

Diffusion Models Beat GANs on Image Synthesis

Prafulla Dhariwal*

OpenAI

prafulla@openai.com

Alex Nichol*

OpenAI

alex@openai.com

Abstract

We show that diffusion models can achieve image sample quality superior to the current state-of-the-art generative models. We achieve this on **unconditional image synthesis** by finding a better architecture through a series of ablations. For conditional image synthesis, we further improve sample quality with classifier guidance: a simple, compute-efficient method for trading off diversity for fidelity using gradients from a classifier. We achieve an FID of 2.97 on ImageNet 128×128, 4.59 on ImageNet 256×256, and 7.72 on ImageNet 512×512, and we match BigGAN-deep even with as few as 25 forward passes per sample, all while maintaining better coverage of the distribution. Finally, we find that classifier guidance combines well with upsampling diffusion models, further improving FID to 3.94 on ImageNet 256×256 and 3.85 on ImageNet 512×512. We release our code at <https://github.com/openai/guided-diffusion>.

1 Introduction



Figure 1: Selected samples from our best ImageNet 512×512 model (FID 3.85)

Over the past few years, generative models have gained the ability to generate human-like natural language [6], infinite high-quality synthetic images [5, 28, 51] and highly diverse human speech and music [64, 13]. These models can be used in a variety of ways, such as generating images from text prompts [72, 50] or learning useful feature representations [14, 7]. While these models are already

*Equal contribution

capable of producing realistic images and sound, there is still much room for improvement beyond the current state-of-the-art, and better generative models could have wide-ranging impacts on graphic design, games, music production, and countless other fields.

GANs [19] currently hold the state-of-the-art on most image generation tasks [5, 68, 28] as measured by sample quality metrics such as FID [23], Inception Score [54] and Precision [32]. However, some of these metrics do not fully capture diversity, and it has been shown that GANs capture less diversity than state-of-the-art likelihood-based models [51, 43, 42]. Furthermore, GANs are often difficult to train, collapsing without carefully selected hyperparameters and regularizers [5, 41, 4].

While GANs hold the state-of-the-art, their drawbacks make them difficult to scale and apply to new domains. As a result, much work has been done to achieve GAN-like sample quality with likelihood-based models [51, 25, 42, 9]. While these models capture more diversity and are typically easier to scale and train than GANs, they still fall short in terms of visual sample quality. Furthermore, except for VAEs, sampling from these models is slower than GANs in terms of wall-clock time.

Diffusion models are a class of likelihood-based models which have recently been shown to produce high-quality images [56, 59, 25] while offering desirable properties such as distribution coverage, a stationary training objective, and easy scalability. These models generate samples by gradually removing noise from a signal, and their training objective can be expressed as a reweighted variational lower-bound [25]. This class of models already holds the state-of-the-art [60] on CIFAR-10 [31], but still lags behind GANs on difficult generation datasets like LSUN and ImageNet. Nichol and Dhariwal [43] found that these models improve reliably with increased compute, and can produce high-quality samples even on the difficult ImageNet 256×256 dataset using an upsampling stack. However, the FID of this model is still not competitive with BigGAN-deep [5], the current state-of-the-art on this dataset.

We hypothesize that the gap between diffusion models and GANs stems from at least two factors: first, that the model architectures used by recent GAN literature have been heavily explored and refined; second, that GANs are able to trade off diversity for fidelity, producing high quality samples but not covering the whole distribution. We aim to bring these benefits to diffusion models, first by improving model architecture and then by devising a scheme for trading off diversity for fidelity. With these improvements, we achieve a new state-of-the-art, surpassing GANs on several different metrics and datasets.

The rest of the paper is organized as follows. In Section 2, we give a brief background of diffusion models based on Ho et al. [25] and the improvements from Nichol and Dhariwal [43] and Song et al. [57], and we describe our evaluation setup. In Section 3, we introduce simple architecture improvements that give a substantial boost to FID. In Section 4, we describe a method for using gradients from a classifier to guide a diffusion model during sampling. We find that a single hyperparameter, the scale of the classifier gradients, can be tuned to trade off diversity for fidelity, and we can increase this gradient scale factor by an order of magnitude without obtaining adversarial examples [61]. Finally, in Section 5 we show that models with our improved architecture achieve state-of-the-art on unconditional image synthesis tasks, and with classifier guidance achieve state-of-the-art on conditional image synthesis. When using classifier guidance, we find that we can sample with as few as 25 forward passes while maintaining FIDs comparable to BigGAN. We also compare our improved models to upsampling stacks, finding that the two approaches give complementary improvements and that combining them gives the best results on ImageNet 256×256 and 512×512 .

2 Background

In this section, we provide a brief overview of diffusion models. For a more detailed mathematical description, we refer the reader to Appendix B.

On a high level, diffusion models sample from a distribution by reversing a gradual noising process. In particular, sampling starts with noise x_T and produces gradually less-noisy samples x_{T-1}, x_{T-2}, \dots until reaching a final sample x_0 . Each timestep t corresponds to a certain noise level, and x_t can be thought of as a mixture of a signal x_0 with some noise ϵ where the signal to noise ratio is determined by the timestep t . For the remainder of this paper, we assume that the noise ϵ is drawn from a diagonal Gaussian distribution, which works well for natural images and simplifies various derivations.

A diffusion model learns to produce a slightly more “denoised” x_{t-1} from x_t . Ho et al. [25] parameterize this model as a function $\epsilon_\theta(x_t, t)$ which predicts the noise component of a noisy sample x_t . To train these models, each sample in a minibatch is produced by randomly drawing a data sample x_0 , a timestep t , and noise ϵ , which together give rise to a noised sample x_t (Equation 17). The training objective is then $\|\epsilon_\theta(x_t, t) - \epsilon\|^2$, i.e. a simple mean-squared error loss between the true noise and the predicted noise (Equation 26).

It is not immediately obvious how to sample from a noise predictor $\epsilon_\theta(x_t, t)$. Recall that diffusion sampling proceeds by repeatedly predicting x_{t-1} from x_t , starting from x_T . Ho et al. [25] show that, under reasonable assumptions, we can model the distribution $p_\theta(x_{t-1}|x_t)$ of x_{t-1} given x_t as a diagonal Gaussian $\mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$, where the mean $\mu_\theta(x_t, t)$ can be calculated as a function of $\epsilon_\theta(x_t, t)$ (Equation 27). The variance $\Sigma_\theta(x_t, t)$ of this Gaussian distribution can be fixed to a known constant [25] or learned with a separate neural network head [43], and both approaches yield high-quality samples when the total number of diffusion steps T is large enough.

Ho et al. [25] observe that the simple mean-squared error objective, L_{simple} , works better in practice than the actual variational lower bound L_{vib} that can be derived from interpreting the denoising diffusion model as a VAE. They also note that training with this objective and using their corresponding sampling procedure is equivalent to the denoising score matching model from Song and Ermon [58], who use Langevin dynamics to sample from a denoising model trained with multiple noise levels to produce high quality image samples. We often use “diffusion models” as shorthand to refer to both classes of models.

2.1 Improvements

Following the breakthrough work of Song and Ermon [58] and Ho et al. [25], several recent papers have proposed improvements to diffusion models. Here we describe a few of these improvements, which we employ for our models.

Nichol and Dhariwal [43] find that fixing the variance $\Sigma_\theta(x_t, t)$ to a constant as done in Ho et al. [25] is sub-optimal for sampling with fewer diffusion steps, and propose to parameterize $\Sigma_\theta(x_t, t)$ as a neural network whose output v is interpolated as:

$$\Sigma_\theta(x_t, t) = \exp(v \log \beta_t + (1 - v) \log \tilde{\beta}_t) \quad (1)$$

Here, β_t and $\tilde{\beta}_t$ (Equation 19) are the variances in Ho et al. [25] corresponding to upper and lower bounds for the reverse process variances. Additionally, Nichol and Dhariwal [43] propose a hybrid objective for training both $\epsilon_\theta(x_t, t)$ and $\Sigma_\theta(x_t, t)$ using the weighted sum $L_{\text{simple}} + \lambda L_{\text{vib}}$. Learning the reverse process variances with their hybrid objective allows sampling with fewer steps without much drop in sample quality. We adopt this objective and parameterization, and use it throughout our experiments.

Song et al. [57] propose DDIM, which formulates an alternative non-Markovian noising process that has the same forward marginals as DDPM, but allows producing different reverse samplers by changing the variance of the reverse noise. By setting this noise to 0, they provide a way to turn any model $\epsilon_\theta(x_t, t)$ into a deterministic mapping from latents to images, and find that this provides an alternative way to sample with fewer steps. We adopt this sampling approach when using fewer than 50 sampling steps, since Nichol and Dhariwal [43] found it to be beneficial in this regime.

2.2 Sample Quality Metrics

For comparing sample quality across models, we perform quantitative evaluations using the following metrics. While these metrics are often used in practice and correspond well with human judgement, they are not a perfect proxy, and finding better metrics for sample quality evaluation is still an open problem.

Inception Score (IS) was proposed by Salimans et al. [54], and it measures how well a model captures the full ImageNet class distribution while still producing individual samples that are convincing examples of a single class. One drawback of this metric is that it does not reward covering the whole distribution or capturing diversity within a class, and models which memorize a small subset of the full dataset will still have high IS [3]. To better capture diversity than IS, Fréchet Inception Distance (FID) was proposed by Heusel et al. [23], who argued that it is more consistent with human

| Channels | Depth | Heads | Attention resolutions | BigGAN up/downsample | Rescale resblock | FID 700K | FID 1200K |
|----------|-------|-------|-----------------------|----------------------|------------------|--------------|--------------|
| 160 | 2 | 1 | 16 | \times | \times | 15.33 | 13.21 |
| 128 | 4 | 4 | 32,16,8 | \checkmark | \checkmark | -0.21 | -0.48 |
| | | | | | | -0.54 | -0.82 |
| | | | | | | -0.72 | -0.66 |
| | | | | | | -1.20 | -1.21 |
| | | | | | | 0.16 | 0.25 |
| | | | | | | -3.14 | -3.00 |

Table 1: Ablation of various architecture changes, evaluated at 700K and 1200K iterations

judgement than Inception Score. FID provides a symmetric measure of the distance between two image distributions in the Inception-V3 [62] latent space. Recently, sFID was proposed by Nash et al. [42] as a version of FID that uses spatial features rather than the standard pooled features. They find that this metric better captures spatial relationships, rewarding image distributions with coherent high-level structure. Finally, Kynkänniemi et al. [32] proposed Improved Precision and Recall metrics to separately measure sample fidelity as the fraction of model samples which fall into the data manifold (precision), and diversity as the fraction of data samples which fall into the sample manifold (recall).

We use FID as our default metric for overall sample quality comparisons as it captures both diversity and fidelity and has been the de facto standard metric for state-of-the-art generative modeling work [27, 28, 5, 25]. We use Precision or IS to measure fidelity, and Recall to measure diversity or distribution coverage. When comparing against other methods, we re-compute these metrics using public samples or models whenever possible. This is for two reasons: first, some papers [27, 28, 25] compare against arbitrary subsets of the training set which are not readily available; and second, subtle implementation differences can affect the resulting FID values [45]. To ensure consistent comparisons, we use the entire training set as the reference batch [23, 5], and evaluate metrics for all models using the same codebase.

3 Architecture Improvements

In this section we conduct several architecture ablations to find the model architecture that provides the best sample quality for diffusion models.

Ho et al. [25] introduced the UNet architecture for diffusion models, which Jolicoeur-Martineau et al. [26] found to substantially improve sample quality over the previous architectures [58, 33] used for denoising score matching. The UNet model uses a stack of residual layers and downsampling convolutions, followed by a stack of residual layers with upsampling convolutions, with skip connections connecting the layers with the same spatial size. In addition, they use a global attention layer at the 16×16 resolution with a single head, and add a projection of the timestep embedding into each residual block. Song et al. [60] found that further changes to the UNet architecture improved performance on the CIFAR-10 [31] and CelebA-64 [34] datasets. We show the same result on ImageNet 128×128 , finding that architecture can indeed give a substantial boost to sample quality on much larger and more diverse datasets at a higher resolution.

We explore the following architectural changes:

- Increasing depth versus width, holding model size relatively constant.
- Increasing the number of attention heads.
- Using attention at 32×32 , 16×16 , and 8×8 resolutions rather than only at 16×16 .
- Using the BigGAN [5] residual block for upsampling and downsampling the activations, following [60].
- Rescaling residual connections with $\frac{1}{\sqrt{2}}$, following [60, 27, 28].

For all comparisons in this section, we train models on ImageNet 128×128 with batch size 256, and sample using 250 sampling steps. We train models with the above architecture changes and compare

| Number of heads | Channels per head | FID |
|-----------------|-------------------|-------|
| 1 | | 14.08 |
| 2 | | -0.50 |
| 4 | | -0.97 |
| 8 | | -1.17 |
| | 32 | -1.36 |
| | 64 | -1.03 |
| | 128 | -1.08 |

Table 2: Ablation of various attention configurations. More heads or lower channels per heads both lead to improved FID.

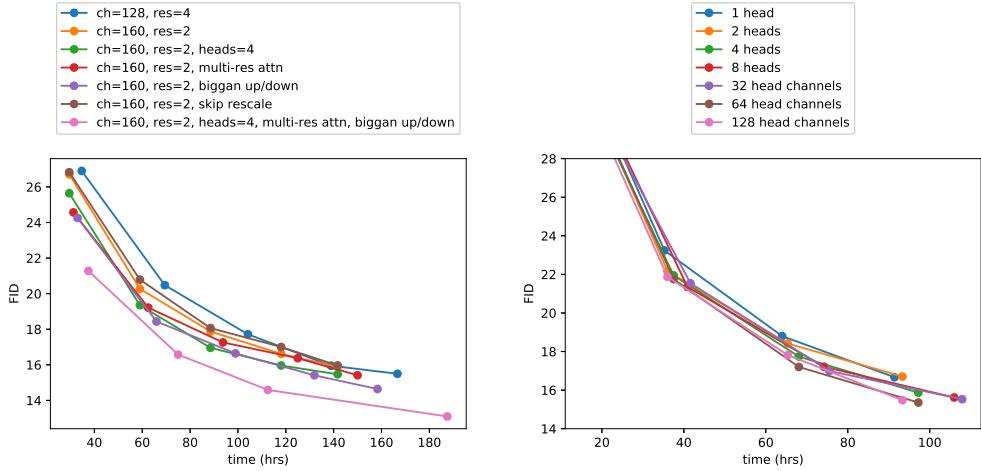


Figure 2: Ablation of various architecture changes, showing FID as a function of wall-clock time. FID evaluated over 10k samples instead of 50k for efficiency.

| Operation | FID |
|----------------------|-------|
| AdaGN | 13.06 |
| Addition + GroupNorm | 15.08 |

Table 3: Ablating the element-wise operation used when projecting timestep and class embeddings into each residual block. Replacing AdaGN with the Addition + GroupNorm layer from Ho et al. [25] makes FID worse.

them on FID, evaluated at two different points of training, in Table 1. Aside from rescaling residual connections, all of the other modifications improve performance and have a positive compounding effect. We observe in Figure 2 that while increased depth helps performance, it increases training time and takes longer to reach the same performance as a wider model, so we opt not to use this change in further experiments.

We also study other attention configurations that better match the Transformer architecture [66]. To this end, we experimented with either fixing attention heads to a constant, or fixing the number of channels per head. For the rest of the architecture, we use 128 base channels, 2 residual blocks per resolution, multi-resolution attention, and BigGAN up/downsampling, and we train the models for 700K iterations. Table 2 shows our results, indicating that more heads or fewer channels per head improves FID. In Figure 2, we see 64 channels is best for wall-clock time, so we opt to use 64 channels per head as our default. We note that this choice also better matches modern transformer architectures, and is on par with our other configurations in terms of final FID.

3.1 Adaptive Group Normalization

We also experiment with a layer [43] that we refer to as adaptive group normalization (AdaGN), which incorporates the timestep and class embedding into each residual block after a group normalization operation [69], similar to adaptive instance norm [27] and FiLM [48]. We define this layer as $\text{AdaGN}(h, y) = y_s \text{GroupNorm}(h) + y_b$, where h is the intermediate activations of the residual block following the first convolution, and $y = [y_s, y_b]$ is obtained from a linear projection of the timestep and class embedding.

We had already seen AdaGN improve our earliest diffusion models, and so had it included by default in all our runs. In Table 3, we explicitly ablate this choice, and find that the adaptive group normalization layer indeed improved FID. Both models use 128 base channels and 2 residual blocks per resolution, multi-resolution attention with 64 channels per head, and BigGAN up/downsampling, and were trained for 700K iterations.

In the rest of the paper, we use this final improved model architecture as our default: variable width with 2 residual blocks per resolution, multiple heads with 64 channels per head, attention at 32, 16 and 8 resolutions, BigGAN residual blocks for up and downampling, and adaptive group normalization for injecting timestep and class embeddings into residual blocks.

4 Classifier Guidance

In addition to employing well designed architectures, GANs for conditional image synthesis [39, 5] make heavy use of class labels. This often takes the form of class-conditional normalization statistics [16, 11] as well as discriminators with heads that are explicitly designed to behave like classifiers $p(y|x)$ [40]. As further evidence that class information is crucial to the success of these models, Lucic et al. [36] find that it is helpful to generate synthetic labels when working in a label-limited regime.

Given this observation for GANs, it makes sense to explore different ways to condition diffusion models on class labels. We already incorporate class information into normalization layers (Section 3.1). Here, we explore a different approach: exploiting a classifier $p(y|x)$ to improve a diffusion generator. Sohl-Dickstein et al. [56] and Song et al. [60] show one way to achieve this, wherein a pre-trained diffusion model can be conditioned using the gradients of a classifier. In particular, we can train a classifier $p_\phi(y|x_t, t)$ on noisy images x_t , and then use gradients $\nabla_{x_t} \log p_\phi(y|x_t, t)$ to guide the diffusion sampling process towards an arbitrary class label y .

In this section, we first review two ways of deriving conditional sampling processes using classifiers. We then describe how we use such classifiers in practice to improve sample quality. We choose the notation $p_\phi(y|x_t, t) = p_\phi(y|x_t)$ and $\epsilon_\theta(x_t, t) = \epsilon_\theta(x_t)$ for brevity, noting that they refer to separate functions for each timestep t and at training time the models must be conditioned on the input t .

4.1 Conditional Reverse Noising Process

We start with a diffusion model with an unconditional reverse noising process $p_\theta(x_t|x_{t+1})$. To condition this on a label y , it suffices to sample each transition² according to

$$p_{\theta,\phi}(x_t|x_{t+1}, y) = Z p_\theta(x_t|x_{t+1}) p_\phi(y|x_t) \quad (2)$$

where Z is a normalizing constant (proof in Appendix H). It is typically intractable to sample from this distribution exactly, but Sohl-Dickstein et al. [56] show that it can be approximated as a perturbed Gaussian distribution. Here, we review this derivation.

Recall that our diffusion model predicts the previous timestep x_t from timestep x_{t+1} using a Gaussian distribution:

$$p_\theta(x_t|x_{t+1}) = \mathcal{N}(\mu, \Sigma) \quad (3)$$

$$\log p_\theta(x_t|x_{t+1}) = -\frac{1}{2}(x_t - \mu)^T \Sigma^{-1} (x_t - \mu) + C \quad (4)$$

²We must also sample x_T conditioned on y , but a noisy enough diffusion process causes x_T to be nearly Gaussian even in the conditional case.

Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $p_\phi(y|x_t)$, and gradient scale s .

```

Input: class label  $y$ , gradient scale  $s$ 
 $x_T \leftarrow$  sample from  $\mathcal{N}(0, \mathbf{I})$ 
for all  $t$  from  $T$  to 1 do
     $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$ 
     $x_{t-1} \leftarrow$  sample from  $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$ 
end for
return  $x_0$ 
```

Algorithm 2 Classifier guided DDIM sampling, given a diffusion model $\epsilon_\theta(x_t)$, classifier $p_\phi(y|x_t)$, and gradient scale s .

```

Input: class label  $y$ , gradient scale  $s$ 
 $x_T \leftarrow$  sample from  $\mathcal{N}(0, \mathbf{I})$ 
for all  $t$  from  $T$  to 1 do
     $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1 - \bar{\alpha}_t} \nabla_{x_t} \log p_\phi(y|x_t)$ 
     $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \left( \frac{x_t - \sqrt{1 - \bar{\alpha}_t} \hat{\epsilon}}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1 - \bar{\alpha}_{t-1}} \hat{\epsilon}$ 
end for
return  $x_0$ 
```

We can assume that $\log p_\phi(y|x_t)$ has low curvature compared to Σ^{-1} . This assumption is reasonable in the limit of infinite diffusion steps, where $\|\Sigma\| \rightarrow 0$. In this case, we can approximate $\log p_\phi(y|x_t)$ using a Taylor expansion around $x_t = \mu$ as

$$\log p_\phi(y|x_t) \approx \log p_\phi(y|x_t)|_{x_t=\mu} + (x_t - \mu) \nabla_{x_t} \log p_\phi(y|x_t)|_{x_t=\mu} \quad (5)$$

$$= (x_t - \mu)g + C_1 \quad (6)$$

Here, $g = \nabla_{x_t} \log p_\phi(y|x_t)|_{x_t=\mu}$, and C_1 is a constant. This gives

$$\log(p_\theta(x_t|x_{t+1})p_\phi(y|x_t)) \approx -\frac{1}{2}(x_t - \mu)^T \Sigma^{-1}(x_t - \mu) + (x_t - \mu)g + C_2 \quad (7)$$

$$= -\frac{1}{2}(x_t - \mu - \Sigma g)^T \Sigma^{-1}(x_t - \mu - \Sigma g) + \frac{1}{2}g^T \Sigma g + C_2 \quad (8)$$

$$= -\frac{1}{2}(x_t - \mu - \Sigma g)^T \Sigma^{-1}(x_t - \mu - \Sigma g) + C_3 \quad (9)$$

$$= \log p(z) + C_4, z \sim \mathcal{N}(\mu + \Sigma g, \Sigma) \quad (10)$$

We can safely ignore the constant term C_4 , since it corresponds to the normalizing coefficient Z in Equation 2. We have thus found that the conditional transition operator can be approximated by a Gaussian similar to the unconditional transition operator, but with its mean shifted by Σg . Algorithm 1 summarizes the corresponding sampling algorithm. We include an optional scale factor s for the gradients, which we describe in more detail in Section 4.3.

4.2 Conditional Sampling for DDIM

The above derivation for conditional sampling is only valid for the stochastic diffusion sampling process, and cannot be applied to deterministic sampling methods like DDIM [57]. To this end, we use a score-based conditioning trick adapted from Song et al. [60], which leverages the connection between diffusion models and score matching [59]. In particular, if we have a model $\epsilon_\theta(x_t)$ that predicts the noise added to a sample, then this can be used to derive a score function:

$$\nabla_{x_t} \log p_\theta(x_t) = -\frac{1}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t) \quad (11)$$



Figure 3: Samples from an unconditional diffusion model with classifier guidance to condition on the class "Pembroke Welsh corgi". Using classifier scale 1.0 (left; FID: 33.0) does not produce convincing samples in this class, whereas classifier scale 10.0 (right; FID: 12.0) produces much more class-consistent images.

We can now substitute this into the score function for $p(x_t)p(y|x_t)$:

$$\nabla_{x_t} \log(p_\theta(x_t)p_\phi(y|x_t)) = \nabla_{x_t} \log p_\theta(x_t) + \nabla_{x_t} \log p_\phi(y|x_t) \quad (12)$$

$$= -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t) + \nabla_{x_t} \log p_\phi(y|x_t) \quad (13)$$

Finally, we can define a new epsilon prediction $\hat{\epsilon}(x_t)$ which corresponds to the score of the joint distribution:

$$\hat{\epsilon}(x_t) := \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t} \nabla_{x_t} \log p_\phi(y|x_t) \quad (14)$$

We can then use the exact same sampling procedure as used for regular DDIM, but with the modified noise predictions $\hat{\epsilon}_\theta(x_t)$ instead of $\epsilon_\theta(x_t)$. Algorithm 2 summarizes the corresponding sampling algorithm.

4.3 Scaling Classifier Gradients

To apply classifier guidance to a large scale generative task, we train classification models on ImageNet. Our classifier architecture is simply the downsampling trunk of the UNet model with an attention pool [49] at the 8x8 layer to produce the final output. We train these classifiers on the same noising distribution as the corresponding diffusion model, and also add random crops to reduce overfitting. After training, we incorporate the classifier into the sampling process of the diffusion model using Equation 10, as outlined by Algorithm 1.

In initial experiments with unconditional ImageNet models, we found it necessary to scale the classifier gradients by a constant factor larger than 1. When using a scale of 1, we observed that the classifier assigned reasonable probabilities (around 50%) to the desired classes for the final samples, but these samples did not match the intended classes upon visual inspection. Scaling up the classifier gradients remedied this problem, and the class probabilities from the classifier increased to nearly 100%. Figure 3 shows an example of this effect.

To understand the effect of scaling classifier gradients, note that $s \cdot \nabla_x \log p(y|x) = \nabla_x \log \frac{1}{Z} p(y|x)^s$, where Z is an arbitrary constant. As a result, the conditioning process is still theoretically grounded in a re-normalized classifier distribution proportional to $p(y|x)^s$. When $s > 1$, this distribution becomes sharper than $p(y|x)$, since larger values are amplified by the exponent. In other words, using a larger gradient scale focuses more on the modes of the classifier, which is potentially desirable for producing higher fidelity (but less diverse) samples.

In the above derivations, we assumed that the underlying diffusion model was unconditional, modeling $p(x)$. It is also possible to train conditional diffusion models, $p(x|y)$, and use classifier guidance in the exact same way. Table 4 shows that the sample quality of both unconditional and conditional models can be greatly improved by classifier guidance. We see that, with a high enough scale, the guided unconditional model can get quite close to the FID of an unguided conditional model, although training directly with the class labels still helps. Guiding a conditional model further improves FID.

Table 4 also shows that classifier guidance improves precision at the cost of recall, thus introducing a trade-off in sample fidelity versus diversity. We explicitly evaluate how this trade-off varies with

| Conditional | Guidance | Scale | FID | sFID | IS | Precision | Recall |
|-------------|----------|-------|--------------|-------------|---------------|-------------|-------------|
| \times | \times | | 26.21 | 6.35 | 39.70 | 0.61 | 0.63 |
| \times | ✓ | 1.0 | 33.03 | 6.99 | 32.92 | 0.56 | 0.65 |
| \times | ✓ | 10.0 | 12.00 | 10.40 | 95.41 | 0.76 | 0.44 |
| ✓ | \times | | 10.94 | 6.02 | 100.98 | 0.69 | 0.63 |
| ✓ | ✓ | 1.0 | 4.59 | 5.25 | 186.70 | 0.82 | 0.52 |
| ✓ | ✓ | 10.0 | 9.11 | 10.93 | 283.92 | 0.88 | 0.32 |

Table 4: Effect of classifier guidance on sample quality. Both conditional and unconditional models were trained for 2M iterations on ImageNet 256×256 with batch size 256.

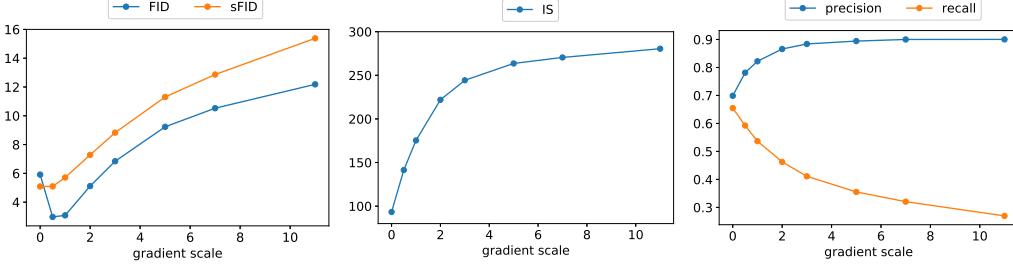


Figure 4: Change in sample quality as we vary scale of the classifier gradients for a class-conditional ImageNet 128×128 model.

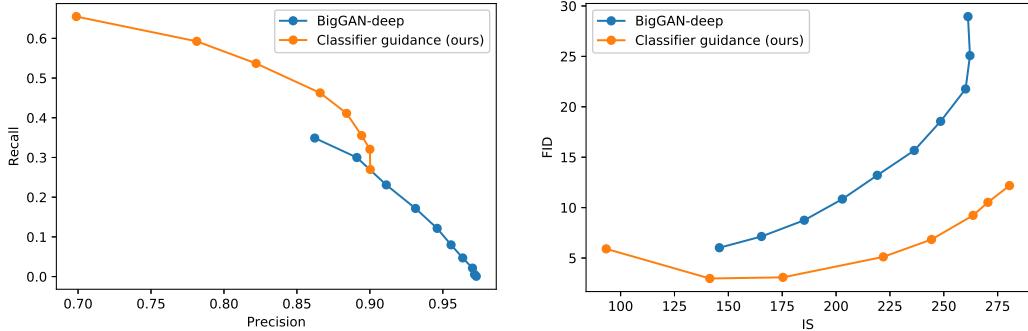


Figure 5: Trade-offs when varying truncation for BigGAN-deep and gradient scale for classifier guidance. Models are evaluated on ImageNet 128×128. The BigGAN-deep results were produced using the TFHub model [12] at truncation levels [0.1, 0.2, 0.3, ..., 1.0].

the gradient scale in Figure 4. We see that scaling the gradients beyond 1.0 smoothly trades off recall (a measure of diversity) for higher precision and IS (measures of fidelity). Since FID and sFID depend on both diversity and fidelity, their best values are obtained at an intermediate point. We also compare our guidance with the truncation trick from BigGAN in Figure 5. We find that classifier guidance is strictly better than BigGAN-deep when trading off FID for Inception Score. Less clear cut is the precision/recall trade-off, which shows that classifier guidance is only a better choice up until a certain precision threshold, after which point it cannot achieve better precision.

5 Results

To evaluate our improved model architecture on unconditional image generation, we train separate diffusion models on three LSUN [71] classes: bedroom, horse, and cat. To evaluate classifier guidance, we train conditional diffusion models on the ImageNet [52] dataset at 128×128, 256×256, and 512×512 resolution.

| Model | FID | sFID | Prec | Rec | |
|---------------------------------|-------------|-------------|-------------|-------------|--|
| LSUN Bedrooms 256×256 | | | | | |
| DCTransformer [†] [42] | 6.40 | 6.66 | 0.44 | 0.56 | |
| DDPM [25] | 4.89 | 9.07 | 0.60 | 0.45 | |
| IDDPM [43] | 4.24 | 8.21 | 0.62 | 0.46 | |
| StyleGAN [27] | 2.35 | 6.62 | 0.59 | 0.48 | |
| ADM (dropout) | 1.90 | 5.59 | 0.66 | 0.51 | |
| LSUN Horses 256×256 | | | | | |
| StyleGAN2 [28] | 3.84 | 6.46 | 0.63 | 0.48 | |
| ADM | 2.95 | 5.94 | 0.69 | 0.55 | |
| ADM (dropout) | 2.57 | 6.81 | 0.71 | 0.55 | |
| LSUN Cats 256×256 | | | | | |
| DDPM [25] | 17.1 | 12.4 | 0.53 | 0.48 | |
| StyleGAN2 [28] | 7.25 | 6.33 | 0.58 | 0.43 | |
| ADM (dropout) | 5.57 | 6.69 | 0.63 | 0.52 | |
| ImageNet 64×64 | | | | | |
| BigGAN-deep* [5] | 4.06 | 3.96 | 0.79 | 0.48 | |
| IDDPM [43] | 2.92 | 3.79 | 0.74 | 0.62 | |
| ADM | 2.61 | 3.77 | 0.73 | 0.63 | |
| ADM (dropout) | 2.07 | 4.29 | 0.74 | 0.63 | |
| ImageNet 128×128 | | | | | |
| BigGAN-deep [5] | 6.02 | 7.18 | 0.86 | 0.35 | |
| LOGAN [†] [68] | 3.36 | | | | |
| ADM | 5.91 | 5.09 | 0.70 | 0.65 | |
| ADM-G (25 steps) | 5.98 | 7.04 | 0.78 | 0.51 | |
| ADM-G | 2.97 | 5.09 | 0.78 | 0.59 | |
| ImageNet 256×256 | | | | | |
| DCTransformer [†] [42] | 36.51 | 8.24 | 0.36 | 0.67 | |
| VQ-VAE-2 ^{†‡} [51] | 31.11 | 17.38 | 0.36 | 0.57 | |
| IDDPM [‡] [43] | 12.26 | 5.42 | 0.70 | 0.62 | |
| SR3 ^{†‡} [53] | 11.30 | | | | |
| BigGAN-deep [5] | 6.95 | 7.36 | 0.87 | 0.28 | |
| ADM | 10.94 | 6.02 | 0.69 | 0.63 | |
| ADM-G (25 steps) | 5.44 | 5.32 | 0.81 | 0.49 | |
| ADM-G | 4.59 | 5.25 | 0.82 | 0.52 | |
| ImageNet 512×512 | | | | | |
| BigGAN-deep [5] | 8.43 | 8.13 | 0.88 | 0.29 | |
| ADM | 23.24 | 10.19 | 0.73 | 0.60 | |
| ADM-G (25 steps) | 8.41 | 9.67 | 0.83 | 0.47 | |
| ADM-G | 7.72 | 6.57 | 0.87 | 0.42 | |

Table 5: Sample quality comparison with state-of-the-art generative models for each task. ADM refers to our **ablated diffusion model**, and ADM-G additionally uses classifier guidance. LSUN diffusion models are sampled using 1000 steps (see Appendix J). ImageNet diffusion models are sampled using 250 steps, except when we use the DDIM sampler with 25 steps. *No BigGAN-deep model was available at this resolution, so we trained our own. [†]Values are taken from a previous paper, due to lack of public models or samples. [‡]Results use two-resolution stacks.

5.1 State-of-the-art Image Synthesis

Table 5 summarizes our results. Our diffusion models can obtain the best FID on each task, and the best sFID on all but one task. With the improved architecture, we already obtain state-of-the-art image generation on LSUN and ImageNet 64×64. For higher resolution ImageNet, we observe that classifier guidance allows our models to substantially outperform the best GANs. These models obtain perceptual quality similar to GANs, while maintaining a higher coverage of the distribution as measured by recall, and can even do so using only 25 diffusion steps.

Figure 6 compares random samples from the best BigGAN-deep model to our best diffusion model. While the samples are of similar perceptual quality, the diffusion model contains more modes than the GAN, such as zoomed ostrich heads, single flamingos, different orientations of cheeseburgers, and a tinca fish with no human holding it. We also check our generated samples for nearest neighbors in the Inception-V3 feature space in Appendix C, and we show additional samples in Appendices K-M.

5.2 Comparison to Upsampling

We also compare guidance to using a two-stage upsampling stack. Nichol and Dhariwal [43] and Saharia et al. [53] train two-stage diffusion models by combining a low-resolution diffusion model with a corresponding upsampling diffusion model. In this approach, the upsampling model is trained to upsample images from the training set, and conditions on low-resolution images that are concatenated channel-wise to the model input using a simple interpolation (e.g. bilinear). During sampling, the low-resolution model produces a sample, and then the upsampling model is conditioned on this sample. This greatly improves FID on ImageNet 256×256, but does not reach the same performance as state-of-the-art models like BigGAN-deep [43, 53], as seen in Table 5.

In Table 6, we show that guidance and upsampling improve sample quality along different axes. While upsampling improves precision while keeping a high recall, guidance provides a knob to trade



Figure 6: Samples from BigGAN-deep with truncation 1.0 (FID 6.95, left) vs samples from our diffusion model with guidance (FID 4.59, middle) and samples from the training set (right).

| Model | S_{base} | $S_{upsample}$ | FID | sFID | IS | Precision | Recall |
|-------------------------|------------|----------------|-------------|-------------|---------------|-------------|-------------|
| ImageNet 256×256 | | | | | | | |
| ADM | 250 | | 10.94 | 6.02 | 100.98 | 0.69 | 0.63 |
| ADM-U | 250 | 250 | 7.49 | 5.13 | 127.49 | 0.72 | 0.63 |
| ADM-G | 250 | | 4.59 | 5.25 | 186.70 | 0.82 | 0.52 |
| ADM-G, ADM-U | 250 | 250 | 3.94 | 6.14 | 215.84 | 0.83 | 0.53 |
| ImageNet 512×512 | | | | | | | |
| ADM | 250 | | 23.24 | 10.19 | 58.06 | 0.73 | 0.60 |
| ADM-U | 250 | 250 | 9.96 | 5.62 | 121.78 | 0.75 | 0.64 |
| ADM-G | 250 | | 7.72 | 6.57 | 172.71 | 0.87 | 0.42 |
| ADM-G, ADM-U | 25 | 25 | 5.96 | 12.10 | 187.87 | 0.81 | 0.54 |
| ADM-G, ADM-U | 250 | 25 | 4.11 | 9.57 | 219.29 | 0.83 | 0.55 |
| ADM-G, ADM-U | 250 | 250 | 3.85 | 5.86 | 221.72 | 0.84 | 0.53 |

Table 6: Comparing our single, upsampling and classifier guided models. For upsampling, we use the upsampling stack from Nichol and Dhariwal [43] combined with our architecture improvements, which we refer to as ADM-U. The base resolution for the two-stage upsampling models is 64 and 128 for the 256 and 512 models, respectively. When combining classifier guidance with upsampling, we only guide the lower resolution model.

off diversity for much higher precision. We achieve the best FIDs by using guidance at a lower resolution before upsampling to a higher resolution, indicating that these approaches complement one another.

6 Related Work

Score based generative models were introduced by Song and Ermon [59] as a way of modeling a data distribution using its gradients, and then sampling using Langevin dynamics [67]. Ho et al. [25] found a connection between this method and diffusion models [56], and achieved excellent sample quality by leveraging this connection. After this breakthrough work, many works followed up with more promising results: Kong et al. [30] and Chen et al. [8] demonstrated that diffusion models

work well for audio; Jolicoeur-Martineau et al. [26] found that a GAN-like setup could improve samples from these models; Song et al. [60] explored ways to leverage techniques from stochastic differential equations to improve the sample quality obtained by score-based models; Song et al. [57] and Nichol and Dhariwal [43] proposed methods to improve sampling speed; Nichol and Dhariwal [43] and Saharia et al. [53] demonstrated promising results on the difficult ImageNet generation task using upsampling diffusion models. Also related to diffusion models, and following the work of Sohl-Dickstein et al. [56], Goyal et al. [21] described a technique for learning a model with learned iterative generation steps, and found that it could achieve good image samples when trained with a likelihood objective.

One missing element from previous work on diffusion models is a way to trade off diversity for fidelity. Other generative techniques provide natural levers for this trade-off. Brock et al. [5] introduced the truncation trick for GANs, wherein the latent vector is sampled from a truncated normal distribution. They found that increasing truncation naturally led to a decrease in diversity but an increase in fidelity. More recently, Razavi et al. [51] proposed to use classifier rejection sampling to filter out bad samples from an autoregressive likelihood-based model, and found that this technique improved FID. Most likelihood-based models also allow for low-temperature sampling [1], which provides a natural way to emphasize modes of the data distribution (see Appendix G).

Other likelihood-based models have been shown to produce high-fidelity image samples. VQ-VAE [65] and VQ-VAE-2 [51] are autoregressive models trained on top of quantized latent codes, greatly reducing the computational resources required to train these models on large images. These models produce diverse and high quality images, but still fall short of GANs without expensive rejection sampling and special metrics to compensate for blurriness. DCTransformer [42] is a related method which relies on a more intelligent compression scheme. VAEs are another promising class of likelihood-based models, and recent methods such as NVAE [63] and VDVAE [9] have successfully been applied to difficult image generation domains. Energy-based models are another class of likelihood-based models with a rich history [1, 10, 24]. Sampling from the EBM distribution is challenging, and Xie et al. [70] demonstrate that Langevin dynamics can be used to sample coherent images from these models. Du and Mordatch [15] further improve upon this approach, obtaining high quality images. More recently, Gao et al. [18] incorporate diffusion steps into an energy-based model, and find that doing so improves image samples from these models.

Other works have controlled generative models with a pre-trained classifier. For example, an emerging body of work [17, 47, 2] aims to optimize GAN latent spaces for text prompts using pre-trained CLIP [49] models. More similar to our work, Song et al. [60] uses a classifier to generate class-conditional CIFAR-10 images with a diffusion model. In some cases, classifiers can act as stand-alone generative models. For example, Santurkar et al. [55] demonstrate that a robust image classifier can be used as a stand-alone generative model, and Grathwohl et al. [22] train a model which is jointly a classifier and an energy-based model.

7 Limitations and Future Work

While we believe diffusion models are an extremely promising direction for generative modeling, they are still slower than GANs at sampling time due to the use of multiple denoising steps (and therefore forward passes). One promising work in this direction is from Luhman and Luhman [37], who explore a way to distill the DDIM sampling process into a single step model. The samples from the single step model are not yet competitive with GANs, but are much better than previous single-step likelihood-based models. Future work in this direction might be able to completely close the sampling speed gap between diffusion models and GANs without sacrificing image quality.

Our proposed classifier guidance technique is currently limited to labeled datasets, and we have provided no effective strategy for trading off diversity for fidelity on unlabeled datasets. In the future, our method could be extended to unlabeled data by clustering samples to produce synthetic labels [36] or by training discriminative models to predict when samples are in the true data distribution or from the sampling distribution.

The effectiveness of classifier guidance demonstrates that we can obtain powerful generative models from the gradients of a classification function. This could be used to condition pre-trained models in a plethora of ways, for example by conditioning an image generator with a text caption using a noisy version of CLIP [49], similar to recent methods that guide GANs using text prompts [17, 47],

2]. It also suggests that large unlabeled datasets could be leveraged in the future to pre-train powerful diffusion models that can later be improved by using a classifier with desirable properties.

8 Conclusion

We have shown that diffusion models, a class of likelihood-based models with a stationary training objective, can obtain better sample quality than state-of-the-art GANs. Our improved architecture is sufficient to achieve this on unconditional image generation tasks, and our classifier guidance technique allows us to do so on class-conditional tasks. In the latter case, we find that the scale of the classifier gradients can be adjusted to trade off diversity for fidelity. These guided diffusion models can reduce the sampling time gap between GANs and diffusion models, although diffusion models still require multiple forward passes during sampling. Finally, by combining guidance with upsampling, we can further improve sample quality on high-resolution conditional image synthesis.

9 Acknowledgements

We thank Alec Radford, Mark Chen, Pranav Shyam and Raul Puri for providing feedback on this work.

References

- [1] David Ackley, Geoffrey Hinton, and Terrence Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.
- [2] Adverb. The big sleep. <https://twitter.com/advadnoun/status/1351038053033406468>, 2021.
- [3] Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv:1801.01973*, 2018.
- [4] Andrew Brock, Theodore Lim, J. M. Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. *arXiv:1609.07093*, 2016.
- [5] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv:1809.11096*, 2018.
- [6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv:2005.14165*, 2020.
- [7] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International Conference on Machine Learning*, pages 1691–1703. PMLR, 2020.
- [8] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J. Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. *arXiv:2009.00713*, 2020.
- [9] Rewon Child. Very deep vaes generalize autoregressive models and can outperform them on images. *arXiv:2011.10650*, 2021.
- [10] Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.
- [11] Harm de Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron Courville. Modulating early visual processing by language. *arXiv:1707.00683*, 2017.
- [12] DeepMind. Biggan-deep 128x128 on tensorflow hub. <https://tfhub.dev/deepmind/biggan-deep-128/1>, 2018.

- [13] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv:2005.00341*, 2020.
- [14] Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. *arXiv:1907.02544*, 2019.
- [15] Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *arXiv:1903.08689*, 2019.
- [16] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *arXiv:1610.07629*, 2017.
- [17] Federico A. Galatolo, Mario G. C. A. Cimino, and Gigliola Vaglini. Generating images from caption and vice versa via clip-guided generative latent space search. *arXiv:2102.01645*, 2021.
- [18] Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P. Kingma. Learning energy-based models by diffusion recovery likelihood. *arXiv:2012.08125*, 2020.
- [19] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv:1406.2661*, 2014.
- [20] Google. Cloud tpus. <https://cloud.google.com/tpu/>, 2018.
- [21] Anirudh Goyal, Nan Rosemary Ke, Surya Ganguli, and Yoshua Bengio. Variational walkback: Learning a transition operator as a stochastic recurrent net. *arXiv:1711.02282*, 2017.
- [22] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. *arXiv:1912.03263*, 2019.
- [23] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017.
- [24] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.
- [25] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv:2006.11239*, 2020.
- [26] Alexia Jolicoeur-Martineau, Rémi Piché-Taillefer, Rémi Tachet des Combes, and Ioannis Mitliagkas. Adversarial score matching and improved sampling for image generation. *arXiv:2009.05475*, 2020.
- [27] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *arXiv:arXiv:1812.04948*, 2019.
- [28] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. *arXiv:1912.04958*, 2019.
- [29] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.
- [30] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv:2009.09761*, 2020.
- [31] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. CIFAR-10 (Canadian Institute for Advanced Research), 2009. URL <http://www.cs.toronto.edu/~kriz/cifar.html>.
- [32] Tuomas Kynkänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *arXiv:1904.06991*, 2019.
- [33] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. *arXiv:1611.06612*, 2016.

- [34] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.
- [35] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv:1711.05101*, 2017.
- [36] Mario Lucic, Michael Tschannen, Marvin Ritter, Xiaohua Zhai, Olivier Bachem, and Sylvain Gelly. High-fidelity image generation with fewer labels. *arXiv:1903.02271*, 2019.
- [37] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv:2101.02388*, 2021.
- [38] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. *arXiv:1710.03740*, 2017.
- [39] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv:1411.1784*, 2014.
- [40] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. *arXiv:1802.05637*, 2018.
- [41] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv:1802.05957*, 2018.
- [42] Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W. Battaglia. Generating images with sparse representations. *arXiv:2103.03841*, 2021.
- [43] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *arXiv:2102.09672*, 2021.
- [44] NVIDIA. Stylegan2. <https://github.com/NVlabs/stylegan2>, 2019.
- [45] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On buggy resizing libraries and surprising subtleties in fid calculation. *arXiv:2104.11222*, 2021.
- [46] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv:1912.01703*, 2019.
- [47] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. *arXiv:2103.17249*, 2021.
- [48] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. *arXiv:1709.07871*, 2017.
- [49] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *arXiv:2103.00020*, 2021.
- [50] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv:2102.12092*, 2021.
- [51] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. *arXiv:1906.00446*, 2019.
- [52] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *arXiv:1409.0575*, 2014.
- [53] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J. Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *arXiv:arXiv:2104.07636*, 2021.

- [54] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *arXiv:1606.03498*, 2016.
- [55] Shibani Santurkar, Dimitris Tsipras, Brandon Tran, Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Image synthesis with a single (robust) classifier. *arXiv:1906.09453*, 2019.
- [56] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *arXiv:1503.03585*, 2015.
- [57] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv:2010.02502*, 2020.
- [58] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *arXiv:2006.09011*, 2020.
- [59] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *arXiv:arXiv:1907.05600*, 2020.
- [60] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv:2011.13456*, 2020.
- [61] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv:1312.6199*, 2013.
- [62] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *arXiv:1512.00567*, 2015.
- [63] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *arXiv:2007.03898*, 2020.
- [64] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv:1609.03499*, 2016.
- [65] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *arXiv:1711.00937*, 2017.
- [66] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv:1706.03762*, 2017.
- [67] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.
- [68] Yan Wu, Jeff Donahue, David Balduzzi, Karen Simonyan, and Timothy Lillicrap. Logan: Latent optimisation for generative adversarial networks. *arXiv:1912.00953*, 2019.
- [69] Yuxin Wu and Kaiming He. Group normalization. *arXiv:1803.08494*, 2018.
- [70] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. A theory of generative convnet. *arXiv:1602.03264*, 2016.
- [71] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv:1506.03365*, 2015.
- [72] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *arXiv:1612.03242*, 2016.
- [73] Ligeng Zhu. Thop. <https://github.com/Lyken17/pytorch-OpCounter>, 2018.

A Computational Requirements

Compute is essential to modern machine learning applications, and more compute typically yields better results. It is thus important to compare our method’s compute requirements to competing methods. In this section, we demonstrate that we can achieve results better than StyleGAN2 and BigGAN-deep with the same or lower compute budget.

A.1 Throughput

We first benchmark the throughput of our models in Table 7. For the theoretical throughput, we measure the theoretical FLOPs for our model using THOP [73], and assume 100% utilization of an NVIDIA Tesla V100 (120 TFLOPs), while for the actual throughput we use measured wall-clock time. We include communication time across two machines whenever our training batch size doesn’t fit on a single machine, where each of our machines has 8 V100s.

We find that a naive implementation of our models in PyTorch 1.7 is very inefficient, utilizing only 20-30% of the hardware. We also benchmark our optimized version, which use larger per-GPU batch sizes, fused GroupNorm-Swish and fused Adam CUDA ops. For our ImageNet 128×128 model in particular, we find that we can increase the per-GPU batch size from 4 to 32 while still fitting in GPU memory, and this makes a large utilization difference. Our implementation is still far from optimal, and further optimizations should allow us to reach higher levels of utilization.

| Model | Implementation | Batch Size per GPU | Throughput Imgs per V100-sec | Utilization |
|-----------|----------------|-----------------------|---------------------------------|-------------|
| 64×64 | Theoretical | - | 182.3 | 100% |
| | Naive | 32 | 37.0 | 20% |
| | Optimized | 96 | 74.1 | 41% |
| 128×128 | Theoretical | - | 65.2 | 100% |
| | Naive | 4 | 11.5 | 18% |
| | Optimized | 32 | 24.8 | 38% |
| 256×256 | Theoretical | - | 17.9 | 100% |
| | Naive | 4 | 4.4 | 25% |
| | Optimized | 8 | 6.4 | 36% |
| 64 → 256 | Theoretical | - | 31.7 | 100% |
| | Naive | 4 | 6.3 | 20% |
| | Optimized | 12 | 9.5 | 30% |
| 128 → 512 | Theoretical | - | 8.0 | 100% |
| | Naive | 2 | 1.9 | 24% |
| | Optimized | 2 | 2.3 | 29% |

Table 7: Throughput of our ImageNet models, measured in Images per V100-sec.

A.2 Early stopping

In addition, we can train for many fewer iterations while maintaining sample quality superior to BigGAN-deep. Table 8 and 9 evaluate our ImageNet 128×128 and 256×256 models throughout training. We can see that the ImageNet 128×128 model beats BigGAN-deep’s FID (6.02) after 500K training iterations, only one eighth of the way through training. Similarly, the ImageNet 256×256 model beats BigGAN-deep after 750K iterations, roughly a third of the way through training.

| Iterations | FID | sFID | Precision | Recall |
|------------|------|------|-----------|--------|
| 250K | 7.97 | 6.48 | 0.80 | 0.50 |
| 500K | 5.31 | 5.97 | 0.83 | 0.49 |
| 1000K | 4.10 | 5.80 | 0.81 | 0.51 |
| 2000K | 3.42 | 5.69 | 0.83 | 0.53 |
| 4360K | 3.09 | 5.59 | 0.82 | 0.54 |

Table 8: Evaluating an ImageNet 128×128 model throughout training (classifier scale 1.0).

| Iterations | FID | sFID | Precision | Recall |
|------------|-------|------|-----------|--------|
| 250K | 12.21 | 6.15 | 0.78 | 0.50 |
| 500K | 7.95 | 5.51 | 0.81 | 0.50 |
| 750K | 6.49 | 5.39 | 0.81 | 0.50 |
| 1000K | 5.74 | 5.29 | 0.81 | 0.52 |
| 1500K | 5.01 | 5.20 | 0.82 | 0.52 |
| 1980K | 4.59 | 5.25 | 0.82 | 0.52 |

Table 9: Evaluating an ImageNet 256×256 model throughout training (classifier scale 1.0).

A.3 Compute comparison

Finally, in Table 10 we compare the compute of our models with StyleGAN2 and BigGAN-deep, and show we can obtain better FIDs with a similar compute budget. For BigGAN-deep, Brock et al. [5] do not explicitly describe the compute requirements for training their models, but rather provide rough estimates in terms of days on a Google TPUv3 pod [20]. We convert their TPU-v3 estimates to V100 days according to 2 TPU-v3 day = 1 V100 day. For StyleGAN2, we use the reported throughput of 25M images over 32 days 13 hour on one V100 for config-f [44]. We note that our classifier training is relatively lightweight compared to training the generative model.

| Model | Generator Compute | Classifier Compute | Total Compute | FID | sFID | Precision | Recall |
|---|----------------------|-----------------------|------------------|-------------|-------------|-------------|-------------|
| LSUN Horse 256×256 | | | | | | | |
| StyleGAN2 [28] | | | 130 | 3.84 | 6.46 | 0.63 | 0.48 |
| ADM (250K) | 116 | - | 116 | 2.95 | 5.94 | 0.69 | 0.55 |
| ADM (dropout, 250K) | 116 | - | 116 | 2.57 | 6.81 | 0.71 | 0.55 |
| LSUN Cat 256×256 | | | | | | | |
| StyleGAN2 [28] | | | 115 | 7.25 | 6.33 | 0.58 | 0.43 |
| ADM (dropout, 200K) | 92 | - | 92 | 5.57 | 6.69 | 0.63 | 0.52 |
| ImageNet 128×128 | | | | | | | |
| BigGAN-deep [5] | | | 64-128 | 6.02 | 7.18 | 0.86 | 0.35 |
| ADM-G (4360K) | 521 | 9 | 530 | 3.09 | 5.59 | 0.82 | 0.54 |
| ADM-G (450K) | 54 | 9 | 63 | 5.67 | 6.19 | 0.82 | 0.49 |
| ImageNet 256×256 | | | | | | | |
| BigGAN-deep [5] | | | 128-256 | 6.95 | 7.36 | 0.87 | 0.28 |
| ADM-G (1980K) | 916 | 46 | 962 | 4.59 | 5.25 | 0.82 | 0.52 |
| ADM-G (750K) | 347 | 46 | 393 | 6.49 | 5.39 | 0.81 | 0.50 |
| ADM-G (750K) | 347 | 14 [†] | 361 | 6.68 | 5.34 | 0.81 | 0.51 |
| ADM-G (540K), ADM-U (500K) | 329 | 30 | 359 | 3.85 | 5.86 | 0.84 | 0.53 |
| ADM-G (540K), ADM-U (150K) | 219 | 30 | 249 | 4.15 | 6.14 | 0.82 | 0.54 |
| ADM-G (200K), ADM-U (150K) | 110 | 10 [‡] | 126 | 4.93 | 5.82 | 0.82 | 0.52 |
| ImageNet 512×512 | | | | | | | |
| BigGAN-deep [5] | | | 256-512 | 8.43 | 8.13 | 0.88 | 0.29 |
| ADM-G (4360K), ADM-U (1050K) | 1878 | 36 | 1914 | 3.85 | 5.86 | 0.84 | 0.53 |
| ADM-G (500K), ADM-U (100K) | 189 | 9* | 198 | 7.59 | 6.84 | 0.84 | 0.53 |

Table 10: Training compute requirements for our diffusion models compared to StyleGAN2 and BigGAN-deep. Training iterations for each diffusion model are mentioned in parenthesis. Compute is measured in V100-days. [†]ImageNet 256×256 classifier with 150K iterations (instead of 500K). [‡]ImageNet 64×64 classifier with batch size 256 (instead of 1024). *ImageNet 128×128 classifier with batch size 256 (instead of 1024).

B Detailed Formulation of DDPM

Here, we provide a detailed review of the formulation of Gaussian diffusion models from Ho et al. [25]. We start by defining our data distribution $x_0 \sim q(x_0)$ and a Markovian noising process q which gradually adds noise to the data to produce noised samples x_1 through x_T . In particular, each step of the noising process adds Gaussian noise according to some variance schedule given by β_t :

$$q(x_t|x_{t-1}) := \mathcal{N}(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t \mathbf{I}) \quad (15)$$

Ho et al. [25] note that we need not apply q repeatedly to sample from $x_t \sim q(x_t|x_0)$. Instead, $q(x_t|x_0)$ can be expressed as a Gaussian distribution. With $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=0}^t \alpha_s$

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)\mathbf{I}) \quad (16)$$

$$= \sqrt{\bar{\alpha}_t}x_0 + \epsilon \sqrt{1 - \bar{\alpha}_t}, \epsilon \sim \mathcal{N}(0, \mathbf{I}) \quad (17)$$

Here, $1 - \bar{\alpha}_t$ tells us the variance of the noise for an arbitrary timestep, and we could equivalently use this to define the noise schedule instead of β_t .

Using Bayes theorem, one finds that the posterior $q(x_{t-1}|x_t, x_0)$ is also a Gaussian with mean $\tilde{\mu}_t(x_t, x_0)$ and variance $\tilde{\beta}_t$ defined as follows:

$$\tilde{\mu}_t(x_t, x_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}x_t \quad (18)$$

$$\tilde{\beta}_t := \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t}\beta_t \quad (19)$$

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t \mathbf{I}) \quad (20)$$

If we wish to sample from the data distribution $q(x_0)$, we can first sample from $q(x_T)$ and then sample reverse steps $q(x_{t-1}|x_t)$ until we reach x_0 . Under reasonable settings for β_t and T , the distribution $q(x_T)$ is nearly an isotropic Gaussian distribution, so sampling x_T is trivial. All that is left is to approximate $q(x_{t-1}|x_t)$ using a neural network, since it cannot be computed exactly when the data distribution is unknown. To this end, Sohl-Dickstein et al. [56] note that $q(x_{t-1}|x_t)$ approaches a diagonal Gaussian distribution as $T \rightarrow \infty$ and correspondingly $\beta_t \rightarrow 0$, so it is sufficient to train a neural network to predict a mean μ_θ and a diagonal covariance matrix Σ_θ :

$$p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \quad (21)$$

To train this model such that $p(x_0)$ learns the true data distribution $q(x_0)$, we can optimize the following variational lower-bound L_{vlb} for $p_\theta(x_0)$:

$$L_{\text{vlb}} := L_0 + L_1 + \dots + L_{T-1} + L_T \quad (22)$$

$$L_0 := -\log p_\theta(x_0|x_1) \quad (23)$$

$$L_{t-1} := D_{KL}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t)) \quad (24)$$

$$L_T := D_{KL}(q(x_T|x_0) \parallel p(x_T)) \quad (25)$$

While the above objective is well-justified, Ho et al. [25] found that a different objective produces better samples in practice. In particular, they do not directly parameterize $\mu_\theta(x_t, t)$ as a neural network, but instead train a model $\epsilon_\theta(x_t, t)$ to predict ϵ from Equation 17. This simplified objective is defined as follows:

$$L_{\text{simple}} := E_{t \sim [1, T], x_0 \sim q(x_0), \epsilon \sim \mathcal{N}(0, \mathbf{I})} [\|\epsilon - \epsilon_\theta(x_t, t)\|^2] \quad (26)$$

During sampling, we can use substitution to derive $\mu_\theta(x_t, t)$ from $\epsilon_\theta(x_t, t)$:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) \quad (27)$$

Note that L_{simple} does not provide any learning signal for $\Sigma_\theta(x_t, t)$. Ho et al. [25] find that instead of learning $\Sigma_\theta(x_t, t)$, they can fix it to a constant, choosing either $\beta_t \mathbf{I}$ or $\tilde{\beta}_t \mathbf{I}$. These values correspond to upper and lower bounds for the true reverse step variance.

C Nearest Neighbors for Samples

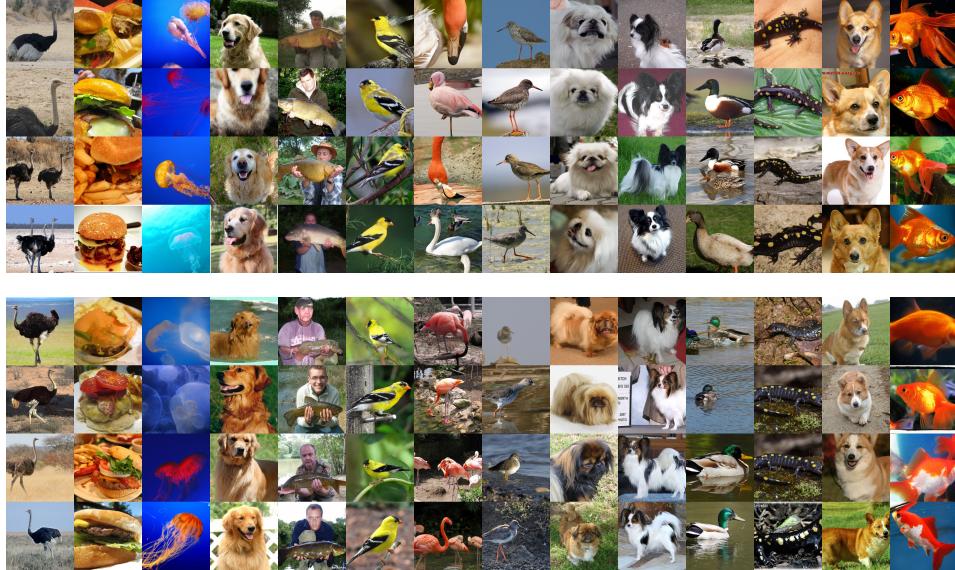


Figure 7: Nearest neighbors for samples from a classifier guided model on ImageNet 256×256. For each image, the top row is a sample, and the remaining rows are the top 3 nearest neighbors from the dataset. The top samples were generated with classifier scale 1 and 250 diffusion sampling steps (FID 4.59). The bottom samples were generated with classifier scale 2.5 and 25 DDIM steps (FID 5.44).

Our models achieve their best FID when using a classifier to reduce the diversity of the generations. One might fear that such a process could cause the model to recall existing images from the training dataset, especially as the classifier scale is increased. To test this, we looked at the nearest neighbors (in InceptionV3 [62] feature space) for a handful of samples. Figure 7 shows our results, revealing that the samples are indeed unique and not stored in the training set.

D Effect of Varying the Classifier Scale



Figure 8: Samples when increasing the classifier scale from 0.0 (left) to 5.5 (right). Each row corresponds to a fixed noise seed. We observe that the classifier drastically changes some images, while leaving others relatively unaffected.

E LSUN Diversity Comparison

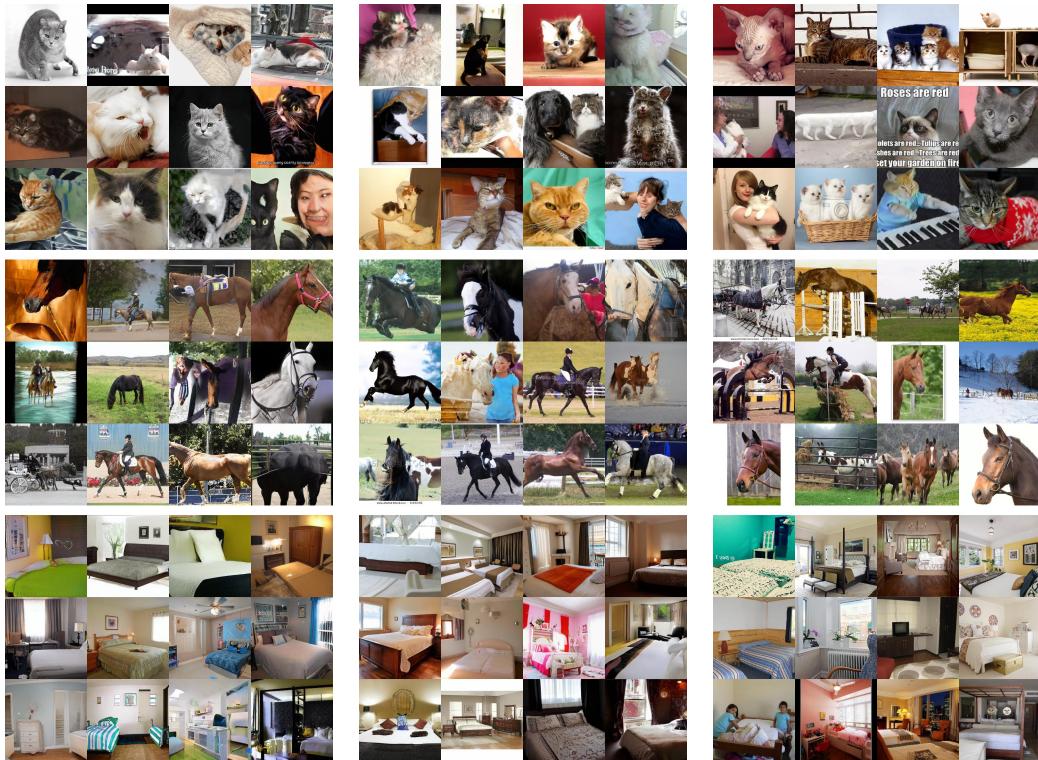


Figure 9: Samples from StyleGAN2 (or StyleGAN for bedrooms) with truncation 1.0 (left) vs samples from our diffusion models (middle) and samples from the training set (right).

F Interpolating Between Dataset Images Using DDIM

The DDIM [57] sampling process is deterministic given the initial noise x_T , thus giving rise to an implicit latent space. It corresponds to integrating an ODE in the forward direction, and we can run the process in reverse to get the latents that produce a given real image. Here, we experiment with encoding real images into this latent space and then interpolating between them.

Equation 13 for the generative pass in DDIM looks like

$$x_{t-1} - x_t = \sqrt{\bar{\alpha}_{t-1}} \left[\left(\sqrt{1/\bar{\alpha}_t} - \sqrt{1/\bar{\alpha}_{t-1}} \right) x_t + \left(\sqrt{1/\bar{\alpha}_{t-1}-1} - \sqrt{1/\bar{\alpha}_t-1} \right) \epsilon_\theta(x_t) \right]$$

Thus, in the limit of small steps, we can expect the reversal of this ODE in the forward direction looks like

$$x_{t+1} - x_t = \sqrt{\bar{\alpha}_{t+1}} \left[\left(\sqrt{1/\bar{\alpha}_t} - \sqrt{1/\bar{\alpha}_{t+1}} \right) x_t + \left(\sqrt{1/\bar{\alpha}_{t+1}-1} - \sqrt{1/\bar{\alpha}_t-1} \right) \epsilon_\theta(x_t) \right]$$

We found that this reverse ODE approximation gives latents with reasonable reconstructions, even with as few as 250 reverse steps. However, we noticed some noise artifacts when reversing all 250 steps, and find that reversing the first 249 steps gives much better reconstructions. To interpolate the latents, class embeddings, and classifier log probabilities, we use $\cos(\theta)x_0 + \sin(\theta)x_1$ where θ sweeps linearly from 0 to $\frac{\pi}{2}$.

Figures 10a through 10c show DDIM latent space interpolations on a class-conditional 256×256 model, while varying the classifier scale. The left and rightmost images are ground truth dataset examples, and between them are reconstructed interpolations in DDIM latent space (including both endpoints). We see that the model with no guidance has almost perfect reconstructions due to its high recall, whereas raising the guidance scale to 2.5 only finds approximately similar reconstructions.

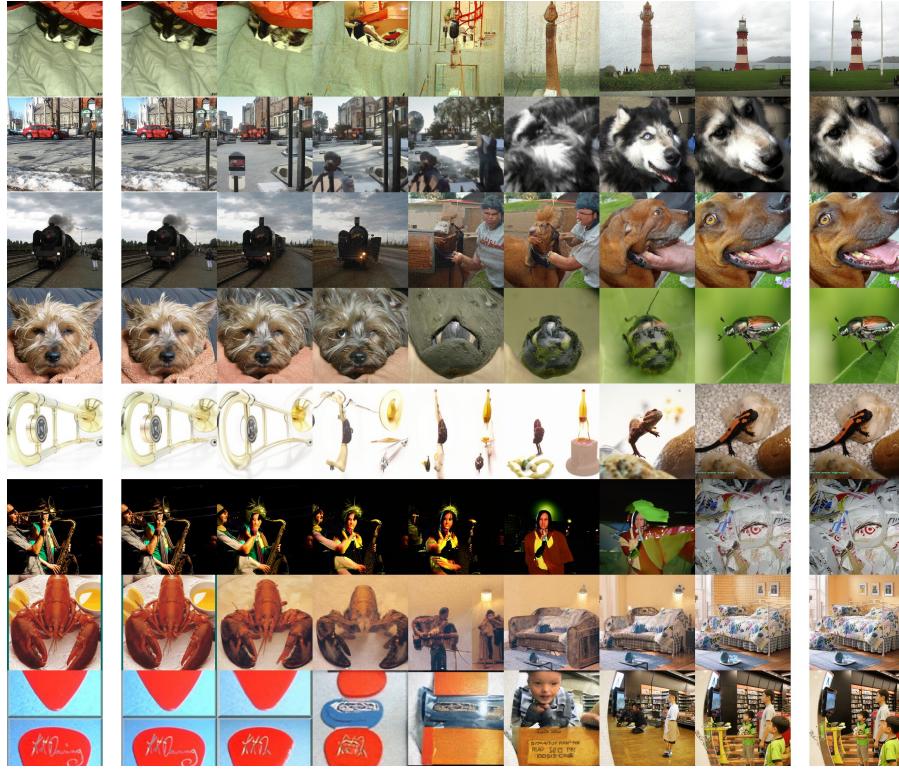


Figure 10a: DDIM latent reconstructions and interpolations on real images with no classifier guidance.

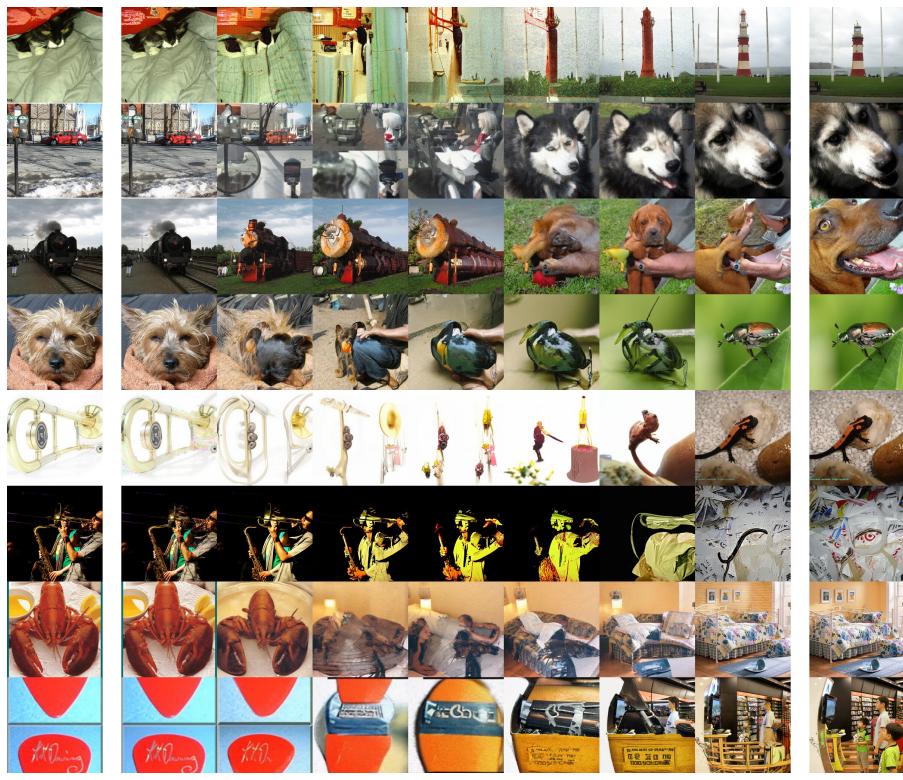


Figure 10b: DDIM latent reconstructions and interpolations on real images with classifier scale 1.0.

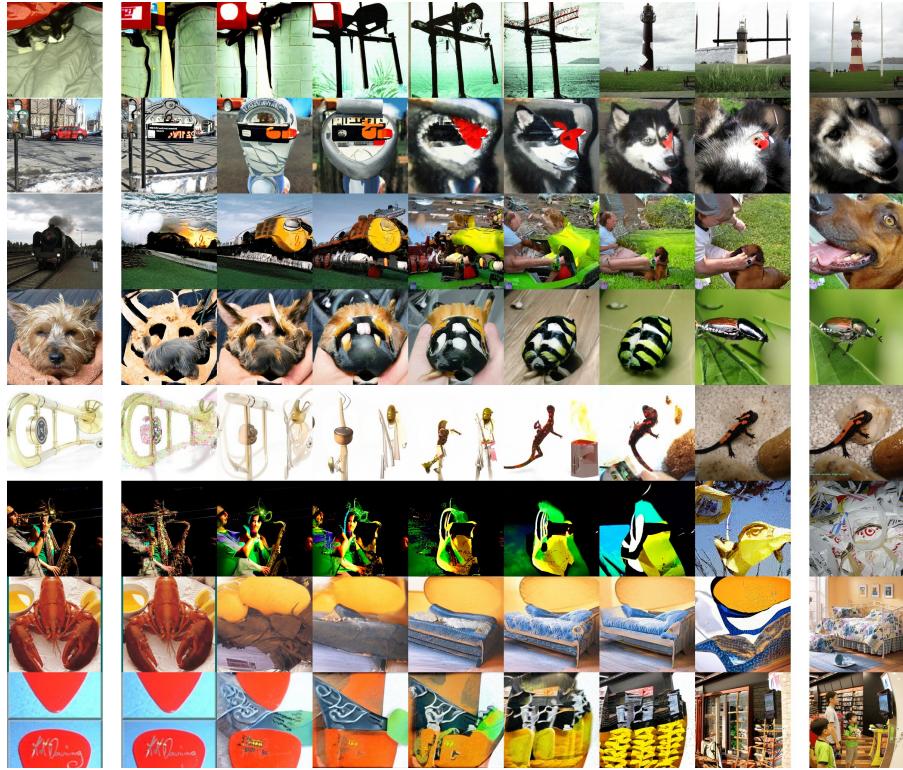


Figure 10c: DDIM latent reconstructions and interpolations on real images with classifier scale 2.5.

G Reduced Temperature Sampling

We achieved our best ImageNet samples by reducing the diversity of our models using classifier guidance. For many classes of generative models, there is a much simpler way to reduce diversity: reducing the temperature [1]. The temperature parameter τ is typically setup so that $\tau = 1.0$ corresponds to standard sampling, and $\tau < 1.0$ focuses more on high-density samples. We experimented with two ways of implementing this for diffusion models: first, by scaling the Gaussian noise used for each transition by τ , and second by dividing $\epsilon_\theta(x_t)$ by τ . The latter implementation makes sense when thinking about ϵ as a re-scaled score function (see Section 4.2), and scaling up the score function is similar to scaling up classifier gradients.

To measure how temperature scaling affects samples, we experimented with our ImageNet 128×128 model, evaluating FID, Precision, and Recall across different temperatures (Figure 11). We find that two techniques behave similarly, and neither technique provides any substantial improvement in our evaluation metrics. We also find that low temperatures have both low precision and low recall, indicating that the model is not focusing on modes of the real data distribution. Figure 12 highlights this effect, indicating that reducing temperature produces blurry, smooth images.

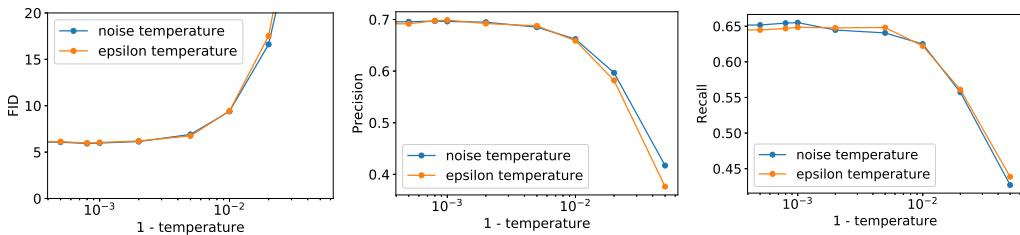


Figure 11: The effect of changing temperature for an ImageNet 128×128 model.

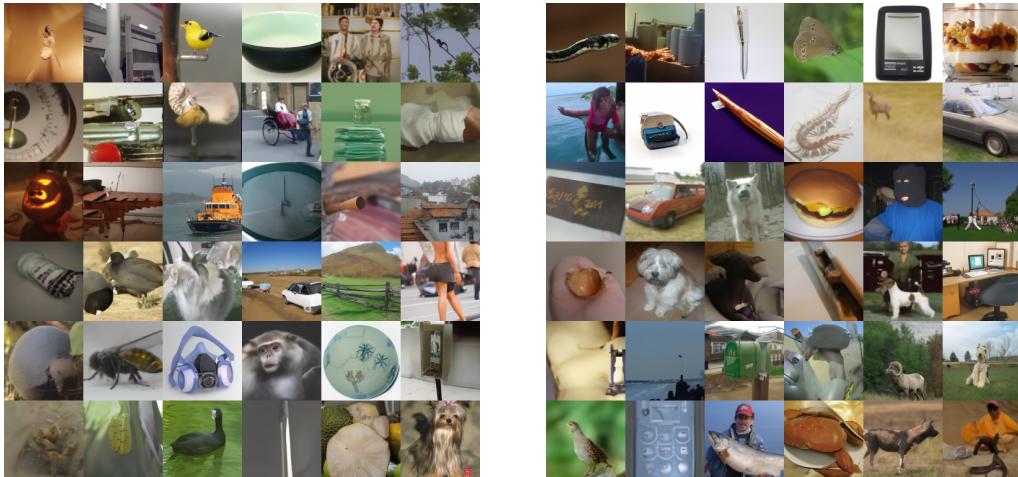


Figure 12: Samples at temperature 0.98 with epsilon scaling (left) and noise scaling (right).

H Conditional Diffusion Process

In this section, we show that conditional sampling can be achieved with a transition operator proportional to $p_\theta(x_t|x_{t+1})p_\phi(y|x_t)$, where $p_\theta(x_t|x_{t+1})$ approximates $q(x_t|x_{t+1})$ and $p_\phi(y|x_t)$ approximates the label distribution for a noised sample x_t .

We start by defining a conditional Markovian noising process \hat{q} similar to q , and assume that $\hat{q}(y|x_0)$ is a known and readily available label distribution for each sample.

$$\hat{q}(x_0) := q(x_0) \quad (28)$$

$$\hat{q}(y|x_0) := \text{Known labels per sample} \quad (29)$$

$$\hat{q}(x_{t+1}|x_t, y) := q(x_{t+1}|x_t) \quad (30)$$

$$\hat{q}(x_{1:T}|x_0, y) := \prod_{t=1}^T \hat{q}(x_t|x_{t-1}, y) \quad (31)$$

While we defined the noising process \hat{q} conditioned on y , we can prove that \hat{q} behaves exactly like q when not conditioned on y . Along these lines, we first derive the unconditional noising operator $\hat{q}(x_{t+1}|x_t)$:

$$\hat{q}(x_{t+1}|x_t) = \int_y \hat{q}(x_{t+1}, y|x_t) dy \quad (32)$$

$$= \int_y \hat{q}(x_{t+1}|x_t, y)\hat{q}(y|x_t) dy \quad (33)$$

$$= \int_y q(x_{t+1}|x_t)\hat{q}(y|x_t) dy \quad (34)$$

$$= q(x_{t+1}|x_t) \int_y \hat{q}(y|x_t) dy \quad (35)$$

$$= q(x_{t+1}|x_t) \quad (36)$$

$$= \hat{q}(x_{t+1}|x_t, y) \quad (37)$$

Following similar logic, we find the joint distribution $\hat{q}(x_{1:T}|x_0)$:

$$\hat{q}(x_{1:T}|x_0) = \int_y \hat{q}(x_{1:T}, y|x_0) dy \quad (38)$$

$$= \int_y \hat{q}(y|x_0)\hat{q}(x_{1:T}|x_0, y) dy \quad (39)$$

$$= \int_y \hat{q}(y|x_0) \prod_{t=1}^T \hat{q}(x_t|x_{t-1}, y) dy \quad (40)$$

$$= \int_y \hat{q}(y|x_0) \prod_{t=1}^T q(x_t|x_{t-1}) dy \quad (41)$$

$$= \prod_{t=1}^T q(x_t|x_{t-1}) \int_y \hat{q}(y|x_0) dy \quad (42)$$

$$= \prod_{t=1}^T q(x_t|x_{t-1}) \quad (43)$$

$$= q(x_{1:T}|x_0) \quad (44)$$

Using Equation 44, we can now derive $\hat{q}(x_t)$:

$$\hat{q}(x_t) = \int_{x_{0:t-1}} \hat{q}(x_0, \dots, x_t) dx_{0:t-1} \quad (45)$$

$$= \int_{x_{0:t-1}} \hat{q}(x_0) \hat{q}(x_1, \dots, x_t | x_0) dx_{0:t-1} \quad (46)$$

$$= \int_{x_{0:t-1}} q(x_0) q(x_1, \dots, x_t | x_0) dx_{0:t-1} \quad (47)$$

$$= \int_{x_{0:t-1}} q(x_0, \dots, x_t) dx_{0:t-1} \quad (48)$$

$$= q(x_t) \quad (49)$$

$$(50)$$

Using the identities $\hat{q}(x_t) = q(x_t)$ and $\hat{q}(x_{t+1}|x_t) = q(x_{t+1}|x_t)$, it is trivial to show via Bayes rule that the unconditional reverse process $\hat{q}(x_t|x_{t+1}) = q(x_t|x_{t+1})$.

One observation about \hat{q} is that it gives rise to a noisy classification function, $\hat{q}(y|x_t)$. We can show that this classification distribution does not depend on x_{t+1} (a noisier version of x_t), a fact which we will later use:

$$\hat{q}(y|x_t, x_{t+1}) = \hat{q}(x_{t+1}|x_t, y) \frac{\hat{q}(y|x_t)}{\hat{q}(x_{t+1}|x_t)} \quad (51)$$

$$= \hat{q}(x_{t+1}|x_t) \frac{\hat{q}(y|x_t)}{\hat{q}(x_{t+1}|x_t)} \quad (52)$$

$$= \hat{q}(y|x_t) \quad (53)$$

$$(54)$$

We can now derive the conditional reverse process:

$$\hat{q}(x_t|x_{t+1}, y) = \frac{\hat{q}(x_t, x_{t+1}, y)}{\hat{q}(x_{t+1}, y)} \quad (55)$$

$$= \frac{\hat{q}(x_t, x_{t+1}, y)}{\hat{q}(y|x_{t+1}) \hat{q}(x_{t+1})} \quad (56)$$

$$= \frac{\hat{q}(x_t|x_{t+1}) \hat{q}(y|x_t, x_{t+1}) \hat{q}(x_{t+1})}{\hat{q}(y|x_{t+1}) \hat{q}(x_{t+1})} \quad (57)$$

$$= \frac{\hat{q}(x_t|x_{t+1}) \hat{q}(y|x_t, x_{t+1})}{\hat{q}(y|x_{t+1})} \quad (58)$$

$$= \frac{\hat{q}(x_t|x_{t+1}) \hat{q}(y|x_t)}{\hat{q}(y|x_{t+1})} \quad (59)$$

$$= \frac{q(x_t|x_{t+1}) \hat{q}(y|x_t)}{\hat{q}(y|x_{t+1})} \quad (60)$$

$$(61)$$

The $\hat{q}(y|x_{t+1})$ term can be treated as a constant since it does not depend on x_t . We thus want to sample from the distribution $Z q(x_t|x_{t+1}) \hat{q}(y|x_t)$ where Z is a normalizing constant. We already have a neural network approximation of $q(x_t|x_{t+1})$, called $p_\theta(x_t|x_{t+1})$, so all that is left is an approximation of $\hat{q}(y|x_t)$. This can be obtained by training a classifier $p_\phi(y|x_t)$ on noised images x_t derived by sampling from $q(x_t)$.

I Hyperparameters

When choosing optimal classifier scales for our sampler, we swept over $[0.5, 1, 2]$ for ImageNet 128×128 and ImageNet 256×256 , and $[1, 2, 3, 3.5, 4, 4.5, 5]$ for ImageNet 512×512 . For DDIM, we swept over values $[0.5, 0.75, 1.0, 1.25, 2]$ for ImageNet 128×128 , $[0.5, 1, 1.5, 2, 2.5, 3, 3.5]$ for ImageNet 256×256 , and $[3, 4, 5, 6, 7, 9, 11]$ for ImageNet 512×512 .

Hyperparameters for training the diffusion and classification models are in Table 11 and Table 12 respectively. Hyperparameters for guided sampling are in Table 14. Hyperparameters used to train upsampling models are in Table 13. We train all of our models using Adam [29] or AdamW [35] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We train in 16-bit precision using loss-scaling [38], but maintain 32-bit weights, EMA, and optimizer state. We use an EMA rate of 0.9999 for all experiments. We use PyTorch [46], and train on NVIDIA Tesla V100s.

For all architecture ablations, we train with batch size 256, and sample using 250 sampling steps. For our attention heads ablations, we use 128 base channels, 2 residual blocks per resolution, multi-resolution attention, and BigGAN up/downsampling, and we train the models for 700K iterations. By default, all of our experiments use adaptive group normalization, except when explicitly ablating for it.

When sampling with 1000 timesteps, we use the same noise schedule as for training. On ImageNet, we use the uniform stride from Nichol and Dhariwal [43] for 250 step samples and the slightly different uniform stride from Song et al. [57] for 25 step DDIM.

| | LSUN | ImageNet 64 | ImageNet 128 | ImageNet 256 | ImageNet 512 |
|----------------------|-------------|-------------|--------------|--------------|-----------------|
| Diffusion steps | 1000 | 1000 | 1000 | 1000 | 1000 |
| Noise Schedule | linear | cosine | linear | linear | linear |
| Model size | 552M | 296M | 422M | 554M | 559M |
| Channels | 256 | 192 | 256 | 256 | 256 |
| Depth | 2 | 3 | 2 | 2 | 2 |
| Channels multiple | 1,1,2,2,4,4 | 1,2,3,4 | 1,1,2,3,4 | 1,1,2,2,4,4 | 0.5,1,1,2,2,4,4 |
| Heads | | | 4 | | |
| Heads Channels | 64 | 64 | | 64 | 64 |
| Attention resolution | 32,16,8 | 32,16,8 | 32,16,8 | 32,16,8 | 32,16,8 |
| BigGAN up/downsample | ✓ | ✓ | ✓ | ✓ | ✓ |
| Dropout | 0.1 | 0.1 | 0.0 | 0.0 | 0.0 |
| Batch size | 256 | 2048 | 256 | 256 | 256 |
| Iterations | varies* | 540K | 4360K | 1980K | 1940K |
| Learning Rate | 1e-4 | 3e-4 | 1e-4 | 1e-4 | 1e-4 |

Table 11: Hyperparameters for diffusion models. *We used 200K iterations for LSUN cat, 250K for LSUN horse, and 500K for LSUN bedroom.

| | ImageNet 64 | ImageNet 128 | ImageNet 256 | ImageNet 512 |
|----------------------|-------------|--------------|--------------|-----------------|
| Diffusion steps | 1000 | 1000 | 1000 | 1000 |
| Noise Schedule | cosine | linear | linear | linear |
| Model size | 65M | 43M | 54M | 54M |
| Channels | 128 | 128 | 128 | 128 |
| Depth | 4 | 2 | 2 | 2 |
| Channels multiple | 1,2,3,4 | 1,1,2,3,4 | 1,1,2,2,4,4 | 0.5,1,1,2,2,4,4 |
| Heads Channels | 64 | 64 | 64 | 64 |
| Attention resolution | 32,16,8 | 32,16,8 | 32,16,8 | 32,16,8 |
| BigGAN up/downsample | ✓ | ✓ | ✓ | ✓ |
| Attention pooling | ✓ | ✓ | ✓ | ✓ |
| Weight decay | 0.2 | 0.05 | 0.05 | 0.05 |
| Batch size | 1024 | 256* | 256 | 256 |
| Iterations | 300K | 300K | 500K | 500K |
| Learning rate | 6e-4 | 3e-4* | 3e-4 | 3e-4 |

Table 12: Hyperparameters for classification models. *For our ImageNet $128 \times 128 \rightarrow 512 \times 512$ upsamples, we used a different classifier for the base model, with batch size 1024 and learning rate 6e-5.

| | ImageNet 64 → 256 | ImageNet 128 → 512 |
|----------------------|-------------------|--------------------|
| Diffusion steps | 1000 | 1000 |
| Noise Schedule | linear | linear |
| Model size | 312M | 309M |
| Channels | 192 | 192 |
| Depth | 2 | 2 |
| Channels multiple | 1,1,2,2,4,4 | 1,1,2,2,4,4* |
| Heads | 4 | |
| Heads Channels | | 64 |
| Attention resolution | 32,16,8 | 32,16,8 |
| BigGAN up/downsample | ✓ | ✓ |
| Dropout | 0.0 | 0.0 |
| Batch size | 256 | 256 |
| Iterations | 500K | 1050K |
| Learning Rate | 1e-4 | 1e-4 |

Table 13: Hyperparameters for upsampling diffusion models. *We chose this as an optimization, with the intuition that a lower-resolution path should be unnecessary for upsampling 128x128 images.

| | ImageNet 64 | ImageNet 128 | ImageNet 256 | ImageNet 512 |
|---------------------------------|-------------|--------------|--------------|--------------|
| Gradient Scale (250 steps) | 1.0 | 0.5 | 1.0 | 4.0 |
| Gradient Scale (DDIM, 25 steps) | - | 1.25 | 2.5 | 9.0 |

Table 14: Hyperparameters for classifier-guided sampling.

J Using Fewer Sampling Steps on LSUN

We initially found that our LSUN models achieved much better results when sampling with 1000 steps rather than 250 steps, contrary to previous results from Nichol and Dhariwal [43]. To address this, we conducted a sweep over sampling-time noise schedules, finding that an improved schedule can largely close the gap. We swept over schedules on LSUN bedrooms, and selected the schedule with the best FID for use on the other two datasets. Table 15 details the findings of this sweep, and Table 16 applies this schedule to three LSUN datasets.

While sweeping over sampling schedules is not as expensive as re-training models from scratch, it does require a significant amount of sampling compute. As a result, we did not conduct an exhaustive sweep, and superior schedules are likely to exist.

| Schedule | FID |
|---------------------|-------------|
| 50, 50, 50, 50, 50 | 2.31 |
| 70, 60, 50, 40, 30 | 2.17 |
| 90, 50, 40, 40, 30 | 2.10 |
| 90, 60, 50, 30, 20 | 2.09 |
| 80, 60, 50, 30, 30 | 2.09 |
| 90, 50, 50, 30, 30 | 2.07 |
| 100, 50, 40, 30, 30 | 2.03 |
| 90, 60, 60, 20, 20 | 2.02 |

Table 15: Results of sweeping over 250 step sampling schedules on LSUN bedrooms. The schedule is expressed as a sequence of five integers, where each integer is the number of steps allocated to one fifth of the diffusion process. The first integer corresponding to $t \in [0, 199]$ and the last to $t \in [T - 200, T - 1]$. Thus, 50, 50, 50, 50, 50 is a uniform schedule, and 250, 0, 0, 0, 0 is a schedule where all timesteps are spent near $t = 0$.

| Schedule | FID | sFID | Prec | Rec |
|------------------------------|------|------|------|------|
| LSUN Bedrooms 256×256 | | | | |
| 1000 steps | 1.90 | 5.59 | 0.66 | 0.51 |
| 250 steps (uniform) | 2.31 | 6.12 | 0.65 | 0.50 |
| 250 steps (sweep) | 2.02 | 6.12 | 0.67 | 0.50 |
| LSUN Horses 256×256 | | | | |
| 1000 steps | 2.57 | 6.81 | 0.71 | 0.55 |
| 250 steps (uniform) | 3.45 | 7.55 | 0.68 | 0.56 |
| 250 steps (sweep) | 2.83 | 7.08 | 0.69 | 0.56 |
| LSUN Cat 256×256 | | | | |
| 1000 steps | 5.57 | 6.69 | 0.63 | 0.52 |
| 250 steps (uniform) | 7.03 | 8.24 | 0.60 | 0.53 |
| 250 steps (sweep) | 5.94 | 7.43 | 0.62 | 0.52 |

Table 16: Evaluations on LSUN bedrooms, horses, and cats using different sampling schedules. We find that the sweep schedule produces better results than the uniform 250 step schedule on all three datasets, and mostly bridges the gap to the 1000 step schedule.

K Samples from ImageNet 512×512



Figure 13: Samples from our best 512×512 model (FID: 3.85). Classes are 1: goldfish, 279: arctic fox, 323: monarch butterfly, 386: african elephant, 130: flamingo, 852: tennis ball.

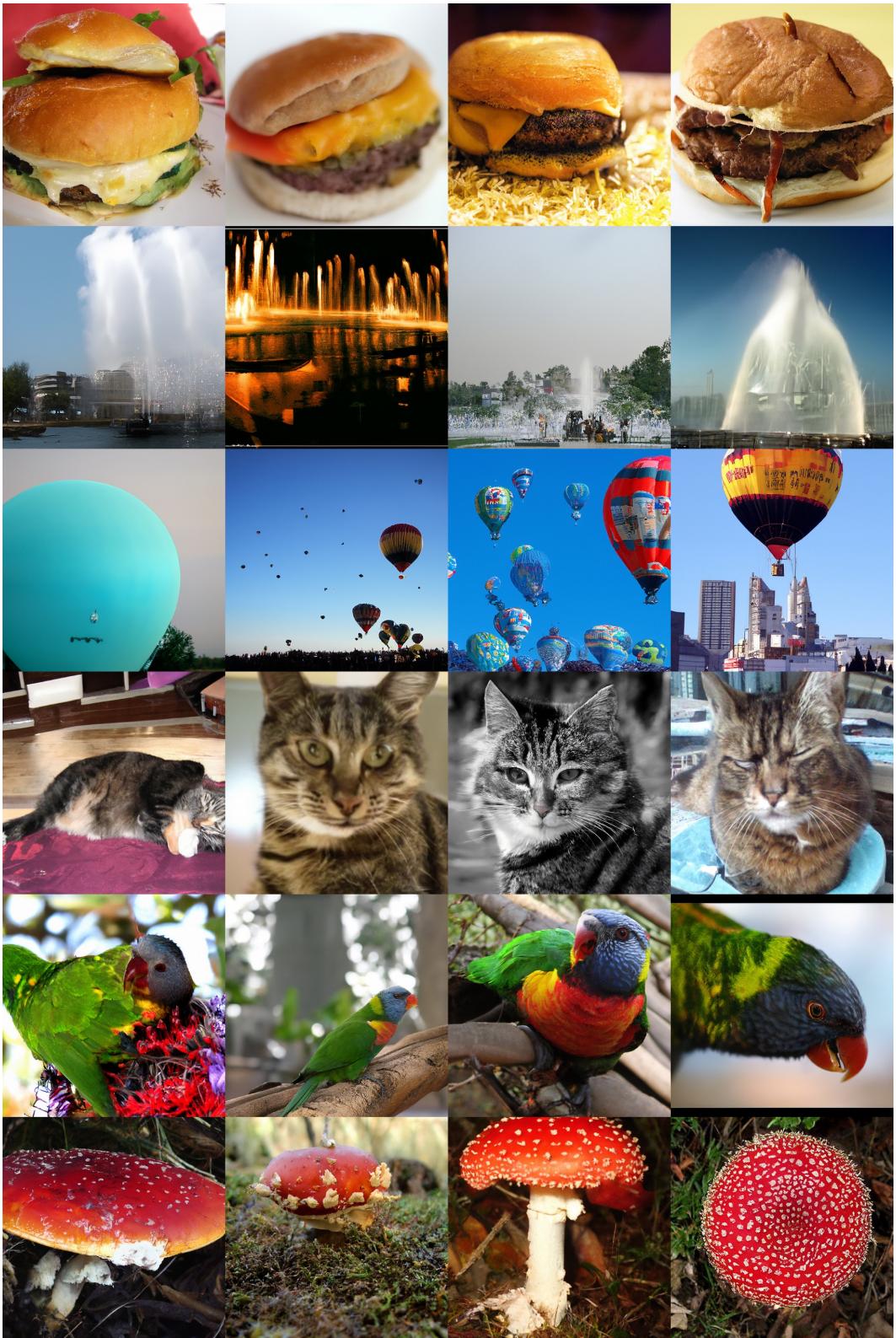


Figure 14: Samples from our best 512×512 model (FID: 3.85). Classes are 933: cheeseburger, 562: fountain, 417: balloon, 281: tabby cat, 90: lorikeet, 992: agaric.



Figure 15: Difficult class samples from our best 512×512 model (FID: 3.85). Classes are 432: bassoon, 468: cab, 424: barbershop, 444: bicycle-built-for-two, 981: ballplayer, 550: espresso maker.



Figure 16: Samples from our guided 512×512 model using 250 steps with classifier scale 4.0 (FID 7.72). Classes are 1: goldfish, 279: arctic fox, 323: monarch butterfly, 386: african elephant, 130: flamingo, 852: tennis ball.



Figure 17: Samples from our guided 512×512 model using 250 steps with classifier scale 4.0 (FID 7.72). Classes are 933: cheeseburger, 562: fountain, 417: balloon, 281: tabby cat, 90: lorikeet, 992: agaric.

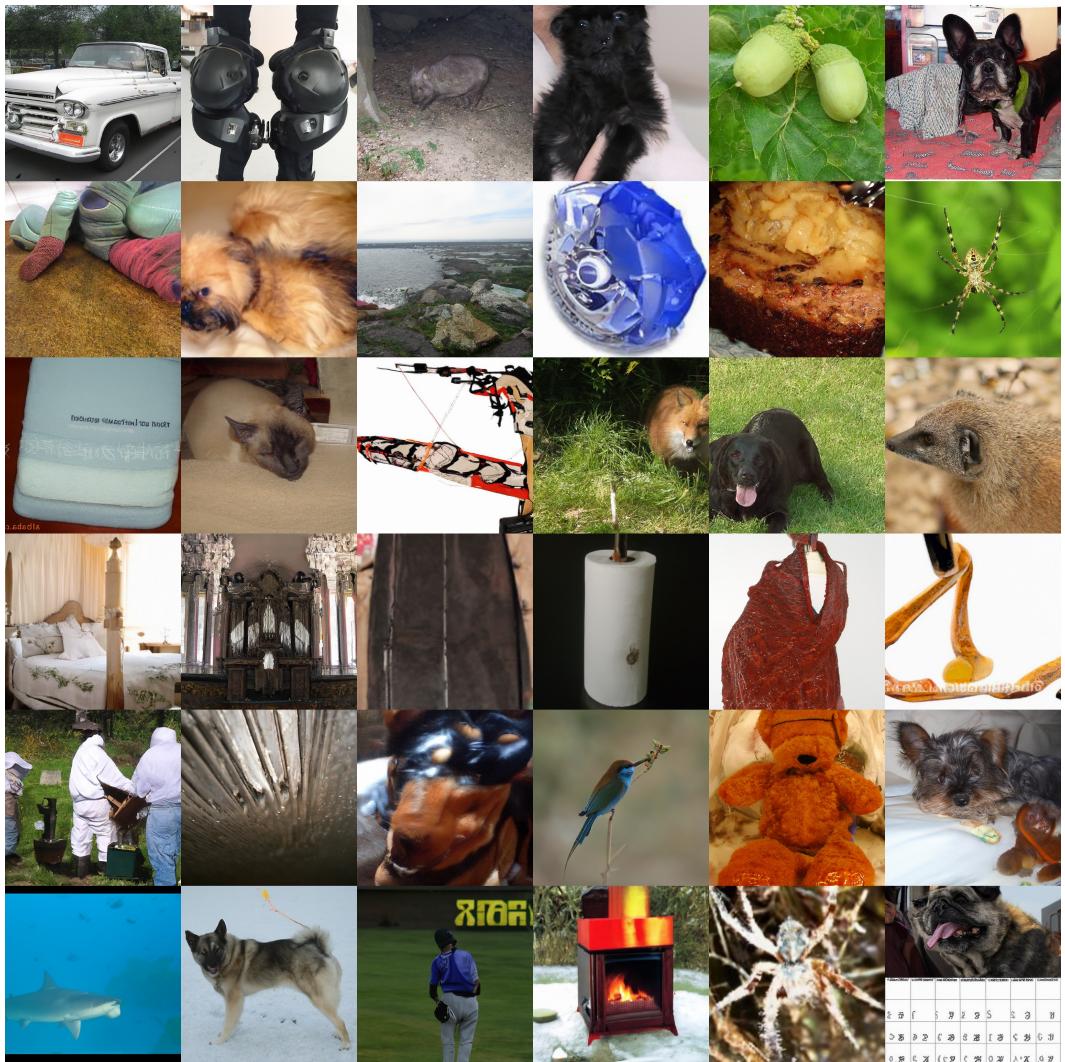


Figure 18: Random samples from our best ImageNet 512×512 model (FID 3.85).



Figure 19: Random samples from our guided 512×512 model using 250 steps with classifier scale 4.0 (FID 7.72).

L Samples from ImageNet 256×256



Figure 20: Samples using our best 256×256 model (FID 3.94). Classes are 1: goldfish, 279: arctic fox, 323: monarch butterfly, 386: african elephant, 130: flamingo, 852: tennis ball, 933: cheeseburger, 562: fountain, 417: balloon, 281: tabby cat, 90: lorikeet, 992: agaric

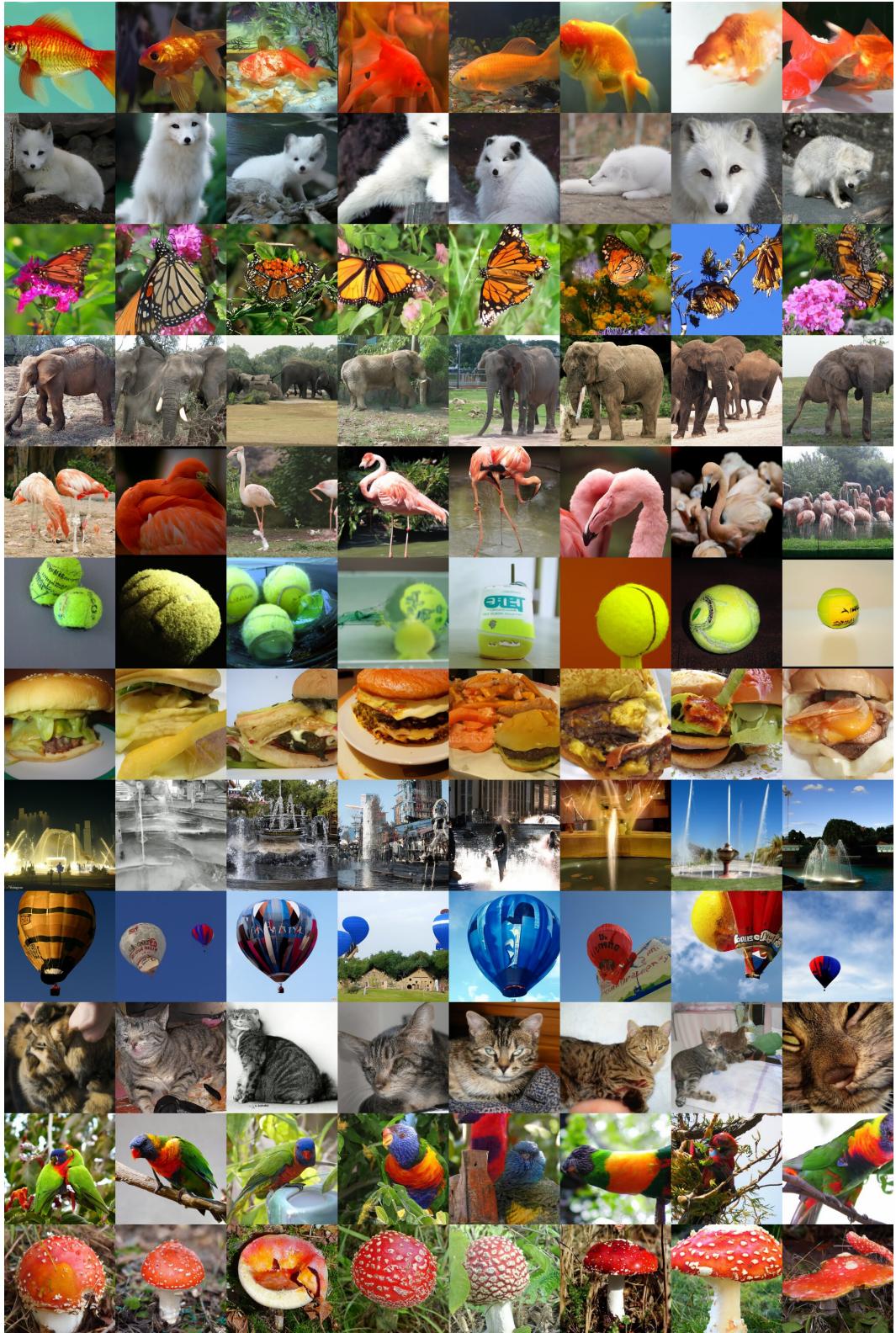


Figure 21: Samples from our guided 256×256 model using 250 steps with classifier scale 1.0 (FID 4.59). Classes are 1: goldfish, 279: arctic fox, 323: monarch butterfly, 386: african elephant, 130: flamingo, 852: tennis ball, 933: cheeseburger, 562: fountain, 417: balloon, 281: tabby cat, 90: lorikeet, 992: agaric

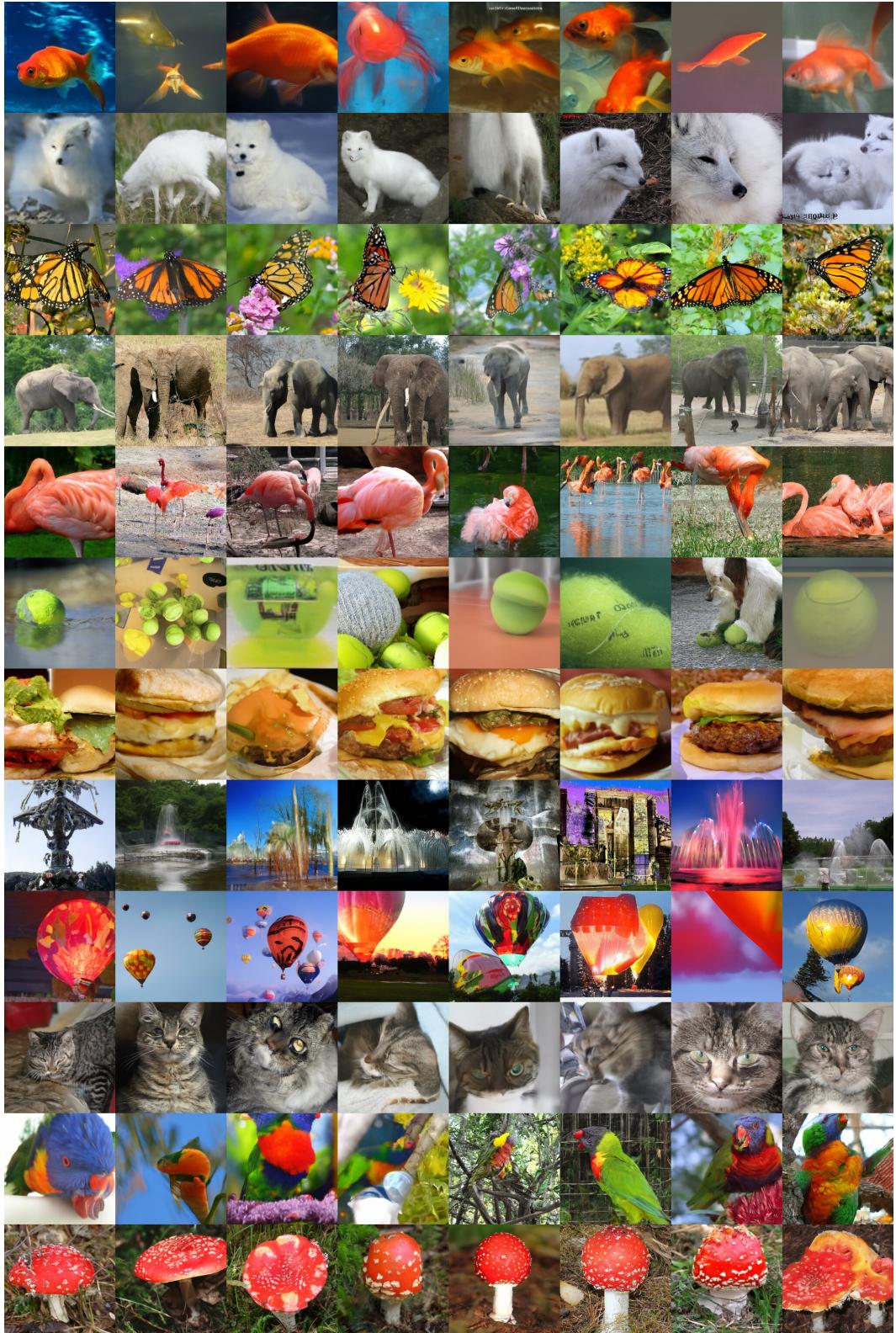


Figure 22: Samples from our guided 256×256 model using 25 DDIM steps with classifier scale 2.5 (FID 5.44). Classes are 1: goldfish, 279: arctic fox, 323: monarch butterfly, 386: african elephant, 130: flamingo, 852: tennis ball, 933: cheeseburger, 562: fountain, 417: balloon, 281: tabby cat, 90: lorikeet, 992: agaric

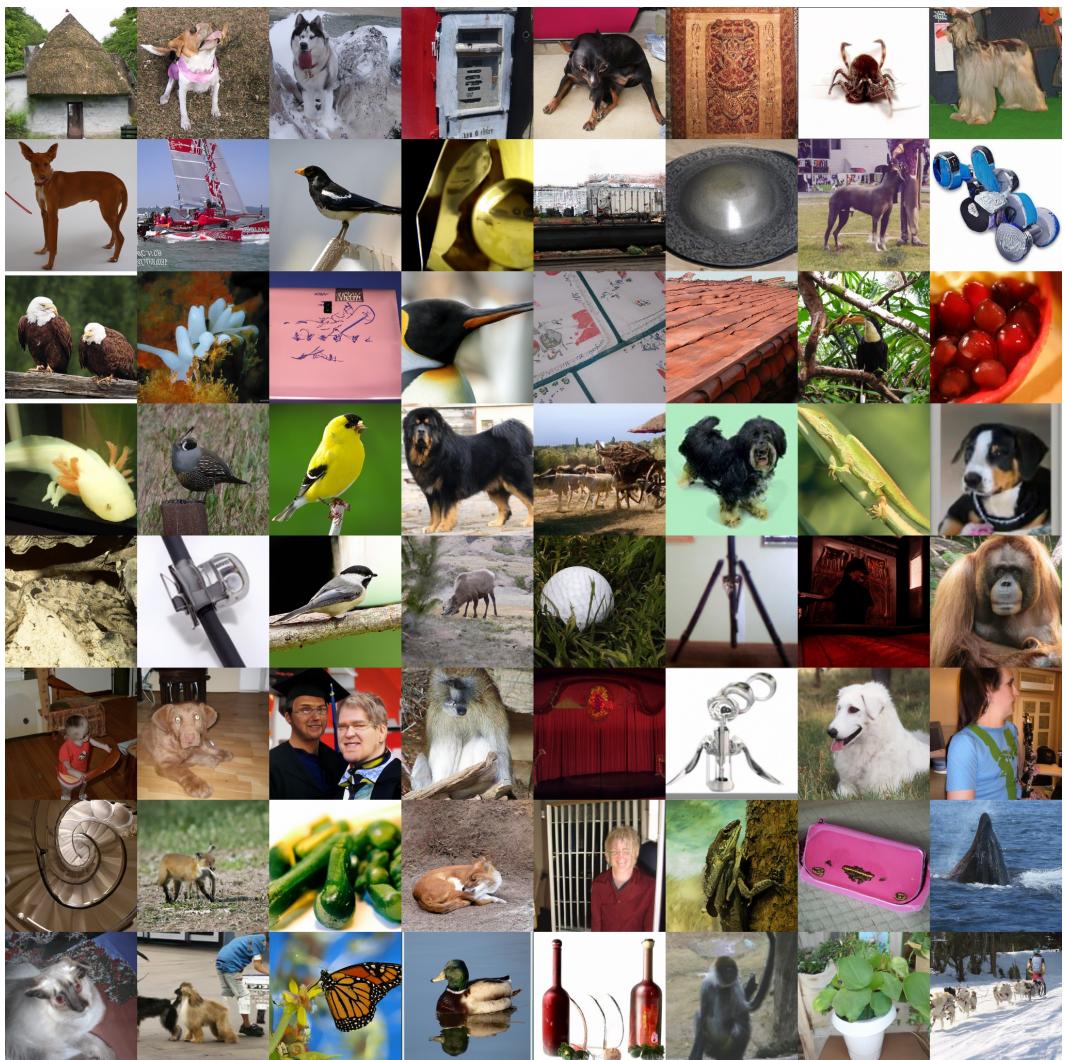


Figure 23: Random samples from our best 256×256 model (FID 3.94).

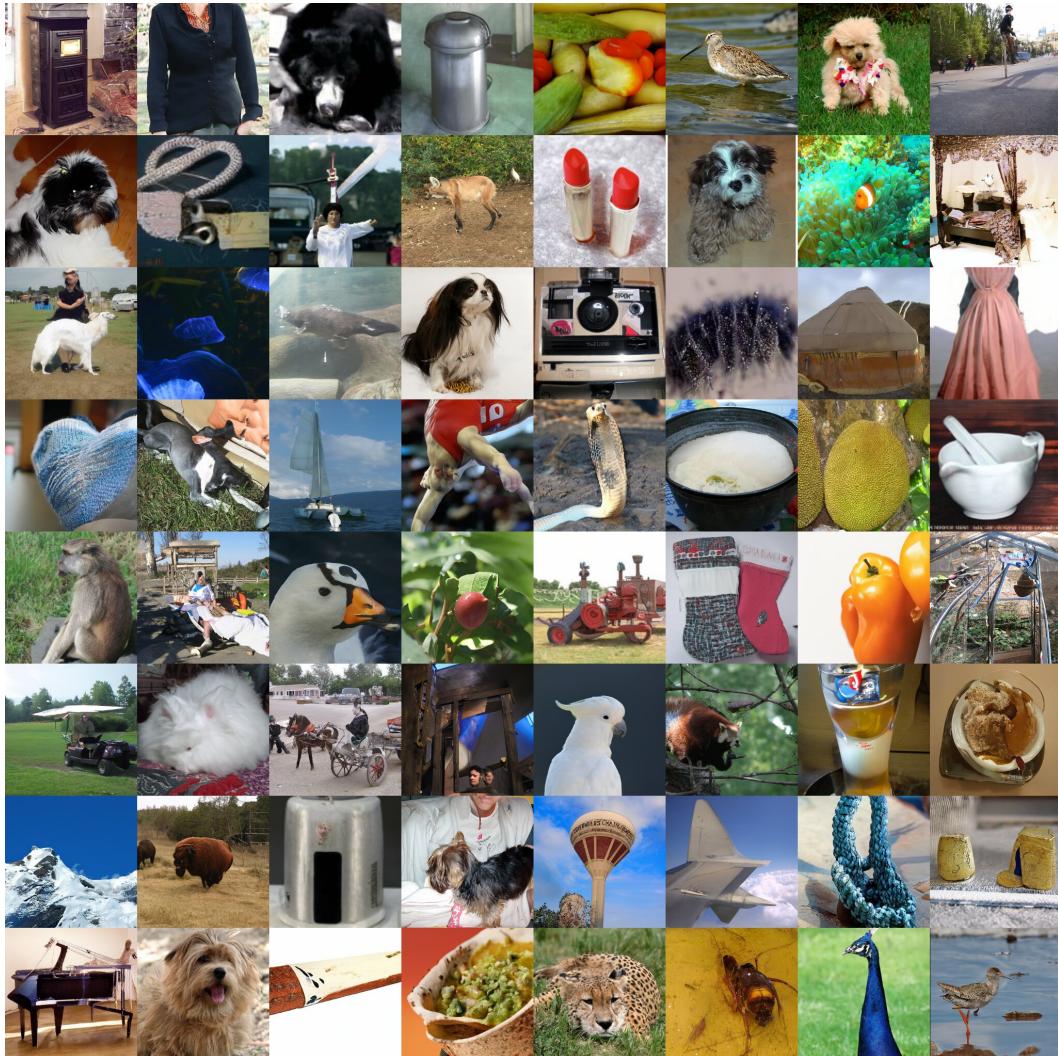


Figure 24: Random samples from our guided 256×256 model using 250 steps with classifier scale 1.0 (FID 4.59).

M Samples from LSUN

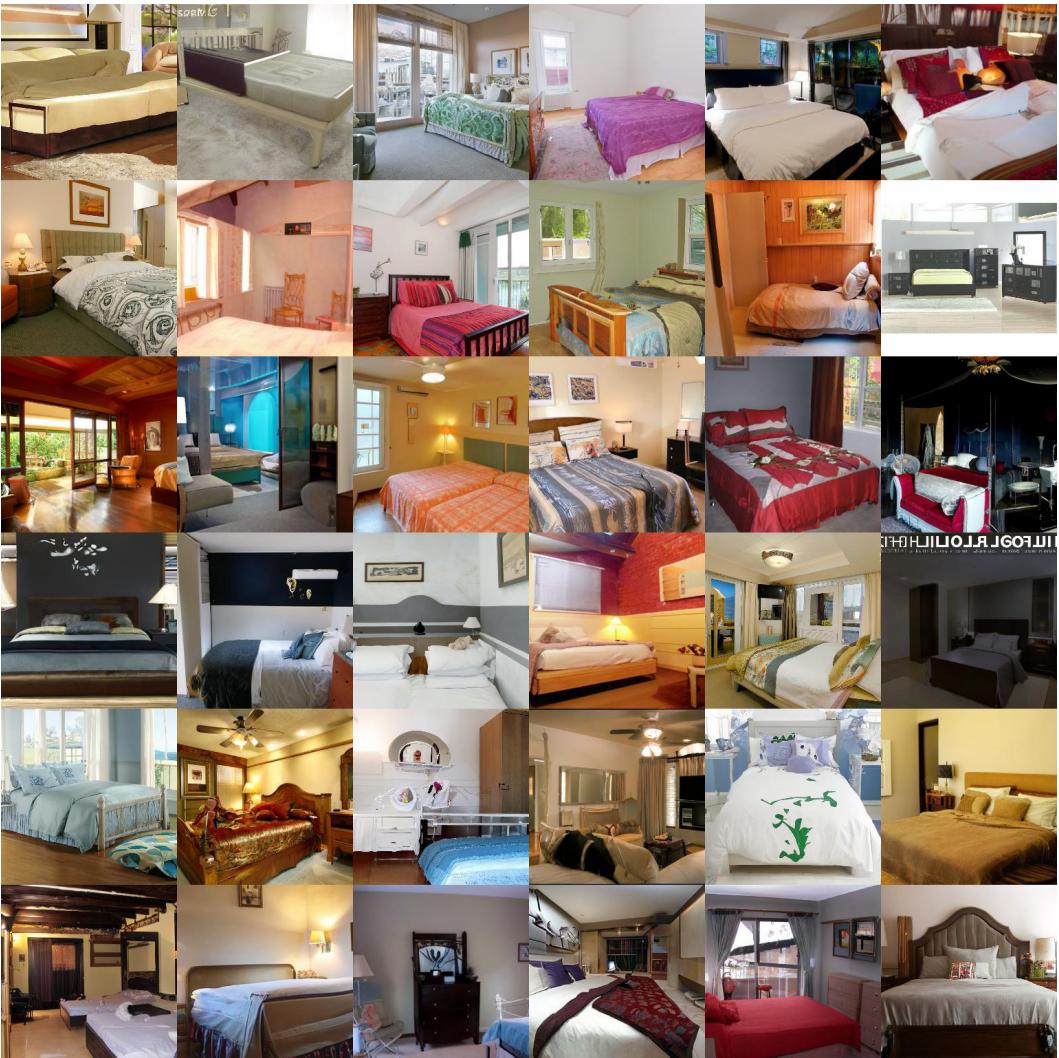


Figure 25: Random samples from our LSUN bedroom model using 1000 sampling steps. (FID 1.90)



Figure 26: Random samples from our LSUN horse model using 1000 sampling steps. (FID 2.57)



Figure 27: Random samples from our LSUN cat model using 1000 sampling steps. (FID 5.57)