Journal of Machine Learning Research 23 (2022) 1-33 Submitted 6/21; Revised 12/21; Published 1/22

Cascaded Diffusion Models for High Fidelity Image Generation

Jonathan Ho* JONATHANHO@GOOGLE.COM
Chitwan Saharia* SAHARIAC@GOOGLE.COM
William Chan WILLIAMCHAN@GOOGLE.COM
David J. Fleet DAVIDFLEET@GOOGLE.COM
Mohammad Norouzi MNOROUZI@GOOGLE.COM
Tim Salimans SALIMANS@GOOGLE.COM

Google, 1600 Amphitheatre Parkway, Mountain View, CA 94043

Editor: Mingyuan Zhou

Abstract

We show that cascaded diffusion models are capable of generating high fidelity images on the class-conditional ImageNet generation benchmark, without any assistance from auxiliary image classifiers to boost sample quality. A cascaded diffusion model comprises a pipeline of multiple diffusion models that generate images of increasing resolution, beginning with a standard diffusion model at the lowest resolution, followed by one or more super-resolution diffusion models that successively upsample the image and add higher resolution details. We find that the sample quality of a cascading pipeline relies crucially on conditioning augmentation, our proposed method of data augmentation of the lower resolution conditioning inputs to the super-resolution models. Our experiments show that conditioning augmentation prevents compounding error during sampling in a cascaded model, helping us to train cascading pipelines achieving FID scores of 1.48 at 64×64, 3.52 at 128×128 and 4.88 at 256×256 resolutions, outperforming BigGAN-deep, and classification accuracy scores of 63.02% (top-1) and 84.06% (top-5) at 256×256, outperforming VQ-VAE-2.

Keywords: generative models, diffusion models, score matching, iterative refinement, super-resolution

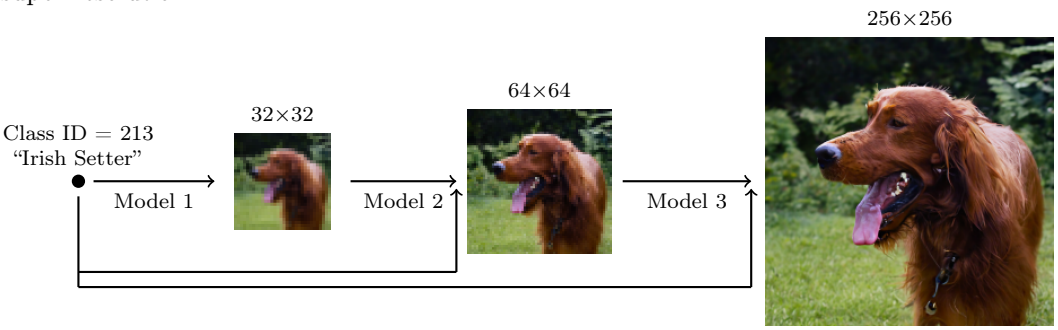


Figure 1: A cascaded diffusion model comprising a base model and two super-resolution models.

*. Equal contribution

Figure 2: Selected synthetic 256×256 ImageNet samples.

1. Introduction

Diffusion models (Sohl-Dickstein et al., 2015) have recently been shown to be capable of synthesizing high quality images and audio (Chen et al., 2021; Ho et al., 2020; Kong et al., 2021; Song and Ermon, 2020): an application of machine learning that has long been dominated by other classes of generative models such as autoregressive models, GANs, VAEs, and flows (Brock et al., 2019; Dinh et al., 2017; Goodfellow et al., 2014; Ho et al., 2019; Kingma and Dhariwal, 2018; Kingma and Welling, 2014; Razavi et al., 2019; van den Oord et al., 2016a,b, 2017). Most previous work on diffusion models demonstrating high quality samples has focused on data sets of modest size, or data with strong conditioning signals. Our goal is to improve the sample quality of diffusion models on large high-fidelity data sets for which no strong conditioning information is available. To showcase the capabilities of the original diffusion formalism, we focus on simple, straightforward techniques to improve the sample quality of diffusion models; for example, we avoid using extra image classifiers to boost sample quality metrics (Dhariwal and Nichol, 2021; Razavi et al., 2019).

Our key contribution is the use of *cascades* to improve the sample quality of diffusion models on class-conditional ImageNet . Here, cascading refers to a simple technique to model high resolution data by learning a pipeline of separately trained models at multiple resolutions; a base model generates low resolution samples, followed by super-resolution models that upsample low resolution samples into high resolution samples. Sampling from a cascading pipeline occurs sequentially, first sampling from the low resolution base model, followed by sampling from super-resolution models in order of increasing resolution. While any type of generative model could be used in a cascading pipeline (*e.g.,* Menick and Kalchbrenner, 2019; Razavi et al., 2019), here we restrict ourselves to diffusion models. Cascading has been shown in recent prior work to improve the sample quality of diffusion models (Saharia et al., 2021; Nichol and Dhariwal, 2021); our work here concerns the improvement of diffusion cascading pipelines to attain the best possible sample quality.

The simplest and most effective technique we found to improve cascading diffusion pipelines is to apply strong data augmentation to the conditioning input of each super-resolution model. We refer to this technique as *conditioning augmentation*. In our experiments, conditioning augmentation is crucial for our cascading pipelines to generate high quality samples at the highest resolution. With this approach we attain FID scores on class-conditional ImageNet generation that are better than BigGAN-Deep (Brock et al., 2019) at any truncation value, and classification accuracy scores that are better than VQ-VAE-2 (Razavi et al., 2019). We empirically find that conditioning augmentation is effective because it alleviates compounding error in cascading pipelines due to train-test mismatch, sometimes referred to as exposure bias in the sequence modeling literature (Bengio et al., 2015; Ranzato et al., 2016).

The key contributions of this paper are as follows:

- We show that our **C**ascaded **D**iffusion **M**odels (CDM) yield high fidelity samples superior to BigGAN-deep (Brock et al., 2019) and VQ-VAE-2 (Razavi et al., 2019) in terms of FID score (Heusel et al., 2017) and classification accuracy score (Ravuri and Vinyals, 2019), the latter by a large margin. We achieve these results with pure generative models that are not combined with any classifier.

- We introduce conditioning augmentation for our super-resolution models, and find it critical towards achieving high sample fidelity. We perform an in-depth exploration of augmentation policies, and find Gaussian augmentation to be a key ingredient for low resolution upsampling, and Gaussian blurring for high resolution upsampling. We also show how to efficiently train models amortized over varying levels of conditioning augmentation to enable post-training hyperparameter search for optimal sample quality.

Section 2 reviews recent work on diffusion models. Section 3 describes the most effective types of conditioning augmentation that we found for class-conditional ImageNet generation. Section 4 contains our sample quality results, ablations, and experiments on additional datasets. Appendix A contains extra samples and Appendix B contains details on hyperparameters and architectures. High resolution figures and additional supplementary material can be found at `https://cascaded-diffusion.github.io/`.

2. Background

We begin with background on diffusion models, their extension to conditional generation, and their associated neural network architectures.

2.1 Diffusion Models

A diffusion model (Sohl-Dickstein et al., 2015; Ho et al., 2020) is defined by a forward process that gradually destroys data $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ over the course of T timesteps

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \qquad q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

and a parameterized reverse process $p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_{0:T})\, d\mathbf{x}_{1:T}$, where

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)).$$

The forward process hyperparameters β_t are set so that $\mathbf{x}_T$ is approximately distributed according to a standard normal distribution, so $p(\mathbf{x}_T)$ is set to a standard normal prior as well. The reverse process is trained to match the joint distribution of the forward process by optimizing the evidence lower bound (ELBO) $-L_\theta(\mathbf{x}_0) \leq \log p_\theta(\mathbf{x}_0)$:

$$L_\theta(\mathbf{x}_0) = \mathbb{E}_q \left[L_T(\mathbf{x}_0) + \sum_{t>1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) \right] \tag{1}$$

where $L_T(\mathbf{x}_0) = D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))$. The forward process posteriors $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ and marginals $q(\mathbf{x}_t|\mathbf{x}_0)$ are Gaussian, and the KL divergences in the ELBO can be calculated in closed form. Thus it is possible to train the diffusion model by taking stochastic gradient steps on random terms of Eq. (1). As previously suggested (Ho et al., 2020; Nichol and Dhariwal, 2021), we use the reverse process parameterizations

$$\begin{aligned} \boldsymbol{\mu}_\theta(\mathbf{x}_t, t) &= \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) \\ \Sigma_\theta^{ii}(\mathbf{x}_t, t) &= \exp(\log \tilde{\beta}_t + (\log \beta_t - \log \tilde{\beta}_t) v_\theta^i(\mathbf{x}_t, t)) \end{aligned}$$

where $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$, and $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \beta_t$.

Sample quality can be improved, at the cost of log likelihood, by optimizing modified losses instead of the ELBO. The particular form of the modified loss depends on whether we are learning $\boldsymbol{\Sigma}_\theta$ or treating it as a fixed hyperparameter (and whether $\boldsymbol{\Sigma}_\theta$ is learned is itself considered a hyperparameter choice that we set experimentally). For the case of non-learned $\boldsymbol{\Sigma}_\theta$, we use the simplified loss

$$L_{\mathrm{simple}}(\theta) = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t \sim \mathcal{U}(\{1, \ldots, T\})} \left[\left\| \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) - \boldsymbol{\epsilon} \right\|^2 \right]$$

which is a weighted form of the ELBO that resembles denoising score matching over multiple noise scales (Ho et al., 2020; Song and Ermon, 2019). For the case of learned $\boldsymbol{\Sigma}_\theta$, we employ a hybrid loss (Nichol and Dhariwal, 2021) implemented using the expression

$$L_{\mathrm{hybrid}}(\theta) = L_{\mathrm{simple}}(\theta) + \lambda L_{\mathrm{vb}}(\theta)$$

where $L_{\mathrm{vb}} = \mathbb{E}_{\mathbf{x}_0}[L_\theta(\mathbf{x}_0)]$ and a stop-gradient is applied to the $\boldsymbol{\epsilon}_\theta$ term inside L_θ. Optimizing this hybrid loss has the effect of simultaneously learning $\boldsymbol{\mu}_\theta$ using L_{simple} and learning $\boldsymbol{\Sigma}_\theta$ using the ELBO.

2.2 Conditional Diffusion Models

In the conditional generation setting, the data $\mathbf{x}_0$ has an associated conditioning signal $\mathbf{c}$, for example a label in the case of class-conditional generation, or a low resolution image in the case of super-resolution (Saharia et al., 2021; Nichol and Dhariwal, 2021). The goal is then

to learn a conditional model $p_\theta(\mathbf{x}_0|\mathbf{c})$. To do so, we modify the diffusion model to include $\mathbf{c}$ as input to the reverse process:

$$p_\theta(\mathbf{x}_{0:T}|\mathbf{c}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c}), \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c}) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t, \mathbf{c}), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t, \mathbf{c}))$$

$$L_\theta(\mathbf{x}_0|\mathbf{c}) = \mathbb{E}_q\left[L_T(\mathbf{x}_0) + \sum_{t>1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c})) - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1, \mathbf{c})\right].$$

The data and conditioning signal $(\mathbf{x}_0, \mathbf{c})$ are sampled jointly from the data distribution, now called $q(\mathbf{x}_0, \mathbf{c})$, and the forward process $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ remains unchanged. The only modification that needs to be made is to inject $\mathbf{c}$ as a extra input to the neural network function approximators: instead of $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ we now have $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t, \mathbf{c})$, and likewise for $\boldsymbol{\Sigma}_\theta$. The particular architectural choices for injecting these extra inputs depends on the type of the conditioning $\mathbf{c}$, as described next.

2.3 Architectures

The current best architectures for image diffusion models are U-Nets (Ronneberger et al., 2015; Salimans et al., 2017), which are a natural choice to map corrupted data $\mathbf{x}_t$ to reverse process parameters $(\boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}_\theta)$ that have the same spatial dimensions as $\mathbf{x}_t$. Scalar conditioning, such as a class label or a diffusion timestep t, is provided by adding embeddings into intermediate layers of the network (Ho et al., 2020). Lower resolution image conditioning is provided by channelwise concatenation of the low resolution image, processed by bilinear or bicubic upsampling to the desired resolution, with the reverse process input $\mathbf{x}_t$, as in the SR3 (Saharia et al., 2021) and Improved DDPM (Nichol and Dhariwal, 2021) models. See Fig. 3 for an illustration of the SR3-based architecture that we use in this work.

3. Conditioning Augmentation in Cascaded Diffusion Models

Suppose $\mathbf{x}_0$ is high resolution data and $\mathbf{z}_0$ is its low resolution counterpart. We use the term *cascading pipeline* to refer to a sequence of generative models. At the low resolution we have a diffusion model $p_\theta(\mathbf{z}_0)$, and at the high resolution, a super-resolution diffusion model $p_\theta(\mathbf{x}_0|\mathbf{z}_0)$. The cascading pipeline forms a latent variable model for high resolution data; i.e., $p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_0|\mathbf{z}_0)p_\theta(\mathbf{z}_0)\, d\mathbf{z}_0$. It is straightforward to extend this to more than two resolutions. It is also straightforward to condition an entire cascading pipeline on class information or other conditioning information $\mathbf{c}$: the models take on the form $p_\theta(\mathbf{z}_0|\mathbf{c})$ and $p_\theta(\mathbf{x}_0|\mathbf{z}_0, \mathbf{c})$, each using the conditioning mechanism described in Section 2.2. An example cascading pipeline is depicted in Fig. 4.

Cascading pipelines have been shown to be useful with other generative model families (Menick and Kalchbrenner, 2019; Razavi et al., 2019). A major benefit to training a cascading pipeline over training a standard model at the highest resolution is that most of the modeling capacity can be dedicated to low resolutions, which empirically are the most important for sample quality, and training and sampling at low resolutions tends to be the most computationally efficient. In addition, cascading allows the individual models to be trained independently, and architecture choices can be tuned at each specific resolution for the best performance of the entire pipeline.

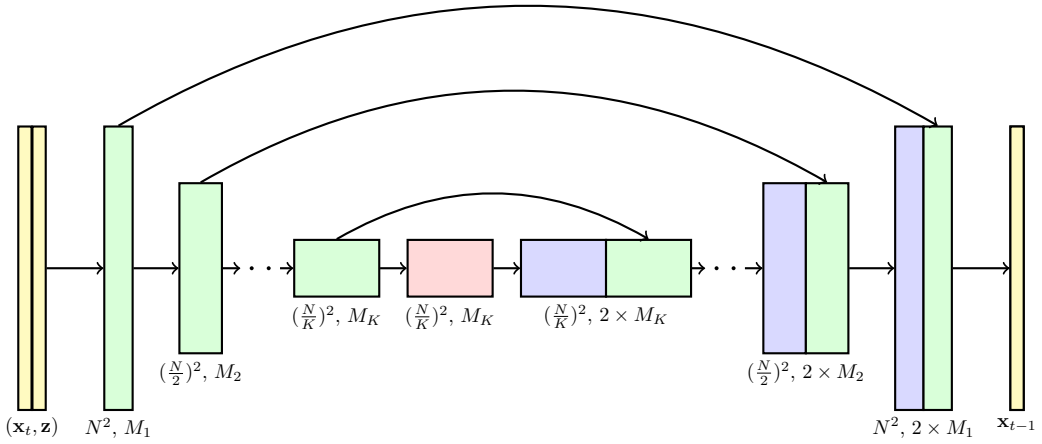


Figure 3: The U-Net architecture used in each model of a CDM pipeline. The first model is a class-conditional diffusion model that receives the noisy image $\mathbf{x}_t$ and the class label y and as input. (The class label y and timestep t are injected into each block as an embedding, not depicted here). The remaining models in the pipeline are class-conditional super-resolution models that receive $\mathbf{x}_t$, y, and an additional upsampled low-resolution image $\mathbf{z}$ as input. The downsampling/upsampling blocks adjust the image input resolution $N \times N$ by a factor of 2 through each of the K blocks. The channel count at each block is specified using channel multipliers M_1, M_2, ..., M_K, and the upsampling pass has concatenation skip connections to the downsampling pass.

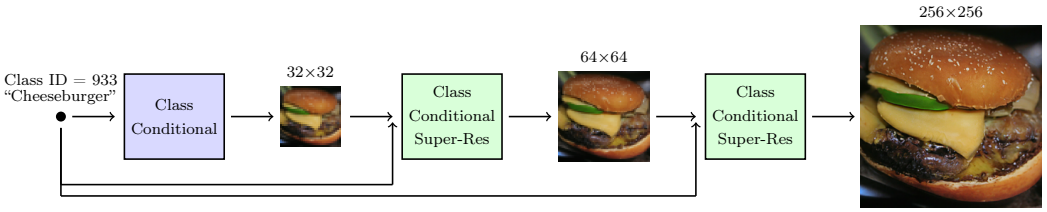


Figure 4: Detailed CDM pipeline for generation of class conditional 256×256 images. The first model is a class-conditional diffusion model, and it is followed by a sequence of two class-conditional super-resolution diffusion models. Each model has a U-Net architecture as depicted in Fig. 3.

The most effective technique we found to improve the sample quality of cascading pipelines is to train each super-resolution model using data augmentation on its low resolution input. We refer to this general technique as *conditioning augmentation*. At a high level, for some super-resolution model $p_\theta(\mathbf{x}_0|\mathbf{z})$ from a low resolution image $\mathbf{z}$ to a high resolution image $\mathbf{x}_0$, conditioning augmentation refers to applying some form of data augmentation to $\mathbf{z}$. This augmentation can take any form, but what we found most effective at low resolutions is adding Gaussian noise (forward process noise), and for high resolutions, randomly applying Gaussian blur to $\mathbf{z}$. In some cases, we found it more practical to train super-resolution models amortized over the strength of conditioning augmentation and pick the best strength in a post-training hyperparameter search for optimal sample quality. Details on conditioning

augmentation and its realization during training and sampling are given in the following sections.

3.1 Blurring Augmentation

One simple instantiation of conditioning augmentation is augmentation of $\mathbf{z}$ by blurring. We found this to be most effective for upsampling to images with resolution 128×128 and 256×256. More specifically, we apply a Gaussian filter of size k and sigma σ to obtain $\mathbf{z}_b$. We use a filter size of $k = (3, 3)$ and randomly sample σ from a fixed range during training. We perform hyper-parameter search to find the range for σ. During training, we apply this blurring augmentation to 50% of the examples. During inference, no augmentation is applied to low resolution inputs. We explored applying different amounts of blurring augmentations during inference, but did not find it helpful in initial experiments.

3.2 Truncated Conditioning Augmentation

Here we describe what we call *truncated conditioning augmentation*, a form of conditioning augmentation that requires a simple modification to the training and architecture of the super-resolution models, but no change to the low resolution model at the initial stage of the cascade. We found this method to be most useful at resolutions smaller than 128×128. Normally, generating a high resolution sample $\mathbf{x}_0$ involves first generating $\mathbf{z}_0$ from the low resolution model $p_\theta(\mathbf{z}_0)$, then feeding that result into the super-resolution model $p_\theta(\mathbf{x}_0|\mathbf{z}_0)$. In other words, generating a high resolution sample is performed using ancestral sampling from the latent variable model

$$p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_0|\mathbf{z}_0)p_\theta(\mathbf{z}_0)\, d\mathbf{z}_0 = \int p_\theta(\mathbf{x}_0|\mathbf{z}_0)p_\theta(\mathbf{z}_{0:T})\, d\mathbf{z}_{0:T}.$$

(For simplicity, we have assumed that the low resolution and super-resolution models both use the same number of timesteps T.) Truncated conditioning augmentation refers to truncating the low resolution reverse process to stop at timestep $s > 0$, instead of 0; i.e.,

$$p^s_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_0|\mathbf{z}_s)p_\theta(\mathbf{z}_s)\, d\mathbf{z}_s = \int p_\theta(\mathbf{x}_0|\mathbf{z}_s)p_\theta(\mathbf{z}_{s:T})\, d\mathbf{z}_{s:T}. \tag{2}$$

The base model is now $p_\theta(\mathbf{z}_s) = \int p_\theta(\mathbf{z}_{s:T})d\mathbf{z}_{s+1:T}$, and the super-resolution model is now $p_\theta(\mathbf{x}_0|\mathbf{z}_s) = \int p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{z}_s)d\mathbf{x}_{1:T}$, where

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{z}_s) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t, \mathbf{z}_s, s), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t, \mathbf{z}_s, s)).$$

The reason truncating the low resolution reverse process is a form of data augmentation is that the training procedure for $p_\theta(\mathbf{x}_0|\mathbf{z}_s)$ involves conditioning on noisy $\mathbf{z}_s \sim q(\mathbf{z}_s|\mathbf{z}_0)$, which, up to scaling, is $\mathbf{z}_0$ augmented with Gaussian noise. To be more precise about training a cascading pipeline with truncated conditioning augmentation, let us examine the ELBO for $p^s_\theta(\mathbf{x}_0)$ in Eq. (2). We can treat $p^s_\theta(\mathbf{x}_0)$ as a VAE with a diffusion model prior, a diffusion model decoder, and the approximate posterior

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

which runs forward processes independently on a low and high resolution pair. The ELBO is

$$-\log p_\theta^s(\mathbf{x}_0) \le \mathbb{E}_q\left[L_T(\mathbf{z}_0) + \sum_{t>s} D_{\mathrm{KL}}(q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{z}_0) \parallel p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)) - \log p_\theta(\mathbf{x}_0|\mathbf{z}_s)\right],$$

where $L_T(\mathbf{z}_0) = D_{\mathrm{KL}}(q(\mathbf{z}_T|\mathbf{z}_0) \parallel p(\mathbf{z}_T))$. Note that the sum over t is truncated at s, and the decoder $p_\theta(\mathbf{x}_0|\mathbf{z}_s)$ is the super-resolution model conditioned on $\mathbf{z}_s$. The decoder itself has an ELBO of the form $-\log p_\theta(\mathbf{x}_0|\mathbf{z}_s) \le L_\theta(\mathbf{x}_0|\mathbf{z}_s)$, where

$$L_\theta(\mathbf{x}_0|\mathbf{z}_s) = \mathbb{E}_q\left[L_T(\mathbf{x}_0) + \sum_{t>1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{z}_s)) - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1, \mathbf{z}_s)\right].$$

Thus we have an ELBO for the combined model

$$-\log p_\theta^s(\mathbf{x}_0) \le \mathbb{E}_q\left[L_T(\mathbf{z}_0) + \sum_{t>s} D_{\mathrm{KL}}(q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{z}_0) \parallel p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)) + L_\theta(\mathbf{x}_0|\mathbf{z}_s)\right]. \tag{3}$$

It is apparent that optimizing Eq. (3) trains the low and high resolution models separately. For a fixed value of s, the low resolution process is trained up to the truncation timestep s, and the super-resolution model is trained on a conditioning signal corrupted using the low resolution forward process stopped at timestep s.

In practice, since we pursue sample quality as our main objective, we do not use these ELBO expressions directly when training models with learnable reverse process variances. Rather, we train on the "simple" unweighted loss or the hybrid loss described in Section 2, and the particular loss we use is considered a hyperparameter reported in Appendix B.

We would like to search over multiple values of s to select for the best sample quality. To make this search practical, we avoid retraining models by amortizing a single super-resolution model over uniform random s at training time. Because each possible truncation time corresponds to a distinct super-resolution task, the super-resolution model for $\boldsymbol{\mu}_\theta$ and $\boldsymbol{\Sigma}_\theta$ must take $\mathbf{z}_s$ as input along with s, and this can be accomplished using a single network with an extra time embedding input for s. We leave the low resolution model training unchanged, because the standard diffusion training procedure already trains with random s. The complete training procedure for a two-stage cascading pipeline is listed in Algorithm 1.

3.3 Non-truncated Conditioning Augmentation

Another form of conditioning augmentation, which we call *non-truncated conditioning augmentation*, uses the same model modifications and training procedure as truncated conditioning augmentation (Section 3.2). The only difference is at sampling time. Instead of truncating the low resolution reverse process, in non-truncated conditioning augmentation we always sample $\mathbf{z}_0$ using the full, non-truncated low resolution reverse process; then we corrupt $\mathbf{z}_0$ using the forward process into $\mathbf{z}'_s \sim q(\mathbf{z}_s|\mathbf{z}_0)$ and feed the corrupted $\mathbf{z}'_s$ into the super-resolution model.

The main advantage of non-truncated conditioning augmentation over truncated conditioning augmentation is a practical one during the search phase over s. In the case of truncated augmentation, if we want to run the super-resolution model over all s in parallel, we must store all low resolution samples $\mathbf{z}_s$ for all values of s considered. In the case of

Algorithm 1 Training a two-stage CDM with Gaussian conditioning augmentation

repeat ▷ Train base model
  $(\mathbf{z}_0, \mathbf{c}) \sim p(\mathbf{z}, \mathbf{c})$ ▷ Sample low-resolution image and label
  $t \sim \mathcal{U}(\{1, \dots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{z}_t = \sqrt{\bar{\alpha}_t}\mathbf{z}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}$
  $\theta \leftarrow \theta - \eta\nabla_\theta \left\|\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t, \mathbf{c}) - \boldsymbol{\epsilon}\right\|^2$ ▷ Simple loss (can be replaced with a hybrid loss)
until converged
repeat ▷ Train super-resolution model (in parallel with the base model)
  $(\mathbf{x}_0, \mathbf{z}_0, \mathbf{c}) \sim p(\mathbf{x}, \mathbf{z}, \mathbf{c})$ ▷ Sample low- and high-resolution images and label
  $s, t \sim \mathcal{U}(\{1, \dots, T\})$
  $\boldsymbol{\epsilon}_\mathbf{z}, \boldsymbol{\epsilon}_\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ Note: $\boldsymbol{\epsilon}_\mathbf{z}, \boldsymbol{\epsilon}_\mathbf{x}$ should have the same shapes as $\mathbf{z}_0, \mathbf{x}_0$, respectively
  $\mathbf{z}_t = \sqrt{\bar{\alpha}_s}\mathbf{z}_0 + \sqrt{1 - \bar{\alpha}_s}\boldsymbol{\epsilon}_\mathbf{z}$ ▷ Apply Gaussian conditioning augmentation
  $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}_\mathbf{x}$
  $\theta \leftarrow \theta - \eta\nabla_\theta \left\|\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \mathbf{z}_s, s, \mathbf{c}) - \boldsymbol{\epsilon}_\mathbf{x}\right\|^2$
until converged

Algorithm 2 Sampling from a two-stage CDM with Gaussian conditioning augmentation

Require: $\mathbf{c}$: class label
Require: s: conditioning augmentation truncation time
$\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
if using truncated conditioning augmentation **then**
  for $t = T, \dots, s + 1$ **do**
    $\mathbf{z}_{t-1} \sim p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{c})$
  end for
else
  for $t = T, \dots, 1$ **do**
    $\mathbf{z}_{t-1} \sim p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{c})$
  end for
  $\mathbf{z}_s \sim q(\mathbf{z}_s|\mathbf{z}_0)$ ▷ Overwrite previously sampled value of $\mathbf{z}_s$
end if
$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ **do**
  $\mathbf{x}_{t-1} \sim p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{z}_s, \mathbf{c})$
end for
return $\mathbf{x}_0$

non-truncated augmentation, we need to store the low resolution samples just once, since sampling $\mathbf{z}'_s \sim q(\mathbf{z}_s|\mathbf{z}_0)$ is computationally inexpensive. These sampling procedures are listed in Algorithm 2.

Truncated and non-truncated conditioning augmentation should perform similarly because $\mathbf{z}_s$ and $\mathbf{z}'_s$ should have similar marginal distributions if the low resolution model is trained well enough. Indeed, in Section 4.3, we empirically find that sample quality metrics are similar for both truncated and non-truncated conditioning augmentation.

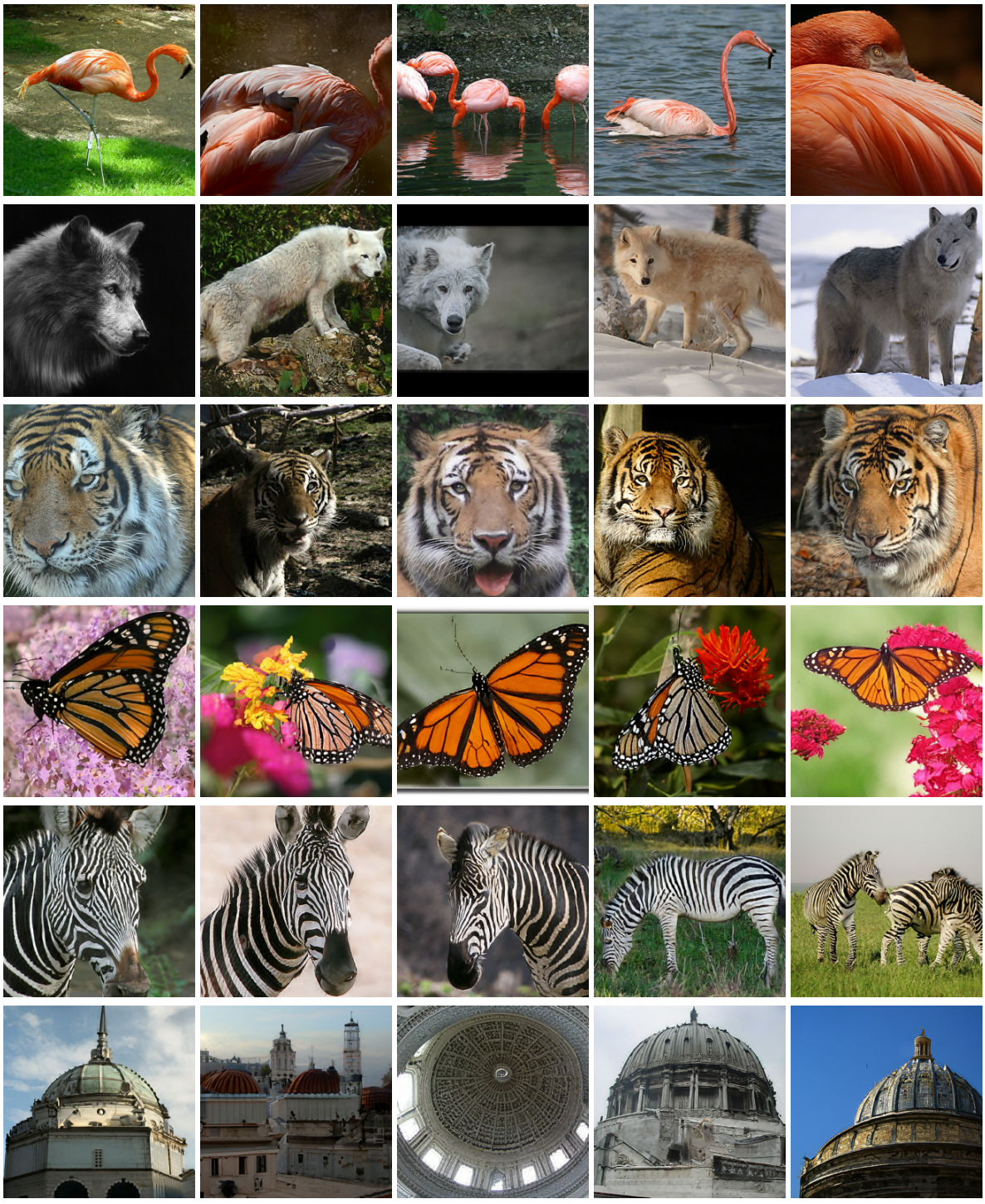


Figure 5: Classwise Synthetic 256×256 ImageNet images. Each row represents a specific ImageNet class. Classes from top to bottom - Flamingo (130), White Wolf (270), Tiger (292), Monarch Butterfly (323), Zebra (340) and Dome (538).

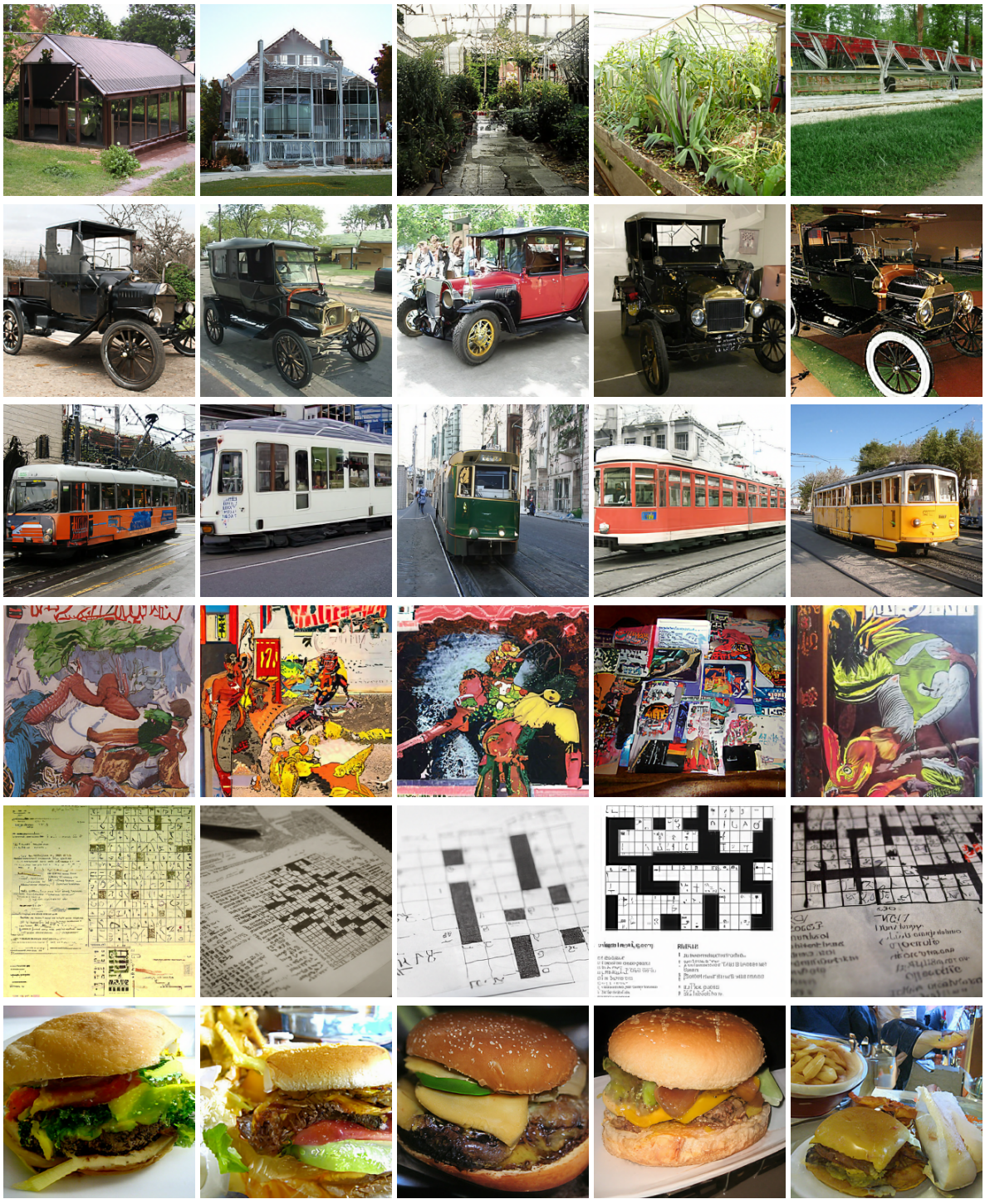


Figure 6: Classwise Synthetic 256×256 ImageNet images. Each row represents a specific ImageNet class. Classes from top to bottom - Greenhouse (580), Model T (661), Streetcar (829), Comic Book (917), Crossword Puzzle (918), Cheeseburger (933).

4. Experiments

We designed experiments to improve the sample quality metrics of cascaded diffusion models on class-conditional ImageNet generation. Our cascading pipelines consist of class-conditional diffusion models at all resolutions, so class information is injected at all resolutions: see Fig. 4. Our final ImageNet results are described in Section 4.1.

To give insight into our cascading pipelines, we begin with improvements on a baseline non-cascaded model at the 64×64 resolution (Section 4.2), then we show that cascading up to 64×64 improves upon our best non-cascaded 64×64 model, but only in conjunction with conditioning augmentation. We also show that truncated and non-truncated conditioning augmentation perform equally well (Section 4.3), and we study random Gaussian blur augmentation to train super-resolution models to resolutions of 128×128 and 256×256 (Section 4.4). Finally, we verify that conditioning augmentation is also effective on the LSUN dataset (Yu et al., 2015) and therefore is not specific to ImageNet (Section 4.5).

We cropped and resized the ImageNet dataset (Russakovsky et al., 2015) in the same manner as BigGAN (Brock et al., 2019). We report Inception scores using the standard practice of generating 50k samples and calculating the mean and standard deviation over 10 splits (Salimans et al., 2016). Generally, throughout our experiments, we selected models and performed early stopping based on FID score calculated over 10k samples, but all reported FID scores are calculated over 50k samples for comparison with other work (Heusel et al., 2017). The FID scores we used for model selection and reporting model performance are calculated against training set statistics according to common practice, but since this can be seen as overfitting on the performance metric, we additionally report model performance using FID scores calculated against validation set statistics. We also report results on Classification Accuracy Score (CAS), which was proposed by Ravuri and Vinyals (2019) due to their findings that non-GAN models may score poorly on FID and IS despite generating visually appealing samples and that FID and IS are not correlated (sometimes anti-correlated) with performance on downstream tasks.

4.1 Main Cascading Pipeline Results

Table 1a reports the main results on the cascaded diffusion model (*CDM*), for the 64×64, 128×128, and 256×256 ImageNet dataset resolutions, along with baselines. CDM outperforms BigGAN-deep in terms of FID score on the image resolutions considered, but GANs perform better in terms of Inception score when their truncation parameter is optimized for Inception score (Brock et al., 2019). We also outperform concurrently released diffusion models that do not use classifier guidance to boost sample quality scores (Dhariwal and Nichol, 2021). See Fig. 8 for a qualitative assessment of sample quality and diversity compared to VQ-VAE-2 (Razavi et al., 2019) and BigGAN-deep (Brock et al., 2019), and see Figs. 5 and 6 for examples of generated images.

Table 1b reports the results on Classification Accuracy Score (CAS) (Ravuri and Vinyals, 2019) for our models at the 128×128 and 256×256 resolutions. We find that CDM outperforms VQ-VAE-2 and BigGAN-deep at both resolutions by a significant margin on the CAS metric, suggesting better potential performance on downstream tasks. Figure 7 compares class-wise classification accuracy scores between classifiers trained on real training data, and CDM samples. The CDM classifier outperforms real data on 96 classes compared to 6 and 31

classes by BigGAN-deep and VQ-VAE-2 respectively. We also show samples from classes with best and worst accuracy scores in Appendix Figure 11 and 12.

Our cascading pipelines are structured as a 32×32 base model, a 32×32→64×64 super-resolution model, followed by 64×64→128×128 or 64×64→256×256 super-resolution models. Models at 32×32 and 64×64 resolutions use 4000 diffusion timesteps and architectures similar to DDPM (Ho et al., 2020) and Improved DDPM (Nichol and Dhariwal, 2021). Models at 128×128 and 256×256 resolutions use 100 sampling steps, determined by post-training hyperparameter search (Section 4.4), and they use architectures similar to SR3 (Saharia et al., 2021). All base resolution and super-resolution models are conditioned on class labels. See Appendix B for details.

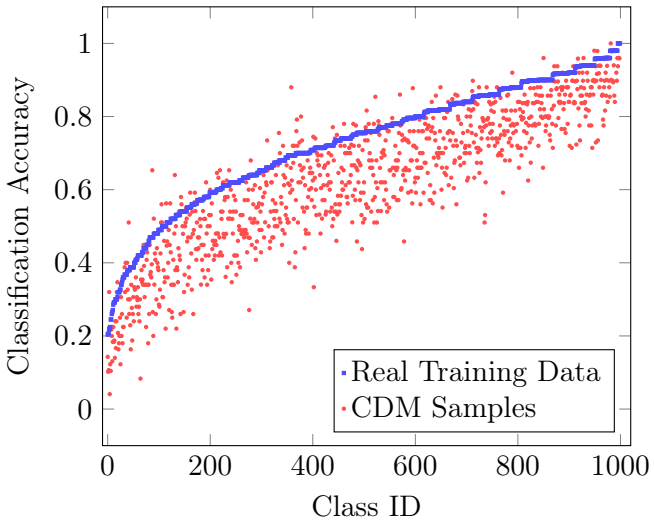


Figure 7: Classwise Classification Accuracy Score comparison between real data (blue) and generated data (red) at the 256×256 resolution. Accompanies Table 1b.

4.2 Baseline Model Improvements

To set a strong baseline for class-conditional ImageNet generation at the 64×64 resolution, we reproduced and improved upon a 4000 timestep non-cascaded 64×64 class-conditional diffusion model from Improved DDPM (Nichol and Dhariwal, 2021). Our reimplementation used dropout and was trained longer than reported by Nichol and Dhariwal; we found that adding dropout generally slowed down convergence of FID and Inception scores, but improved their best values over the course of a longer training period. We further improved the training set FID score and Inception score by adding noise to the trained model's samples using the forward process to the 2000 timestep point, then restarting the reverse process from that point. See Table 2a for the resulting sample quality metrics.

Model	FID vs train	FID vs validation	IS
32×32 resolution			
CDM (ours)	1.11	1.99	26.01 ± 0.59
64×64 resolution			
BigGAN-deep, by (Dhariwal and Nichol, 2021)	4.06		
Improved DDPM (Nichol and Dhariwal, 2021)	2.92		
ADM (Dhariwal and Nichol, 2021)	2.07		
CDM (ours)	**1.48**	2.48	67.95 ± 1.97
128×128 resolution			
BigGAN-deep (Brock et al., 2019)	5.7		124.5
BigGAN-deep, max IS (Brock et al., 2019)	25		**253**
LOGAN (Wu et al., 2019)	**3.36**		148.2
ADM (Dhariwal and Nichol, 2021)	5.91		
CDM (ours)	3.52	3.76	128.80 ± 2.51
256×256 resolution			
BigGAN-deep (Brock et al., 2019)	6.9		171.4
BigGAN-deep, max IS (Brock et al., 2019)	27		**317**
VQ-VAE-2 (Razavi et al., 2019)	31.11		
Improved DDPM (Nichol and Dhariwal, 2021)	12.26		
SR3 (Saharia et al., 2021)	11.30		
ADM (Dhariwal and Nichol, 2021)	10.94		100.98
ADM+upsampling (Dhariwal and Nichol, 2021)	7.49		127.49
CDM (ours)	**4.88**	4.63	158.71 ± 2.26

(a) Class-conditional ImageNet sample quality results for classifier guidance-free methods

Model	Top-1 Accuracy	Top-5 Accuracy
128×128 resolution		
Real	68.82%	88.79%
BigGAN-deep (Brock et al., 2019)	40.64%	64.44%
HAM (De Fauw et al., 2019)	54.05%	77.33%
CDM (ours)	**59.84%**	**81.79%**
256×256 resolution		
Real	73.09%	91.47%
BigGAN-deep (Brock et al., 2019)	42.65%	65.92%
VQ-VAE-2 (Razavi et al., 2019)	54.83%	77.59%
CDM (ours)	**63.02%**	**84.06%**

(b) Classification Accuracy Score (CAS) results

Table 1: Main results. Numbers are bolded only when at least two are available for comparison. CAS for real data and other models are from Ravuri and Vinyals (2019).

CDM (ours) VQ-VAE-2 BigGAN-deep

Figure 8: Comparing the quality and diversity of model samples in selected 256×256 ImageNet classes {Tench(0), Goldfish(1) and Ostrich(9)}. VQVAE-2 and BigGAN samples are taken from Razavi et al. (2019).

Model	FID vs train	FID vs validation	IS
Improved DDPM (Nichol and Dhariwal, 2021)	2.92		
Our reimplementation	2.44	**2.91**	49.81 ± 0.65
+ more sampling steps	**2.35**	**2.91**	**52.72 ± 1.15**

(a) Improvements to a non-cascaded baseline

Conditioning	FID vs train	FID vs validation	IS
No cascading	2.35	2.91	52.72 ± 1.15
16×16→64×64 cascading			
$s = 0$	6.02	5.84	35.59 ± 1.19
$s = 101$	3.41	3.67	44.72 ± 1.12
$s = 1001$	**2.13**	**2.79**	**54.47 ± 1.05**

(b) Small-scale ablation comparing no cascading to 16×16→64×64 cascading

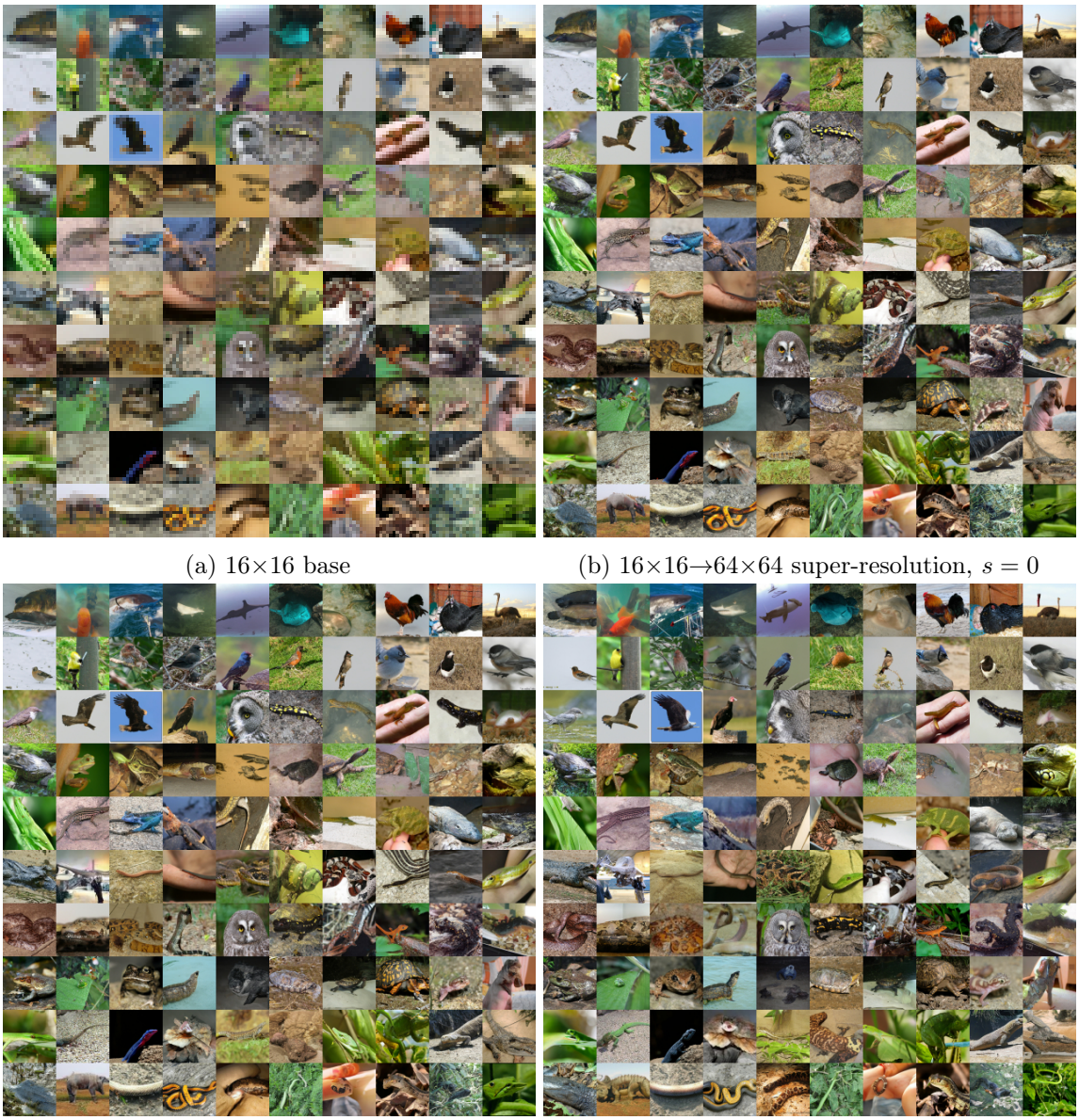
Table 2: 64×64 ImageNet sample quality: ablations.

4.3 Conditioning Augmentation Experiments up to 64×64

Building on our reimplementation in Section 4.2, we verify in a small scale experiment that cascading improves sample quality at the 64×64 resolution. We train a two-stage cascading pipeline that comprises a 16×16 base model and a 16×16→64×64 super-resolution model. The super-resolution model architecture is identical to the best 64×64 non-cascaded baseline model in Section 4.2, except for the trivial modification of adding in the low resolution image conditioning information by channelwise concatenation at the input (see Section 2).

See Table 2b and Fig. 9 for the results of this 16×16→64×64 cascading pipeline. Interestingly, we find that without conditioning augmentation, the cascading pipeline attains lower sample quality than the non-cascaded baseline 64×64 model; the FID score, for example, degrades from 2.35 to 6.02. With sufficient conditioning augmentation, however, the sample quality of the cascading pipeline becomes better than the non-cascaded baseline. We train two super-resolution models with non-truncated conditioning augmentation, one at truncation time $s = 101$ and another at $s = 1001$ (we could have amortized both into a single model, but we chose not to do so in this experiment to prevent potential model capacity issues from confounding the results). The first model achieves better sample quality than the non-augmented model but is still worse than the non-cascaded baseline. The second model achieves a FID score of 2.13, outperforming the non-cascaded baseline. Conditioning augmentation is therefore crucial to improve sample quality in this particular cascading pipeline.

To further improve sample quality at the 64×64 resolution, we found it helpful to increase model sizes and to switch to a cascading pipeline starting with a 32×32 base resolution model. We train a 32×32 base model applying random left-right flips, which we found to help 32×32 scores at the expense of longer training times. Training without random flips, the



(a) 16×16 base

(b) 16×16→64×64 super-resolution, $s = 0$

(c) 16×16→64×64 super-resolution, $s = 101$

(d) 16×16→64×64 super-resolution, $s = 1001$

Figure 9: Generated images for varying amounts of conditioning augmentation (non-truncated) in a small-scale 16×16→64×64 pipeline for ablation purposes. Accompanies Table 2b.

best 32×32 resolution FID score is 1.25 at 300k training steps, while training with random flips it is 1.11 at 700k training steps. The 32×32→64×64 super-resolution model is now amortized over the truncation time s by providing s as an extra time embedding input to the network (Section 2), allowing us to perform a more fine grained search over s without retraining the model.

Table 3a displays the resulting sample quality scores for both truncated and non-truncated augmentation. The sample quality metrics improve and then degrade non-monotonically as the truncation time is increased. This indicates that moderate amounts of conditioning augmentation are beneficial to sample quality of the cascading pipeline, but too much conditioning augmentation causes the super-resolution model to behave as a non-conditioned model unable to benefit from cascading. For comparison, Table 3b shows sample quality when the super-resolution model is conditioned on ground truth data instead of generated data. Here, sample quality monotonically degrades as truncation time is increased. Conditioning augmentation is therefore useful precisely when conditioning on generated samples, so as a technique it is uniquely suited to cascading pipelines.

Based on these findings on non-monotonicity of sample quality with respect to truncation time, we conclude that conditioning augmentation works because it alleviates compounding error from a train-test mismatch for the super-resolution model. This occurs when low-resolution model samples are out of distribution compared to the ground truth data on which the super-resolution model is trained. A sufficient amount of Gaussian conditioning augmentation prevents the super-resolution model from attempting to upsample erroneous, out-of-distribution details in the low resolution generated samples. In contrast, sample quality degrades monotonically with respect to truncation time when conditioning the super-resolution model on ground truth data, because there is no such train-test mismatch.

Table 3a additionally shows that truncated and non-truncated conditioning augmentation are approximately equally effective at improving sample quality of the cascading pipeline, albeit at different values of the truncation time parameter. Thus we generally recommend non-truncated augmentation due to its practical benefits described in Section 3.3.

4.4 Experiments at 128×128 and 256×256

While we found Gaussian noise augmentation to be a key ingredient to boost the performance of our cascaded models at low resolutions, our initial experiments with similar augmentations for 128×128 and 256×256 upsampling yielded negative results. Hence, we explore Gaussian blurring augmentation for these resolutions. As mentioned in Section 3.1, we apply the blurring augmentation 50% of the time during training, and use no blurring during inference. We explored other settings (e.g. applying blurring to all training examples, and using varying amounts of blurring during inference), but found this to be most effective in our initial experiments.

Table 4a shows the results of applying Gaussian blur augmentation to the 64×64 → 256×256 super-resolution model. While any amount of blurring helps improve the scores of the 256×256 samples over the baseline model with no blur, we found that sampling $\sigma \sim \mathcal{U}(0.4, 0.6)$ gives the best results. Table 4b shows further improvements from class conditioning, large batch training, and random flip augmentation for the super-resolution model. While we find class conditioning helpful for upsampling at low resolution settings, it is

Conditioning	FID vs train	FID vs validation	IS
No conditioning augmentation (baseline)			
$s = 0$	1.71	2.46	61.34 ± 1.58
Truncated conditioning augmentation			
$s = 251$	1.50	**2.44**	66.76 ± 1.76
$s = 501$	**1.48**	2.48	67.95 ± 1.97
$s = 751$	**1.48**	2.51	**68.48 ± 1.77**
$s = 1001$	1.49	2.51	67.95 ± 1.51
$s = 1251$	1.51	2.54	67.20 ± 1.94
$s = 1501$	1.54	2.56	67.09 ± 1.67
Non-truncated conditioning augmentation			
$s = 251$	1.58	2.50	66.21 ± 1.51
$s = 501$	1.53	2.51	**67.59 ± 1.85**
$s = 751$	**1.48**	2.47	67.48 ± 1.31
$s = 1001$	1.49	2.48	66.51 ± 1.59
$s = 1251$	**1.48**	**2.46**	66.28 ± 1.49
$s = 1501$	1.50	2.47	65.59 ± 0.86

(a) Base model for low resolution conditioning

Conditioning	FID vs train	FID vs validation	IS
Ground truth training data			
$s = 0$	**0.76**	**1.76**	**74.84 ± 1.43**
$s = 251$	0.87	1.85	71.79 ± 0.89
$s = 501$	0.92	1.91	70.68 ± 1.26
$s = 751$	0.95	1.94	69.93 ± 1.40
$s = 1001$	0.98	1.97	69.03 ± 1.26
$s = 1251$	1.03	1.99	67.92 ± 1.65
$s = 1501$	1.11	2.04	66.7 ± 1.21
Ground truth validation data			
$s = 0$	**1.20**	**0.59**	**64.33 ± 1.24**
$s = 251$	1.27	0.96	63.17 ± 1.19
$s = 501$	1.32	1.17	62.65 ± 0.76
$s = 751$	1.38	1.32	62.21 ± 0.94
$s = 1001$	1.42	1.44	61.53 ± 1.39
$s = 1251$	1.47	1.54	60.58 ± 0.93
$s = 1501$	1.53	1.64	60.02 ± 0.84

(b) Ground truth for low resolution conditioning

Table 3: 64×64 ImageNet sample quality: large scale experiment comparing truncated and non-truncated conditioning augmentation for 32×32→64×64 cascading, using amortized truncation time conditioning.

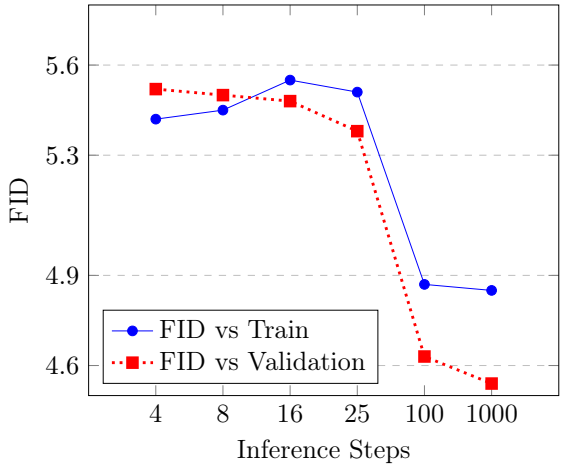


Figure 10: FID on 256×256 images vs inference steps in 64×64 → 256×256 super-resolution.

Blur σ	FID vs train	FID vs validation	IS
$\sigma = 0$ (no blur)	7.26	6.42	134.53 ± 2.97
$\sigma \sim \mathcal{U}(0.4, 0.6)$	**6.18**	**5.57**	**142.71 ± 2.83**
$\sigma \sim \mathcal{U}(0.4, 0.8)$	6.90	6.31	136.57 ± 4.34
$\sigma \sim \mathcal{U}(0.4, 1.0)$	6.35	5.76	141.40 ± 4.34

(a) Gaussian blur noise in conditioning

Model	FID vs train	FID vs validation	IS
Baseline	6.18	5.57	142.71 ± 2.83
+ Class Conditioning	5.75	5.27	152.17 ± 2.29
+ Large Batch Training	5.00	4.71	157.84 ± 2.60
+ Flip LR	**4.88**	**4.63**	**158.71 ± 2.26**

(b) Further improvements on super-resolution

Table 4: 256×256 ImageNet sample quality: experiments on 64×64 → 256×256 super-resolution.

interesting that it still gives a huge boost to the upsampling performance at high resolutions even when the low resolution inputs at 64×64 can be sufficiently informative. We also found increasing the training batch size from 256 to 1024 further improved performance by a significant margin. We also obtain marginal improvements by training the super-resolution model on randomly flipped data.

Since the sampling cost increases quadratically with the target image resolution, we attempt to minimize the number of denoising iterations for our 64×64 → 256×256 and 64×64 → 128×128 super-resolution models. To this end, we train these super-resolution models with continuous noise conditioning, like Saharia et al. (2021) and Chen et al. (2021), and tune the noise schedule for a given number of steps during inference. This tuning is relatively inexpensive as we do not need to retrain the models. We report all results using 100 inference steps for these models. Figure 10 shows FID vs number of inference steps for our 64×64 → 256×256 model. The FID score deteriorates marginally even when using just 4 inference steps. Interestingly, we do not observe any concrete improvement in FID by increasing the number of inference steps from 100 to 1000.

4.5 Experiments on LSUN

While the main results of this work are on class-conditional ImageNet generation, here we study the effectiveness of non-truncated conditioning augmentation for a 64×64→128×128 cascading pipeline on the LSUN Bedroom and Church datasets (Yu et al., 2015) in order to verify that conditioning augmentation is not an ImageNet-specific method. LSUN Bedroom and Church are two separate unconditional datasets that do not have any class labels, so our study here additionally verifies the effectiveness of conditioning augmentation for unconditional generation.

Table 5 displays our LSUN sample quality results, which confirm that a nonzero amount of conditioning augmentation is beneficial to sample quality. (The relatively large FID scores between generated examples and the validation sets are explained by the fact that the LSUN Church and Bedroom validation sets are extremely small, consisting of only 300 examples each.) We observe a similar effect as our ImageNet results in Table 3b: because the super-resolution model is conditioned on base model samples, the sample quality improves then degrades non-monotonically as the truncation time s is increased. See Appendix A for examples of images generated by our LSUN models.

5. Related Work

One way to formulate cascaded diffusion models is to modify the original diffusion formalism of a forward process $q(\mathbf{x}_{0:T})$ at single resolution so that the transition $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ performs downsampling at certain intermediate timesteps, for example at $t \in S \coloneqq \{T/4, 2T/4, 3T/4\}$. The reverse process would then be required to perform upsampling at those timesteps, similar to our cascaded models here. However, there is no guarantee that the reverse transitions at the timesteps in S are conditional Gaussian, unlike the guarantee for reverse transitions at other timesteps for sufficiently slow diffusion. By contrast, our cascaded diffusion model construction dedicates entire conditional diffusion models for these upsampling steps, so it is specified more flexibly.

Conditioning	FID vs train	FID vs validation
LSUN Bedroom		
$s = 0$	2.30	40.68
$s = 251$	**2.06**	40.47
$s = 501$	2.08	**40.44**
$s = 751$	2.14	40.45
$s = 1001$	2.18	40.53
$s = 1251$	2.24	40.58
$s = 1501$	2.28	40.58
LSUN Church		
$s = 0$	3.29	42.21
$s = 251$	2.97	**42.14**
$s = 501$	2.93	42.17
$s = 751$	2.89	42.20
$s = 1001$	2.86	42.26
$s = 1251$	**2.83**	42.28
$s = 1501$	2.84	42.31

Table 5: 128×128 LSUN sample quality: non-truncated conditioning augmentation for a 64×64→128×128 cascading pipeline using the base model for low resolution conditioning.

Recent interest in diffusion models (Sohl-Dickstein et al., 2015) started with work connecting diffusion models to denoising score matching over multiple noise scales (Ho et al., 2020; Song and Ermon, 2019). There have been a number of improvements and alternatives proposed to the diffusion framework, for example generalization to continuous time (Song et al., 2021b), deterministic sampling (Song et al., 2021a), adversarial training (Jolicoeur-Martineau et al., 2021), and others (Gao et al., 2021). For simplicity, we base our models on DDPM (Ho et al., 2020) with modifications from Improved DDPM (Nichol and Dhariwal, 2021) to stay close to the original diffusion framework.

Cascading pipelines have been investigated in work on VQ-VAEs (van den Oord et al., 2016c; Razavi et al., 2019) and autoregressive models (Menick and Kalchbrenner, 2019). Cascading pipelines have also been investigated for diffusion models, such as SR3 (Saharia et al., 2021), Improved DDPM (Nichol and Dhariwal, 2021), and concurrently in ADM (Dhariwal and Nichol, 2021). Our work here focuses on improving cascaded diffusion models for ImageNet generation and is distinguished by the extensive study on conditioning augmentation and deeper cascading pipelines. Our conditioning augmentation work also resembles scheduled sampling in autoregressive sequence generation (Bengio et al., 2015), where noise is used to alleviate the mismatch between train and inference conditions.

Concurrent work (Dhariwal and Nichol, 2021) showed that diffusion models are capable of generating high quality ImageNet samples using an improved architecture, named ADM, and a classifier guidance technique in which a class-conditional diffusion model sampler is modified to simultaneously take gradient steps to maximize the score of an extra trained image

classifier. By contrast, our work focuses solely on improving sample quality by cascading, so we avoid introducing extra model elements such as the image classifier. We are interested in avoiding classifier guidance because the FID and Inception score sample quality metrics that we use to evaluate our models are themselves computed on activations of an image classifier trained on ImageNet, and therefore classifier guidance runs the risk of cheating these metrics.

Avoiding classifier guidance comes at the expense of using thousands of diffusion timesteps in our low resolution models, where ADM uses hundreds. ADM with classifier guidance outperforms our models in terms of FID and Inception scores, while our models outperform ADM without classifier guidance as reported by Dhariwal and Nichol. Our work is a showcase of the effectiveness of cascading alone in a pure generative model, and since classifier guidance and cascading complement each other as techniques to improve sample quality and can be applied together, we expect classifier guidance would improve our results too.

6. Conclusion

We have shown that cascaded diffusion models are capable of outperforming state-of-the-art generative models on the ImageNet class-conditional generation benchmark when paired with conditioning augmentation, our technique of introducing data augmentation into the conditioning information of super-resolution models. Our models outperform BigGAN-deep and VQ-VAE-2 as measured by FID score and classification accuracy score. We found that conditioning augmentation helps sample quality because it combats compounding error in cascading pipelines due to train-test mismatch in super-resolution models, and we proposed practical methods to train and test models amortized over varying levels of conditioning augmentation.

Although there could be negative impact of our work in the form of malicious uses of image generation, our work has the potential to improve beneficial downstream applications such as data compression while advancing the state of knowledge in fundamental machine learning problems. We see our results as a conceptual study of the image synthesis capabilities of diffusion models in their original form with minimal extra techniques, and we hope our work serves as inspiration for future advances in the capabilities of diffusion models.

Acknowledgments

We thank Jascha Sohl-Dickstein, Douglas Eck and the Google Brain team for feedback, research discussions and technical assistance.

Appendix A. Samples

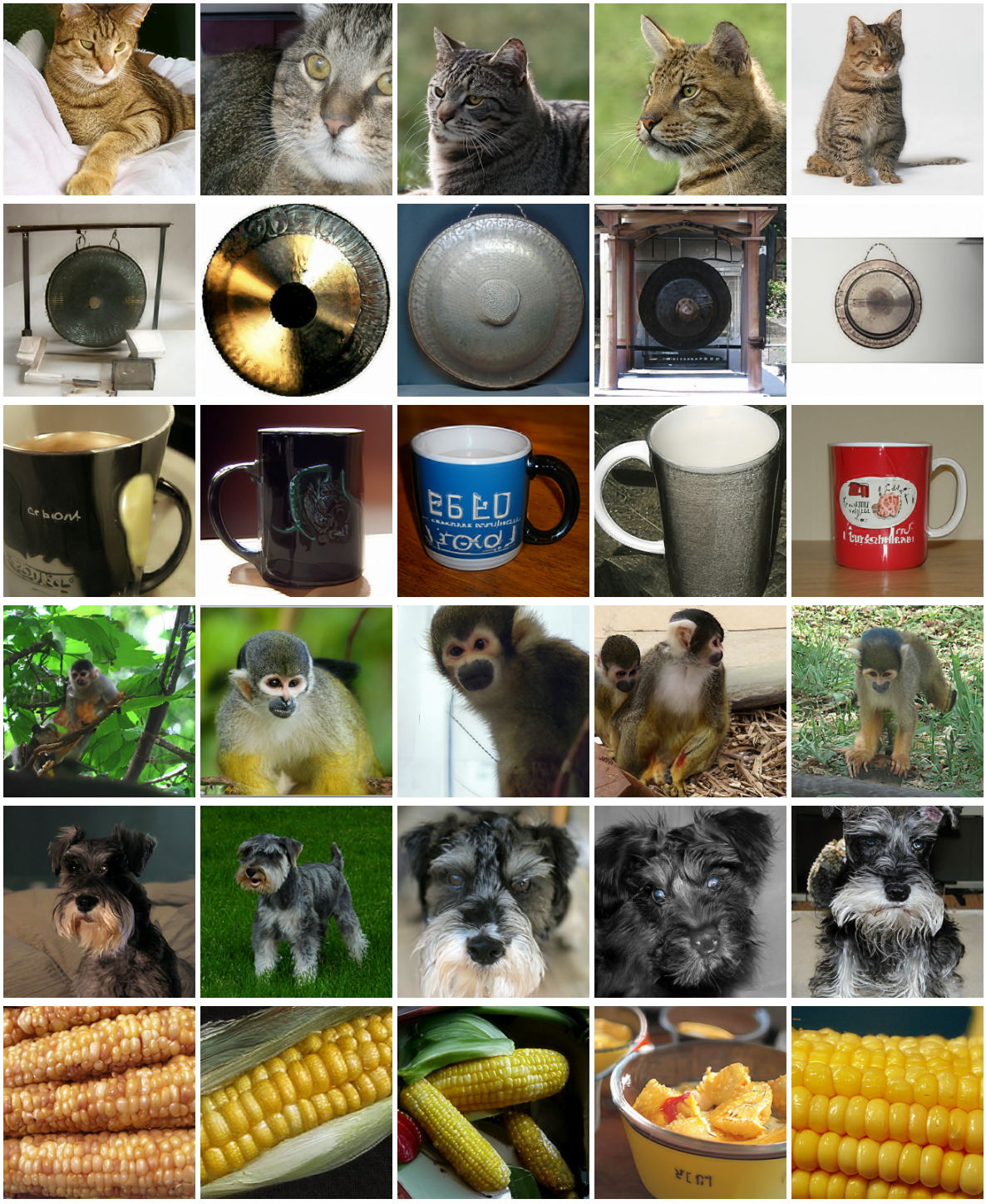


Figure 11: Samples from classes with best relative classification accuracy score. Each row represents a specific ImageNet class. Classes from top to bottom - Tiger Cat (282), Gong (577), Coffee Mug (504), Squirrel Monkey (382), Miniature Schnauzer (196) and Corn (987).

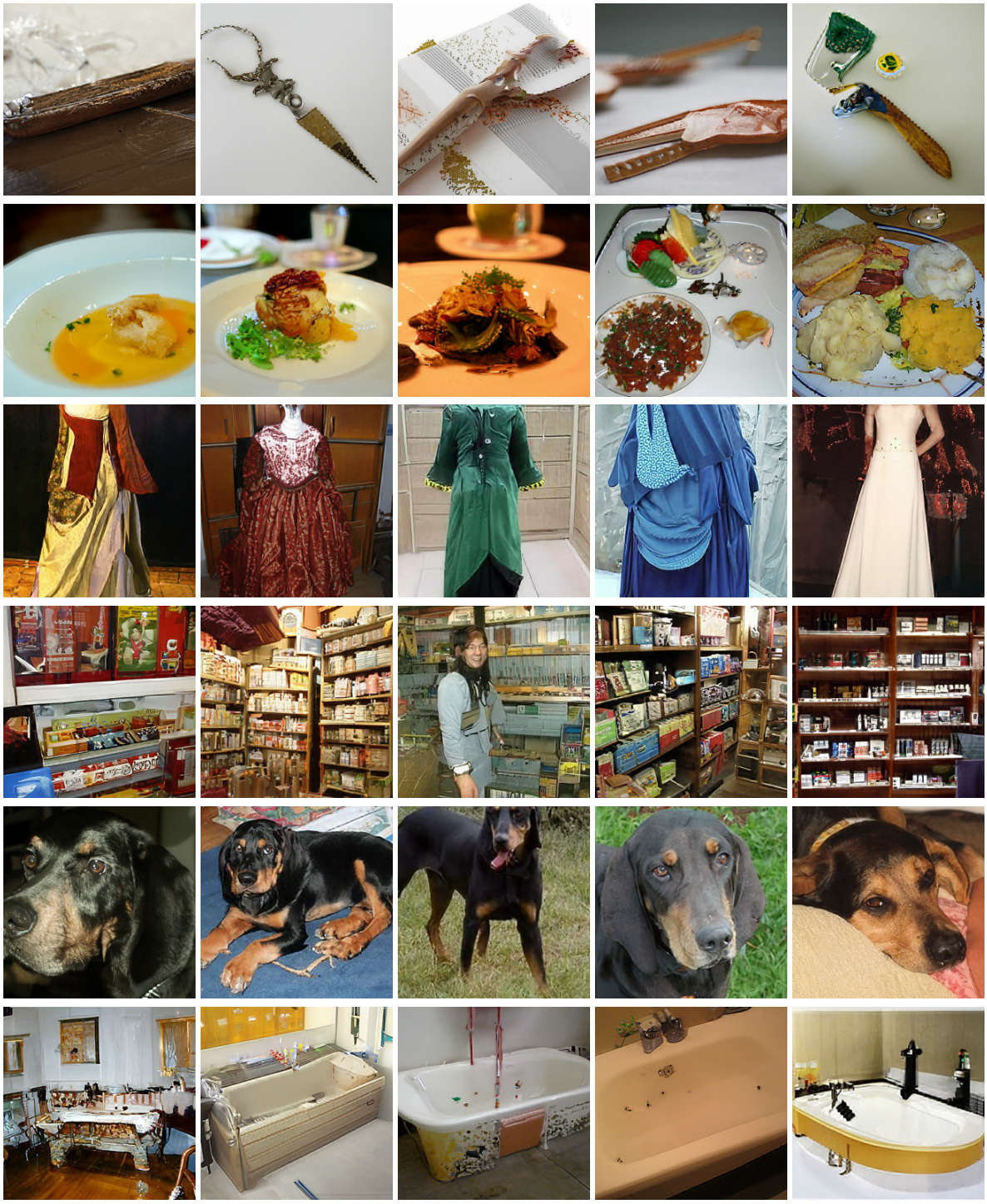


Figure 12: Samples from classes with worst relative classification accuracy score. Each row represents a specific ImageNet class. Classes from top to bottom - Letter Opener (623), Plate (923), Overskirt (689), Tobacco Shop (860), Black-and-tan Coonhound (165) and Bathtub (435).

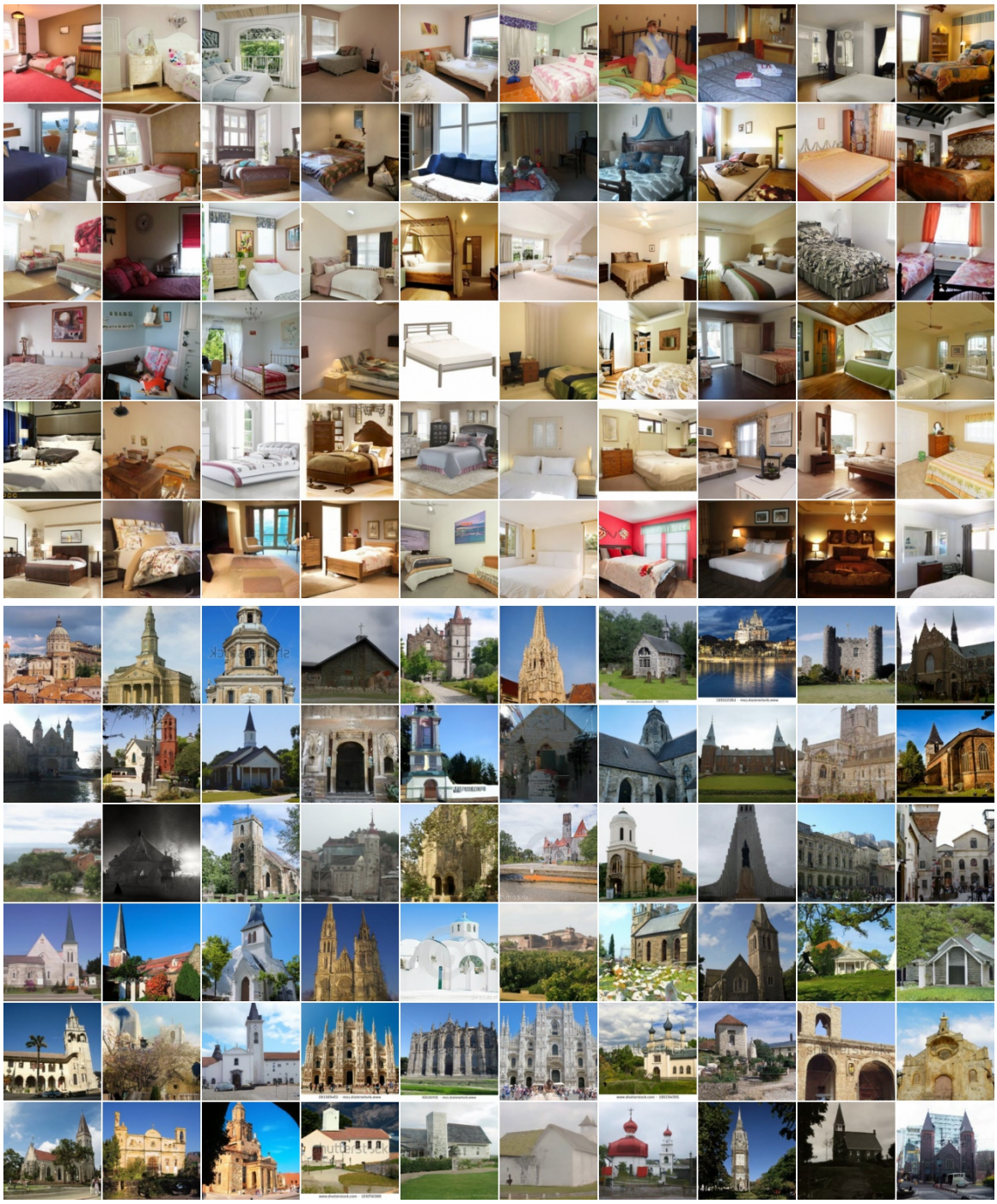


Figure 13: Samples from LSUN 128x128: bedroom subset (first six rows) and church subset (last six rows).

Appendix B. Hyperparameters

B.1 ImageNet

Here we give the hyperparameters of the models in our ImageNet cascading pipelines. Each model in the pipeline is described by its diffusion process, its neural network architecture, and its training hyperparameters. Architecture hyperparameters, such as the base channel count and the list of channel multipliers per resolution, refer to hyperparameters of the U-Net in DDPM and related models (Ho et al., 2020; Nichol and Dhariwal, 2021; Saharia et al., 2021; Salimans et al., 2017). The cosine noise schedule and the hybrid loss method of learning reverse process variances are from Improved DDPM (Nichol and Dhariwal, 2021). Some models are conditioned on $\bar{\alpha}_t$ for post-training sampler tuning (Chen et al., 2021; Saharia et al., 2021).

32×32 base model

- Architecture
  - Base channels: 256
  - Channel multipliers: 1, 2, 3, 4
  - Residual blocks per resolution: 6
  - Attention resolutions: 8, 16
  - Attention heads: 4
- Training
  - Optimizer: Adam
  - Batch size: 2048
  - Learning rate: 1e-4
  - Steps: 700000
  - Dropout: 0.1
  - EMA: 0.9999
  - Hardware: 256 TPU-v3 cores
- Diffusion
  - Timesteps: 4000
  - Noise schedule: cosine
  - Learned variances: yes
  - Loss: hybrid

32×32→64×64 super-resolution

- Architecture
  - Base channels: 256
  - Channel multipliers: 1, 2, 3, 4
  - Residual blocks per resolution: 5
  - Attention resolutions: 8, 16
  - Attention heads: 4
- Training
  - Optimizer: Adam
  - Batch size: 2048
  - Learning rate: 1e-4
  - Steps: 400000
  - Dropout: 0.1
  - EMA: 0.9999
  - Hardware: 256 TPU-v3 cores
- Diffusion
  - Timesteps: 4000
  - Noise schedule: cosine
  - Learned variances: yes
  - Loss: hybrid

64×64→128×128 super-resolution

- Architecture
  - Base channels: 128
  - Channel multipliers: 1, 2, 4, 8, 8
  - Residual blocks per resolution: 3
  - Attention resolutions: 16
  - Attention heads: 1
- Training
  - Optimizer: Adam
  - Batch size: 1024
  - Learning rate: 1e-4
  - Steps: 500000
  - Dropout: 0.0
  - EMA: 0.9999
  - Hardware: 128 TPU-v3 cores
- Diffusion (Training)
  - Timesteps: 2000
  - Noise schedule: linear
  - Learned variances: no
  - Loss: simple
  - Continuous noise conditioning
- Diffusion (Inference)
  - Timesteps: 100
  - Noise schedule: linear

64×64→256×256 super-resolution

- Architecture
  - Base channels: 128
  - Channel multipliers: 1, 2, 4, 4, 8, 8
  - Residual blocks per resolution: 3
  - Attention resolutions: 16
  - Attention heads: 1
- Training
  - Optimizer: Adam
  - Batch size: 1024
  - Learning rate: 1e-4
  - Steps: 500000
  - Dropout: 0.0
  - EMA: 0.9999
  - Hardware: 128 TPU-v3 cores
- Diffusion (Training)
  - Timesteps: 2000
  - Noise schedule: linear
  - Learned variances: no
  - Loss: simple
  - Continuous noise conditioning
- Diffusion (Inference)
  - Timesteps: 100
  - Noise schedule: linear

B.2 LSUN

Here we give the hyperparameters of our LSUN Bedroom and Church pipelines. We used the same hyperparameters for both datasets.

64×64 base model

- Architecture
  - Base channels: 128
  - Channel multipliers: 1, 2, 3, 4
  - Residual blocks per resolution: 3
  - Attention resolutions: 8, 16, 32
  - Attention heads dimension: 64
- Training
  - Optimizer: Adam
  - Batch size: 2048
  - Learning rate: 3e-4
  - Steps: 100000
  - Dropout: 0.1
  - EMA: 0.9999
  - Hardware: 64 TPU-v3 cores
- Diffusion (Training)
  - Noise schedule: cosine
  - Learned variances: no
  - Loss: simple
  - Continuous noise conditioning
- Diffusion (Inference)
  - Timesteps: 256
  - Noise schedule: cosine

64×64→128×128 super-resolution

- Architecture
  - Base channels: 64
  - Channel multipliers: 1, 2, 4, 6, 8
  - Residual blocks per resolution: 3
  - Attention resolutions: 8, 16, 32
  - Attention heads dimension: 64
- Training
  - Optimizer: Adam
  - Batch size: 1024
  - Learning rate: 2e-4
  - Steps: 220000
  - Dropout: 0.1
  - EMA: 0.9999
  - Hardware: 64 TPU-v3 cores
- Diffusion (Training)
  - Noise schedule: cosine
  - Learned variances: no
  - Loss: simple
  - Continuous noise conditioning
- Diffusion (Inference)
  - Timesteps: 256
  - Noise schedule: cosine

References

S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in Neural Information Processing Systems*, 2015.

A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan. WaveGrad: Estimating gradients for waveform generation. *International Conference on Learning Representations*, 2021.

J. De Fauw, S. Dieleman, and K. Simonyan. Hierarchical autoregressive image models with auxiliary decoders. *arXiv preprint arXiv:1903.04933*, 2019.

P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *arXiv preprint arXiv:2105.05233*, 2021.

L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using Real NVP. *International Conference on Learning*, 2017.

R. Gao, Y. Song, B. Poole, Y. N. Wu, and D. P. Kingma. Learning energy-based models by diffusion recovery likelihood. *International Conference on Learning Representations*, 2021.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

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. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

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*, 2019.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, pages 6840–6851, 2020.

A. Jolicoeur-Martineau, R. Piché-Taillefer, R. T. d. Combes, and I. Mitliagkas. Adversarial score matching and improved sampling for image generation. *International Conference on Learning Representations*, 2021.

D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *International Conference on Learning Representations*, 2014.

Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro. DiffWave: A Versatile Diffusion Model for Audio Synthesis. *International Conference on Learning Representations*, 2021.

J. Menick and N. Kalchbrenner. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. In *International Conference on Learning Representations*, 2019.

A. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. *International Conference on Machine Learning*, 2021.

M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. *International Conference on Learning Representations*, 2016.

S. Ravuri and O. Vinyals. Classification Accuracy Score for Conditional Generative Models. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

A. Razavi, A. van den Oord, and O. Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In *Advances in Neural Information Processing Systems*, pages 14837–14847, 2019.

O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi. Image super-resolution via iterative refinement. *arXiv preprint arXiv:2104.07636*, 2021.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications. In *International Conference on Learning Representations*, 2017.

J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265, 2015.

J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. *International Conference on Learning Representations*, 2021a.

Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pages 11895–11907, 2019.

Y. Song and S. Ermon. Improved techniques for training score-based generative. *Advances in Neural Information Processing Systems*, 2020.

Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations*, 2021b.

A. van den 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. *arXiv preprint arXiv:1609.03499*, 2016a.

A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *International Conference on Machine Learning*, 2016b.

A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional image generation with PixelCNN decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016c.

A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. *Advances in Neural Information Processing Systems*, 2017.

Y. Wu, J. Donahue, D. Balduzzi, K. Simonyan, and T. Lillicrap. Logan: Latent optimisation for generative adversarial networks. *arXiv preprint arXiv:1912.00953*, 2019.

F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.