. 3.10!

This result indicates that the standard likelihood-based normalizing flow training has already minimized the KL divergence term between our choice of induced distribution and the standard normal prior. Ideally, we should allow weights from f_{i-1} to change during this process to possibly trade-off the KL-divergence term with the conditional likelihood term, but this further slows down and risks destabilizing the training due to the fact that we are making both a forward and an inverse pass through f_{i-1}. Instead, we keep the parameters of f_{i-1} frozen during the variational training process and only update the conditional likelihood term through training f_i with low-resolution inputs from our approximate posterior, which still tightens the bound on $\log p(\mathbf{x}_i)$.

3.2.1 Improving the Invertibility of Normalizing Flows

Although normalizing flows are analytically stable, a problem that arises often in practice are $\mathbf{z}$ samples that blow up and produce `inf` and `NaN` values when put through the inverse model f^{-1} [53]. This is especially prevalent when the model is provided unexpected out-of-distribution inputs, and happens quite often with our approach since we add noise to $\mathbf{z}$ values.

The main cause of this problem is the affine coupling block discussed in Eq. 2.2.3.3. Scale values in the range $[0, 1]$ or even $[\epsilon, 0]$ for very small ϵ such as 10^{-4} are fine for the forward direction, but cause values to explode in the reverse direction through repeated divisions by small values; especially for deep models that contain up to 128 such blocks.

To remedy this problem while not entirely getting rid of affine coupling blocks, we limit the scales to the range $[\alpha, 1]$ by adding a constant and scaling the sigmoid, shown with function s below:

$$s(\mathbf{x}) = (1 - \alpha)\sigma(\mathbf{x}) + \alpha \tag{3.15}$$

We use $\alpha = 0.5$ in our work and call the approach *sigmoid squishing*. A scale-limiting approach is discussed in Behrmann et al. [53], but how exactly it is implemented is left unclear. Glow [7] uses additive instead of affine coupling blocks in qualitative experiments to avoid this invertibility issue.

While this very significantly reduces the prevalence of inverse stability problems in our models, it is still possible to have samples that blow up, especially with larger σ values. To prevent this from adversely affecting the model, we replace x_i' that contain unreasonably large or non-finite values back with their originals x_i during the optimization process. We also keep track of the amount of x_i' we replace and call its ratio to the total number of produced x_i' the *replacement rate*. As expected, the replacement rate increases as the standard deviation of the approximate posterior σ is increased.

CHAPTER 4

EXPERIMENTS

We now show our experimental results in this chapter. We begin by discussing the datasets we use and our evaluation setup.

4.1 Datasets

We perform experiments on two datasets used widely in generative modeling settings: CelebA [54] and CIFAR-10 [51].

Table 4.1: Details of the Datasets Used in Our Experiments

Dataset	Image Size	Number of Images			Number of Classes
		Training	Validation	Test	
CelebA	256×256	162,770	19,867	19,962	-
CIFAR-10	32×32	50,000	-	10,000	10

The CelebA [54] dataset is composed of more than 200,000 high resolution celebrity images from the internet with large pose and size variations. These images are annotated with landmarks for cropping faces from them as well as attributes, and a pre-cropped version is provided with all face images resized to 218×178. To obtain higher quality images for use as a baseline, we center-crop face images using the provided landmarks and resize them to a uniform 256×256. We further downsample these using a bicubic kernel by powers of two for use in our intermediate models to obtain 128×128, 64×64, 32×32, 16×16 and 8×8 images. We do not make use of the face attribute data in our work.

The CIFAR-10 [51] dataset has a total 60,000 small 32×32 images equally distributed through its ten classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. We downsample these to 16×16 and 8×8 for our intermediate models, and also employ the class labels in some experiments. Since the dataset has no official validation set, we randomly split 5,000 images (500 from each class) from the training set to use for hyper-parameter tuning as our validation set, and use the remaining 45,000 images for training.

In our final experiments, we add the validation set to the training set and report evaluation metrics on the test set.

4.2 Evaluation Metrics

The FID (Fréchet Inception Distance) [12] metric is commonly used to evaluate the generative quality of deep models. FID works by comparing the distribution of two datasets' InceptionV3 embeddings, keeping the underlying pre-trained network the same as the previous IS (Inception Score) metric [10]. The mean and covariance matrix of the embeddings are calculated, and the embeddings are assumed to be normally distributed. Then, the distributions of the difference between the two distributions is calculated with 2-Wasserstein Distance, and the result is the FID. Since it is a distance metric, a lower scalar value for the FID implies better generative properties.

KID (Kernel Inception Distance) [55] was proposed more recently as a statistically unbiased, improved alternative to FID. Another improved approach [56] proposes disentangling the fidelity of the generated images and the coverage of the provided dataset with precision and recall as the two metrics, since FID is a single scalar rewarding both of these: A model generating high fidelity images but covering only a few modes of the dataset will get an FID score similar to a very different model generating low fidelity images but having excellent mode coverage [56].

Even though these recent works offer improvements over FID, FID remains more prevalent as a measure of generative quality in literature, including recent work we compare our results with [57]. For this reason, we also choose FID as the evaluation metric in our experiments. We use 50,000 generated samples to calculate sample statistics in our experiments unless stated otherwise, which is also the number used in the original work [12]. For dataset splits, we use all the available images in the split to compute statistics. Also note that we compare each model with the dataset resized to its target size. For example, a model trained on the resized 16×16 CIFAR-10 dataset is evaluated by calculating its FID relative to the resized 16×16 CIFAR-10 dataset, not the original 32×32 CIFAR-10 dataset.

4.3 Training Setup

4.3.1 Architecture Details

We use the SRFlow [36] architecture for our conditional upscaling models, and Glow [7] as our base model generating small images unconditionally. We use the sigmoid squishing approach discussed in 3.2.1 with $\alpha = 0.5$ in all of our SRFlow models to improve their inverse stability, but not the base Glow models since they are stable enough without sigmoid squishing.

Our base Glow models are composed of $L = 3$ levels at different scales (levels are as explained in 2.2.3.4), with each level containing $K = 32$ triple actnorm-invertible 1x1 convolution-affine coupling blocks. The small neural networks generating scale and shift values have a large $k = 512$ channels in their intermediate activation layers.

Our SRFlow models have the same underlying architecture as Glow and are composed of $L = 4$, with each level containing $K = 16$ triplet blocks. Scale and shift generating networks have $k = 64$ channels in the intermediate activation layers to reduce computation time. The RRDB [52] model generating conditional embeddings from the low-resolution input has the standard 23-block architecture, and we distribute the same embeddings as conditional inputs to each level of the underlying Glow model.

For our main CIFAR-10 experiments, we use class-conditional models as done for the qualitative experiments in Glow [7]. This is achieved by making the prior of the final level latent variables have class-dependent mean and variance, generated from a one-hot encoding with a linear transformation, instead of being standard normals. A binary cross-entropy loss term is also introduced on a linear projection of the latent variable to encourage class separation. This term is weighted by a factor $\lambda_c = 0.01$ and added to the original NLL (negative log-likelihood) loss during training. We also extend our SRFlow models with the same class-conditional modeling capabilities.

4.3.2 Optimization Settings

We use the Adamax [58] instead of the more commonly used Adam [58] for training both our Glow and SRFlow models since we find it improves model convergence.

Our Glow models are trained with learning rate $\lambda = 5 \times 10^{-4}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$, with $\gamma = 5 \times 10^{-5}$ weighted L2 weight decay. Every model is trained for 200,000 iterations, and the learning rate is linearly increased during the first 5 epochs. A batch size of 128 is used in our CelebA experiments, and a batch size of 512 is used in our CIFAR-10 experiments.

For the SRFlow models, we use a slightly reduced learning rate $\lambda = 2.5 \times 10^{-4}$, $\beta_1 = 0.9$ and $\beta_2 = 0.99$ with no L2 weight decay. The models are again trained for 200,000 iterations, and their learning rate is cut in half at 100,000, 150,000, 180,000 and 190,000 iterations to improve final convergence. A batch size of 64 is used all around. The conditional RRDB models are also pre-trained for 200,000 iterations using L1 loss, with Adam as their optimizer and a learning rate of $\lambda = 2 \times 10^{-4}$. During SRFlow training, the RRDB model is frozen for the first 100,000 iterations, and then unfrozen and fine-tuned for the following 100,000.

4.4 Quantitative Results

We now provide quantitative FID results of our experiments on the test splits of the CelebA dataset in Table 4.2 and the CIFAR-10 dataset in Table 4.3. As with Glow [7], we use reduced-temperature sampling, in which the standard deviation of the prior distribution is reduced through multiplication with a constant to decrease the number of outlying high-noise samples, with $\tau = 1.0$ implying an unmodified prior. We use temperature $\tau = 0.8$ for our CelebA sampling and $\tau = 0.9$ for our CIFAR-10 sampling. For calculating FID, we use CleanFID [59], which is careful with the selection of filters used in rescaling, as the choice can significantly affect FID results. Most studies do not report FID on CelebA due to it being a relatively unimodal dataset, and Grcic et al. [57] report FID on the training split rather than test split. We however still calculate FID scores on the CelebA test split to be able to compare different models internally in our work.

Because our quantitative experiments only show at most one variationally-trained model in the cascade, we also provide that model's replacement rate, which is the ratio of noisy augmented samples that we have to replace with their originals due to an unsuccessful inversion attempt producing very large, `NaN` or `inf` values. This rate becomes higher with increasing σ as latent variables get further from the training distribution with added noise.

Our models are named through a combination of the models in their cascade, and values in parentheses represent the σ value used in their variational training if performed. Intermediate FID values are reported on intermediate results. We use the CC suffix to denote class-conditional models for the CIFAR-10 experiments. To clarify with an example: Glow16-SRF32(0.02)-SRF64 implies a cascade in which a Glow model generates a base 16×16 image. Then, an SRFlow model generating 32×32 images goes through variational training using $\sigma = 0.02$ with the previous Glow model. Finally, an SRFlow model generating 64×64 images is trained in the standard fashion (no σ coefficient given) using low-resolution conditioning data from the dataset only.

Our results in terms of FID are lower than results reported in recent literature. We believe that this is due to the lack of in-depth tuning: FID reported by Chen et al. [8] from an official Glow model on CIFAR-10 is 46.90, while we only obtain 60.91 when attempting to train the same model. However, we believe that our results are internally consistent since we use the same settings when training our own models.

The first interesting observation of note when comparing results from the two datasets is the fact that starting unconditional generation from the lowest possible resolution is not always beneficial. FID on CelebA turns out best when starting with Glow from 16×16 rather than 32×32 or 64×64. For CIFAR-10 however, starting from 16×16 is significantly better than starting from 8×8, while also being better than

Table 4.2: FID Results on CelebA

	Model	Replacement Rate	CelebA FID Score		
			16	32	64
Related Work	DenseFlow-74-10 (Grcić *et al.*, NIPS'2021)	-	-	-	17.1
Ours	Glow16-SRF32-SRF64	-	5.27	12.30	28.55
	Glow16-SRF32(0.02)-SRF64	0.00	5.27	**12.09**	27.14
	Glow16-SRF32(0.03)-SRF64	0.00	5.27	12.41	27.84
	Glow16-SRF32(0.05)-SRF64	0.00	5.27	14.17	29.13
	Glow16-SRF32-SRF64(0.25)	0.00	5.27	12.30	25.79
	Glow16-SRF32-SRF64(0.50)	0.00	5.27	12.30	**22.76**
	Glow16-SRF32-SRF64(1.00)	0.11	5.27	12.30	29.13
	Glow16-SRF32(0.02)-SRF64(0.50)$^{(*)}$	-	5.27	12.09	22.03
	Glow32-SRF64	-	-	18.71	36.11
	Glow64	-	-	-	50.52

performing the full generation with Glow at 32×32. We believe that every dataset has its own sweet-spot for the starting resolution: 16×16 images may form a good template for face generation, but 8×8 CIFAR-10 images are little more than high-entropy color blobs and training a generative model on them seems unstable. Adding further information in the form of class-conditioning to the 8×8 setting unexpectedly destabilizes the training and results in a lower FID score than unconditional training, even though class-conditioning significantly improves FID in the 32×32 setting. With the proper starting resolution, our cascade models outperform direct generation even without variational training.

Another interesting result is the fact that our Glow16-SRF32 cascade slightly outperforms the GlowCC32 model that is class-conditional, while being entirerly unconditional. In contrast, our class-conditional cascade model GlowCC16-SRFCC32 shows only slight improvement compared to its conditional counterpart.

Both the CelebA and CIFAR-10 results show that variational training with a sensible σ value can improve generative results, while large values decrease FID due to the inconsistencies they induce. In the CelebA setting, we see that training the final SRF64 model with $\sigma = 0.25$ obtains better FID than standard non-variational training, while $\sigma = 0.5$ achieves even better FID. However, $\sigma = 1.0$ gets worse results than standard training due to excessive noise. We can see similar results in CIFAR-10 with

Table 4.3: FID Results on CIFAR-10

	Model	Replacement Rate	CIFAR-10 FID Score		
			8	16	32
Related Work	i-ResNet (Behrmann *et al.*, ICML'2019)	-	-	-	65.01
	Glow (Kingma & Dhariwal, NIPS'2018)	-	-	-	46.90
	Residual Flow (Chen *et al.*, NIPS'2019)	-	-	-	46.37
	DenseFlow-74-10 (Grcić *et al.*, NIPS'2021)	-	-	-	34.90
Ours	Glow8-SRF16-SRF32	-	8.33	31.90	72.77
	Glow8-SRF16-SRF32(0.75)	0.11	8.33	31.90	73.32
	Glow8-SRF16-SRF32(1.00)	0.55	8.33	31.90	73.17
	Glow16-SRF32	-	-	**22.29**	60.07
	Glow32	-	-	-	77.54
	GlowCC8-SRFCC16-SRFCC32	-	11.93	35.87	73.64
	GlowCC8-SRFCC16(0.02)-SRFCC32	0.00	11.93	32.42	75.70
	GlowCC8-SRFCC16(0.05)-SRFCC32	0.01	11.93	38.12	82.23
	GlowCC8-SRFCC16(0.10)-SRFCC32	0.14	11.93	42.54	87.7
	GlowCC16-SRFCC32	-	-	22.89	**58.35**
	GlowCC32	-	-	-	60.91

SRFCC16's variational training, where $\sigma = 0.05$ is optimal in the 16×16 setting.

4.4.1 Selection of Noise Variance for Variational Training

Intuitively, we want variational training to help train our current model *fix* mistakes that may be made by the previous model. For this reason, we disturb the low-resolution conditioning input at the feature level with σ standard-deviation Gaussian noise while keeping the target high-resolution output the same. For low enough σ, the augmented LR input corresponds to the same image with some unnatural artifacts, while it becomes excessively noisy or something else entirely for large σ.

For Glow, injecting noise at the feature level quickly moves the reconstruction away from the original image as shown in Figure 4.1. We therefore keep σ in the small [0, 0.1] range to have sensible pairs.

SRFlow however is highly resistant to noise as shown in Figure 4.2 due to being conditioned on low-resolution input, and we can keep σ large in the range [0, 1]. We

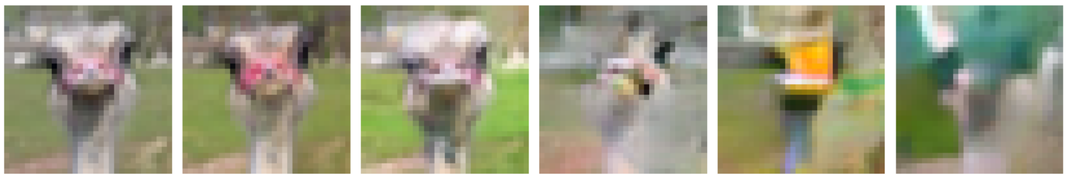


Figure 4.1: Glow Reconstructions with σ linearly increasing from 0 to 0.25

can still see that the injected noise introduces color jitter-like artifacts even for smaller σ upon close inspection, which we believe helps the variationally trained model get used to *fixing* artifacts such as these instead of amplifying them.

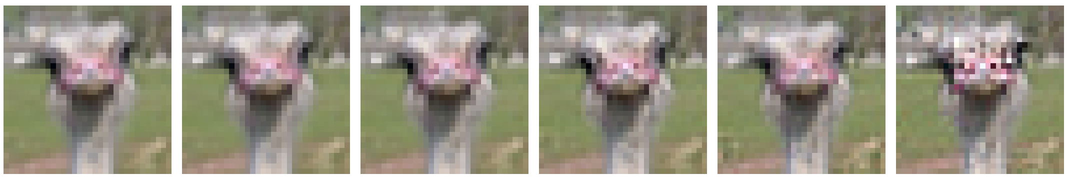


Figure 4.2: SRFlow Reconstructions with σ linearly increasing from 0 to 1

4.4.2 Quantitative Effects of Sigmoid Squishing

The sigmoid squishing modification we apply to the affine coupling is necessary to improve the invertibility of our models for variational training. To illustrate this, we measure *sample failure rate* during training, which is the ratio of images whose reconstructions from the latent space fail after adding σ noise to their latent representations, failure in this instance meaning producing an reconstruction containing `inf` or `NaN` values. How sample failure rate evolves during training for a non-squished SRFlow model ($\alpha = 0$) compared to a squished SRFlow model ($\alpha = 0.5$) on 32×32 CelebA is shown in Figure 4.3. The squished model quickly achieves practically zero failure rate while the non-squished model still fails on more than 60% of the samples for noise $\sigma = 0.5$. The effect is even more pronounced for $\sigma = 1.0$, where the non-squished model always fails, while the squished model successfully manages to learn a reduced failure rate.

To observe possible effects squishing may have on sample quality, we run a small ablation experiment where we train both a squished and non-squished Glow model on the 8×8 and 16×16 CelebA test splits. Their sample FID results are shown in Table 4.4 on a few different temperatures. Rather than having a detrimental effect, the squishing seems to reduce the frequency of outliers for higher temperatures through its inverse stabilization effect, slightly improving FID results for higher temperatures.

Table 4.4: Effect of Sigmoid Squishing Glow on FID on Resized CelebA

Glow Model α	CelebA FID Score					
	$\tau = 0.8$		$\tau = 0.9$		$\tau = 1.0$	
	8	16	8	16	8	16
$\alpha = 0$	4.83	5.27	2.25	4.76	1.79	5.62
$\alpha = 0.5$	3.27	6.01	1.90	4.73	1.51	5.41

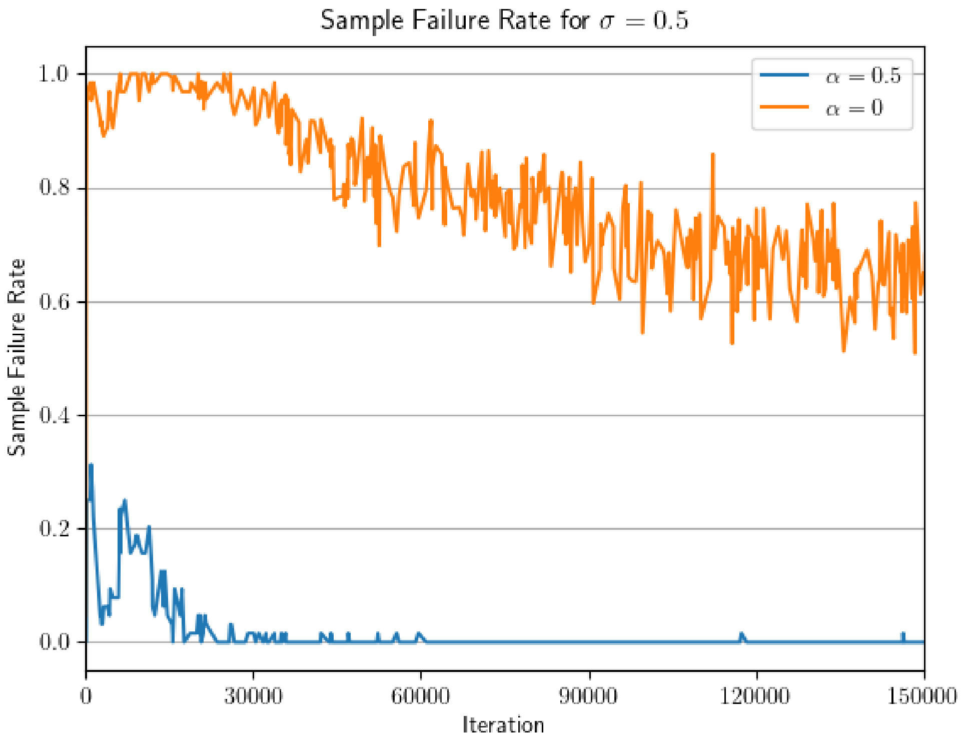
4.4.3 Effect of Sampling Temperature on FID

The effect of temperature on the FID on our best models for the two datasets are shown in Figure 4.4. We see that CIFAR-10 is more adversely affected by low temperature due to the multi-modality of the dataset relative to CelebA and is less affected by outliers brought about by higher temperature for the same reason. The optimal temperature ranges for FID look like [0.8, 0.9] for CIFAR-10 and [0.7, 0.8] for CelebA. We observed similar results with our initial models and erred on the side of generation variety, choosing $\tau = 0.9$ for CIFAR-10 and $\tau = 0.8$ for CIFAR-10.

4.5 Qualitative Results

We show CelebA results from our best model Glow16-SRF32-SRF64(0.5) that achieves 22.76 FID. and compare with the same model that is not trained variationally, Glow16-SRF32-SRF64 that achieves 28.55 FID in Figure 4.5. The variationally trained model seems to produce smoother outputs with less abrupt edge and color jittering. This becomes more easily visible in more unlikely faces as we go down in the figure: the variationally trained model successfully recovers structure while the standard model produces noisy outputs having some inconsistent facial features.

We also show random class-conditional samples from our best CIFAR-10 model GlowCC16-SRFCC32 achieving 58.35 FID in Figure 4.6. While the model has a general idea about classes, it has trouble with the multi-modal dataset and seems to mix classes in some samples, producing objects like an automobile with horse legs and a cat with a bird head.



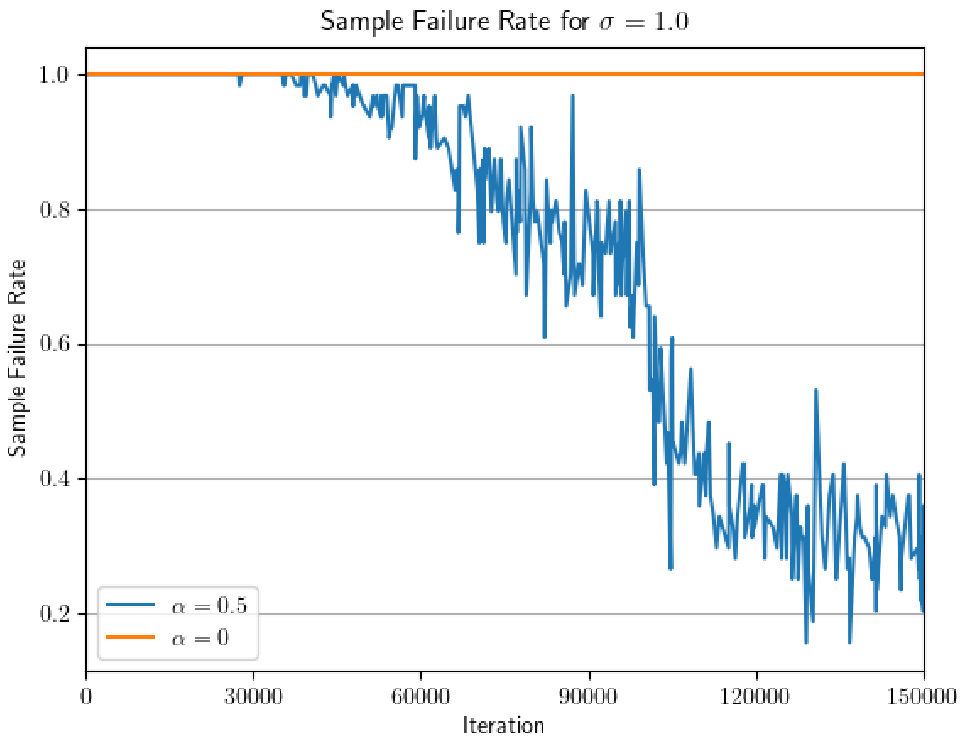


Figure 4.3: Sample Failure Rate during 32×32 CelebA Training with SRFlow

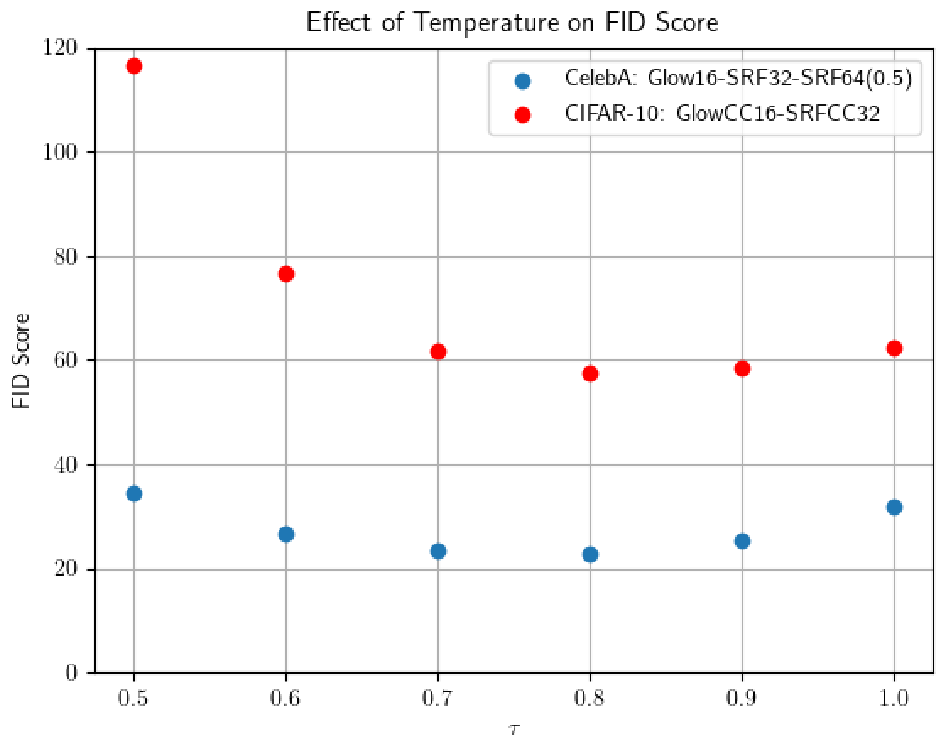


Figure 4.4: Effect of Temperature on FID on the Two Datasets

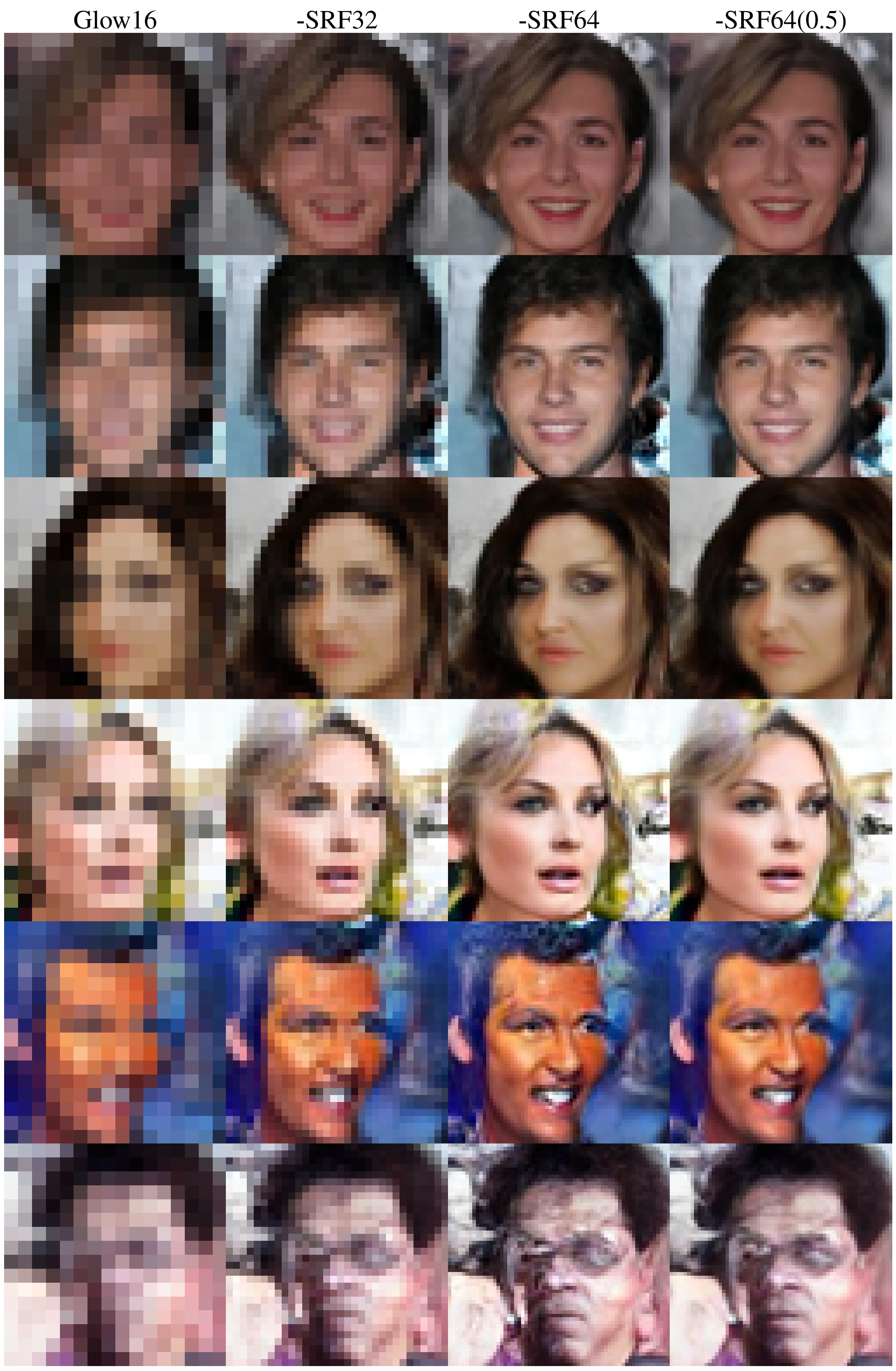


Figure 4.5: Comparison Between Non-Variational and Variational Training Results

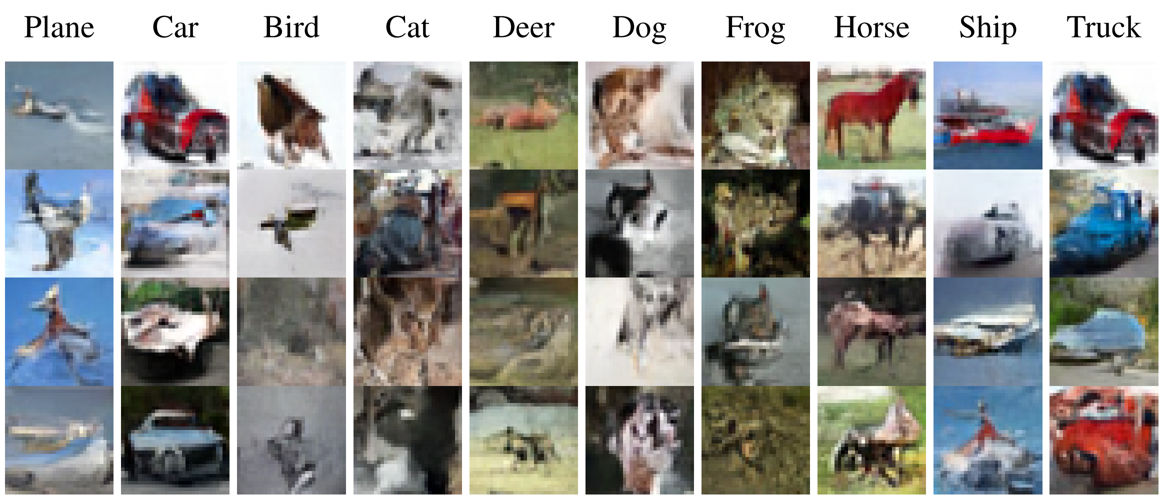


Figure 4.6: Random Samples from Our Best Class-Conditional CIFAR-10 Cascade

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this final chapter, we conclude this thesis with a summary of our contributions and discuss future methods in which our current work can be extended.

5.1 Conclusion

We propose a method to improve image fidelity in normalizing flow models via a cascade composed of a small resolution unconditional normalizing flow model followed by conditional super-resolution normalizing flow models that refine the initially generated image. We thus aim to simplify the task of each model to either generating a simple image or slightly improving an existing one rather than generating a whole high resolution image from scratch with a single model.

We further introduce a principled way of augmenting the low-resolution conditioning data for our intermediate super-resolution models based on variational inference. This augmentation disturbs low-resolution inputs at the feature level while keeping their target super-resolution the same, aiming to render the cascade more resistant to artifacts and noisy outliers that may be produced by initial models in the cascade and prevent their error from being amplified.

While not on par with state of the art methods due to lack of fine-tuning, our experimental results in Chapter 4 internally show that with a correct choice of initial model resolution, our cascade models improve generation fidelity relative to direct generation when evaluated with FID score. Our low-resolution augmentation method is also shown to be able to improve FID when the amount of noise added at the feature level is chosen sensibly. We also demonstrate the necessity of improving the invertibility of the models we use to make our variational training scheme viable, and explore the effect of low-temperature sampling on FID scores.

5.2 Future Work

We plan on further tuning the architecture and training settings of our models and match our own Glow model's results with the official pre-trained ones to be able to really compare results with the state of the art in an unbiased setting in future studies, possibly on additional datasets.

A possible improvement on our variational training scheme is proper noise scaling. While we apply the same amount of noise to each latent vector split in the multi-level Glow architecture, these partial latent vectors do not all have the same underlying standard normal distribution: their means and variances are conditioned on the part of the input that has been split off. Additionally, the final latent variable can also have means and variances conditioned on class labels or as set as learnable parameters. Tuning the scaling of the noise to each of these partial latent vectors individually could improve the performance gains offered by our variational training scheme.

Improving perceptual image quality in models trained via maximum-likelihood approaches remains an interesting problem with many avenues to be explored.

REFERENCES

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[2] I. J. Goodfellow, “On distinguishability criteria for estimating generative models,” May 2015. arXiv:1412.6515 [stat].

[3] G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez, “Invertible Conditional GANs for image editing,” Nov. 2016. arXiv:1611.06355 [cs].

[4] E. G. Tabak and C. V. Turner, “A family of nonparametric density estimation algorithms,” *Communications on Pure and Applied Mathematics*, vol. 66, no. 2, pp. 145–164, 2013.

[5] L. Dinh, D. Krueger, and Y. Bengio, “NICE: non-linear independent components estimation,” in *International Conference on Learning Representations (ICLR)*, 2015.

[6] L. Dinh, J. Sohl-Dickstein, and S. Bengio, “Density estimation using real NVP,” in *International Conference on Learning Representations (ICLR)*, 2017.

[7] D. P. Kingma and P. Dhariwal, “Glow: Generative flow with invertible 1x1 convolutions,” *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[8] R. T. Chen, J. Behrmann, D. K. Duvenaud, and J.-H. Jacobsen, “Residual flows for invertible generative modeling,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[9] A. Brock, J. Donahue, and K. Simonyan, “Large Scale GAN Training for High Fidelity Natural Image Synthesis,” Feb. 2019. arXiv:1809.11096 [cs, stat].

[10] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training gans,” *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[11] A. Grover, M. Dhar, and S. Ermon, "Flow-GAN: Combining Maximum Likelihood and Adversarial Learning in Generative Models," *arXiv:1705.08868 [cs, stat]*, Jan. 2018. arXiv: 1705.08868.

[12] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[13] A. H. Bermano, R. Gal, Y. Alaluf, R. Mokady, Y. Nitzan, O. Tov, O. Patashnik, and D. Cohen-Or, "State-of-the-art in the architecture, methods and applications of stylegan," in *Computer Graphics Forum*, vol. 41, pp. 591–611, Wiley Online Library, 2022.

[14] A. Van Den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *International Conference on Machine Learning (ICML)*, pp. 1747–1756, PMLR, 2016.

[15] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, *et al.*, "Conditional image generation with pixelcnn decoders," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[16] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," Sept. 2016. arXiv:1609.03499 [cs].

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," Dec. 2017. arXiv:1706.03762 [cs].

[18] D. P. Kingma and M. Welling, "Auto-encoding variational {Bayes}," in *International Conference on Learning Representations (ICLR)*, 2014.

[19] D. P. Kingma, M. Welling, *et al.*, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.

[20] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International Conference on Machine Learning (ICML)*, pp. 1530–1538, PMLR, 2015.

[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[22] J. Susskind, A. Anderson, and G. E. Hinton, "The toronto face dataset," 2010.

[23] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *International Conference on Learning Representations (ICLR)*, 2018.

[24] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, "Neural spline flows," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[25] J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel, "Flow++: Improving flow-based generative models with variational dequantization and architecture design," in *International Conference on Machine Learning (ICML)*, pp. 2722–2730, PMLR, 2019.

[26] R. S. Sukthanker, Z. Huang, S. Kumar, R. Timofte, and L. Van Gool, "Generative flows with invertible attentions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11234–11243, 2022.

[27] C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville, "Neural Autoregressive Flows," Apr. 2018. arXiv:1804.00779 [cs, stat].

[28] P. Jaini, K. A. Selby, and Y. Yu, "Sum-of-squares polynomial flow," in *International Conference on Machine Learning (ICML)*, pp. 3009–3018, PMLR, 2019.

[29] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improved variational inference with inverse autoregressive flow," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[30] G. Papamakarios, T. Pavlakou, and I. Murray, "Masked autoregressive flow for density estimation," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[31] J. Behrmann, W. Grathwohl, R. T. Chen, D. Duvenaud, and J.-H. Jacobsen, "Invertible residual networks," in *International Conference on Machine Learning (ICML)*, pp. 573–582, PMLR, 2019.

[32] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[33] C. Winkler, D. Worrall, E. Hoogeboom, and M. Welling, "Learning Likelihoods with Conditional Normalizing Flows," *arXiv:1912.00042 [cs, stat]*, Nov. 2019.

[34] Y. Lu and B. Huang, "Structured output learning with conditional generative flows," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 5005–5012, 2020.

[35] A. Abdelhamed, M. A. Brubaker, and M. S. Brown, "Noise Flow: Noise Modeling with Conditional Normalizing Flows," Aug. 2019. arXiv:1908.08453 [cs, eess].

[36] A. Lugmayr, M. Danelljan, L. V. Gool, and R. Timofte, "Srflow: Learning the super-resolution space with normalizing flow," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 715–732, 2020.

[37] J. Liang, A. Lugmayr, K. Zhang, M. Danelljan, L. Van Gool, and R. Timofte, "Hierarchical conditional flow: A unified framework for image super-resolution and image rescaling," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4076–4085, 2021.

[38] A. Lugmayr, M. Danelljan, F. Yu, L. Van Gool, and R. Timofte, "Normalizing flow as a flexible fidelity objective for photo-realistic super-resolution," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1756–1765, 2022.

[39] Y. Kim and D. Son, "Noise Conditional Flow Model for Learning the Super-Resolution Space," June 2021. arXiv:2106.04428 [cs].

[40] R. Morrow and W.-C. Chiu, "Variational Autoencoders with Normalizing Flow Decoders," *arXiv:2004.0617 [cs, stat]*, Apr. 2020.

[41] T. Lucas, K. Shmelkov, K. Alahari, C. Schmid, and J. Verbeek, "Adversarial training of partially invertible variational autoencoders," in *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models (INNF+)*, 2019.

[42] G. G. F. Pires and M. A. Figueiredo, "Variational mixture of normalizing flows," in *Proceedings of the European Symposium on Artificial Neural Networks*, 2020.

[43] J. J. Yu, K. G. Derpanis, and M. A. Brubaker, "Wavelet flow: Fast training of high resolution normalizing flows," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6184–6196, 2020.

[44] V. Voleti, C. Finlay, A. Oberman, and C. Pal, "Multi-Resolution Continuous Normalizing Flows," in *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models (INNF+)*, 2021.

[45] O. K. Yüksel, S. U. Stich, M. Jaggi, and T. Chavdarova, "Semantic perturbations with normalizing flows for improved generalization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6619–6629, 2021.

[46] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International Conference on Machine Learning (ICML)*, pp. 2256–2265, PMLR, 2015.

[47] Y. Song and S. Ermon, "Improved techniques for training score-based generative models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12438–12448, 2020.

[48] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[49] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans, "Cascaded diffusion models for high fidelity image generation," *Journal of Machine Learning Research*, vol. 23, pp. 47–1, 2022.

[50] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *International conference on machine learning*, pp. 1278–1286, PMLR, 2014.

[51] A. Krizhevsky, "Learning multiple layers of features from tiny images," Sept. 2009.

[52] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy, "Esrgan: Enhanced super-resolution generative adversarial networks," in *European Conference on Computer Vision (ECCV) Workshops*, 2018.

[53] J. Behrmann, P. Vicol, K.-C. Wang, R. Grosse, and J.-H. Jacobsen, "Understanding and mitigating exploding inverses in invertible neural networks," in *International Conference on Artificial Intelligence and Statistics*, pp. 1792–1800, PMLR, 2021.

[54] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3730–3738, 2015.

[55] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying mmd gans," in *International Conference on Learning Representations (ICLR)*, 2018.

[56] M. S. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly, "Assessing generative models via precision and recall," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[57] M. Grcić, I. Grubišić, and S. Šegvić, "Densely connected normalizing flows," *Advances in Neural Information Processing Systems*, vol. 34, pp. 23968–23982, 2021.

[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[59] G. Parmar, R. Zhang, and J.-Y. Zhu, "On aliased resizing and surprising subtleties in gan evaluation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.