Improved Denoising Diffusion Probabilistic Models

Alex Nichol [*1] **Prafulla Dhariwal** [*1]

Abstract

Denoising diffusion probabilistic models (DDPM) are a class of generative models which have recently been shown to produce excellent samples. We show that with a few simple modifications, DDPMs can also achieve competitive log-likelihoods while maintaining high sample quality. Additionally, we find that learning variances of the reverse diffusion process allows sampling with an order of magnitude fewer forward passes with a negligible difference in sample quality, which is important for the practical deployment of these models. We additionally use precision and recall to compare how well DDPMs and GANs cover the target distribution. Finally, we show that the sample quality and likelihood of these models scale smoothly with model capacity and training compute, making them easily scalable. We release our code at `https://github.com/openai/improved-diffusion`.

1. Introduction

Sohl-Dickstein et al. (2015) introduced diffusion probabilistic models, a class of generative models which match a data distribution by learning to reverse a gradual, multi-step noising process. More recently, Ho et al. (2020) showed an equivalence between denoising diffusion probabilistic models (DDPM) and score based generative models (Song & Ermon, 2019; 2020), which learn a gradient of the log-density of the data distribution using denoising score matching (Hyvärinen, 2005). It has recently been shown that this class of models can produce high-quality images (Ho et al., 2020; Song & Ermon, 2020; Jolicoeur-Martineau et al., 2020) and audio (Chen et al., 2020b; Kong et al., 2020), but it has yet to be shown that DDPMs can achieve log-likelihoods competitive with other likelihood-based models such as autoregressive models (van den Oord et al., 2016c) and VAEs (Kingma & Welling, 2013). This raises various questions, such as whether DDPMs are capable of capturing all the modes of a distribution. Furthermore, while Ho et al. (2020) showed extremely good results on the CIFAR-10 (Krizhevsky, 2009) and LSUN (Yu et al., 2015) datasets, it is unclear how well DDPMs scale to datasets with higher diversity such as ImageNet. Finally, while Chen et al. (2020b) found that DDPMs can efficiently generate audio using a small number of sampling steps, it has yet to be shown that the same is true for images.

In this paper, we show that DDPMs can achieve log-likelihoods competitive with other likelihood-based models, even on high-diversity datasets like ImageNet. To more tightly optimise the variational lower-bound (VLB), we learn the reverse process variances using a simple reparameterization and a hybrid learning objective that combines the VLB with the simplified objective from Ho et al. (2020).

We find surprisingly that, with our hybrid objective, our models obtain better log-likelihoods than those obtained by optimizing the log-likelihood directly, and discover that the latter objective has much more gradient noise during training. We show that a simple importance sampling technique reduces this noise and allows us to achieve better log-likelihoods than with the hybrid objective.

After incorporating learned variances into our model, we surprisingly discovered that we could sample in fewer steps from our models with very little change in sample quality. While DDPM (Ho et al., 2020) requires hundreds of forward passes to produce good samples, we can achieve good samples with as few as 50 forward passes, thus speeding up sampling for use in practical applications. In parallel to our work, Song et al. (2020a) develops a different approach to fast sampling, and we compare against their approach, DDIM, in our experiments.

While likelihood is a good metric to compare against other likelihood-based models, we also wanted to compare the distribution coverage of these models with GANs. We use the improved precision and recall metrics (Kynkäänniemi et al., 2019) and discover that diffusion models achieve much higher recall for similar FID, suggesting that they do indeed cover a much larger portion of the target distribution.

Finally, since we expect machine learning models to consume more computational resources in the future, we evaluate the performance of these models as we increase model size and training compute. Similar to (Henighan et al.,

[*]Equal contribution [1]OpenAI, San Francisco, USA. Correspondence to: <alex@openai.com>, <prafulla@openai.com>.

2020), we observe trends that suggest predictable improvements in performance as we increase training compute.

2. Denoising Diffusion Probabilistic Models

We briefly review the formulation of DDPMs from Ho et al. (2020). This formulation makes various simplifying assumptions, such as a fixed noising process q which adds diagonal Gaussian noise at each timestep. For a more general derivation, see Sohl-Dickstein et al. (2015).

2.1. Definitions

Given a data distribution $x_0 \sim q(x_0)$, we define a forward noising process q which produces latents x_1 through x_T by adding Gaussian noise at time t with variance $\beta_t \in (0, 1)$ as follows:

$$q(x_1, ..., x_T|x_0) := \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{1}$$

$$q(x_t|x_{t-1}) := \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathbf{I}) \tag{2}$$

Given sufficiently large T and a well behaved schedule of β_t, the latent x_T is nearly an isotropic Gaussian distribution. Thus, if we know the exact reverse distribution $q(x_{t-1}|x_t)$, we can sample $x_T \sim \mathcal{N}(0, \mathbf{I})$ and run the process in reverse to get a sample from $q(x_0)$. However, since $q(x_{t-1}|x_t)$ depends on the entire data distribution, we approximate it using a neural network as follows:

$$p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{3}$$

The combination of q and p is a variational auto-encoder (Kingma & Welling, 2013), and we can write the variational lower bound (VLB) as follows:

$$L_{\text{vlb}} := L_0 + L_1 + ... + L_{T-1} + L_T \tag{4}$$

$$L_0 := -\log p_\theta(x_0|x_1) \tag{5}$$

$$L_{t-1} := D_{KL}(q(x_{t-1}|x_t, x_0) \,||\, p_\theta(x_{t-1}|x_t)) \tag{6}$$

$$L_T := D_{KL}(q(x_T|x_0) \,||\, p(x_T)) \tag{7}$$

Aside from L_0, each term of Equation 4 is a KL divergence between two Gaussians, and can thus be evaluated in closed form. To evaluate L_0 for images, we assume that each color component is divided into 256 bins, and we compute the probability of $p_\theta(x_0|x_1)$ landing in the correct bin (which is tractable using the CDF of the Gaussian distribution). Also note that while L_T does not depend on θ, it will be close to zero if the forward noising process adequately destroys the data distribution so that $q(x_T|x_0) \approx \mathcal{N}(0, \mathbf{I})$.

As noted in (Ho et al., 2020), the noising process defined in Equation 2 allows us to sample an arbitrary step of the noised latents directly conditioned on the input x_0. With $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=0}^{t} \alpha_s$, we can write the marginal

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{8}$$

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \tag{9}$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$. 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 can calculate the posterior $q(x_{t-1}|x_t, x_0)$ in terms of $\tilde{\beta}_t$ and $\tilde{\mu}_t(x_t, x_0)$ which are defined as follows:

$$\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \tag{10}$$

$$\tilde{\mu}_t(x_t, x_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t \tag{11}$$

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t \mathbf{I}) \tag{12}$$

2.2. Training in Practice

The objective in Equation 4 is a sum of independent terms L_{t-1}, and Equation 9 provides an efficient way to sample from an arbitrary step of the forward noising process and estimate L_{t-1} using the posterior (Equation 12) and prior (Equation 3). We can thus randomly sample t and use the expectation $E_{t,x_0,\epsilon}[L_{t-1}]$ to estimate L_{vlb}. Ho et al. (2020) uniformly sample t for each image in each mini-batch.

There are many different ways to parameterize $\mu_\theta(x_t, t)$ in the prior. The most obvious option is to predict $\mu_\theta(x_t, t)$ directly with a neural network. Alternatively, the network could predict x_0, and this output could be used in Equation 11 to produce $\mu_\theta(x_t, t)$. The network could also predict the noise ϵ and use Equations 9 and 11 to derive

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right) \tag{13}$$

Ho et al. (2020) found that predicting ϵ worked best, especially when combined with a reweighted loss function:

$$L_{\text{simple}} = E_{t,x_0,\epsilon}\left[||\epsilon - \epsilon_\theta(x_t, t)||^2\right] \tag{14}$$

This objective can be seen as a reweighted form of L_{vlb} (without the terms affecting Σ_θ). The authors found that optimizing this reweighted objective resulted in much better sample quality than optimizing L_{vlb} directly, and explain this by drawing a connection to generative score matching (Song & Ermon, 2019; 2020).

One subtlety is that L_{simple} provides no learning signal for $\Sigma_\theta(x_t, t)$. This is irrelevant, however, since Ho et al. (2020) achieved their best results by fixing the variance to $\sigma_t^2\mathbf{I}$ rather than learning it. They found that they achieve similar sample quality using either $\sigma_t^2 = \beta_t$ or $\sigma_t^2 = \tilde{\beta}_t$, which are the upper and lower bounds on the variance given by $q(x_0)$ being either isotropic Gaussian noise or a delta function, respectively.

3. Improving the Log-likelihood

While Ho et al. (2020) found that DDPMs can generate high-fidelity samples according to FID (Heusel et al., 2017) and Inception Score (Salimans et al., 2016), they were unable to achieve competitive log-likelihoods with these models. Log-likelihood is a widely used metric in generative modeling, and it is generally believed that optimizing log-likelihood forces generative models to capture all of the modes of the data distribution (Razavi et al., 2019). Additionally, recent work (Henighan et al., 2020) has shown that small improvements in log-likelihood can have a dramatic impact on sample quality and learnt feature representations. Thus, it is important to explore why DDPMs seem to perform poorly on this metric, since this may suggest a fundamental shortcoming such as bad mode coverage. This section explores several modifications to the algorithm described in Section 2 that, when combined, allow DDPMs to achieve much better log-likelihoods on image datasets, suggesting that these models enjoy the same benefits as other likelihood-based generative models.

To study the effects of different modifications, we train fixed model architectures with fixed hyperparameters on the ImageNet 64 $\times$ 64 (van den Oord et al., 2016b) and CIFAR-10 (Krizhevsky, 2009) datasets. While CIFAR-10 has seen more usage for this class of models, we chose to study ImageNet 64 $\times$ 64 as well because it provides a good trade-off between diversity and resolution, allowing us to train models quickly without worrying about overfitting. Additionally, ImageNet 64×64 has been studied extensively in the context of generative modeling (van den Oord et al., 2016c; Menick & Kalchbrenner, 2018; Child et al., 2019; Roy et al., 2020), allowing us to compare DDPMs directly to many other generative models.

The setup from Ho et al. (2020) (optimizing L_{simple} while setting $\sigma_t^2 = \beta_t$ and $T = 1000$) achieves a log-likelihood of 3.99 (bits/dim) on ImageNet 64 $\times$ 64 after 200K training iterations. We found in early experiments that we could get a boost in log-likelihood by increasing T from 1000 to 4000; with this change, the log-likelihood improves to 3.77. For the remainder of this section, we use $T = 4000$, but we explore this choice in Section 4.

3.1. Learning $\Sigma_\theta(x_t, t)$

In Ho et al. (2020), the authors set $\Sigma_\theta(x_t, t) = \sigma_t^2\mathbf{I}$, where σ_t is not learned. Oddly, they found that fixing σ_t^2 to β_t yielded roughly the same sample quality as fixing it to $\tilde{\beta}_t$.

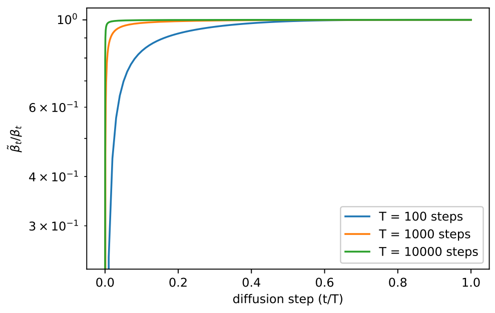


Figure 1. The ratio $\tilde{\beta}_t/\beta_t$ for every diffusion step for diffusion processes of different lengths.

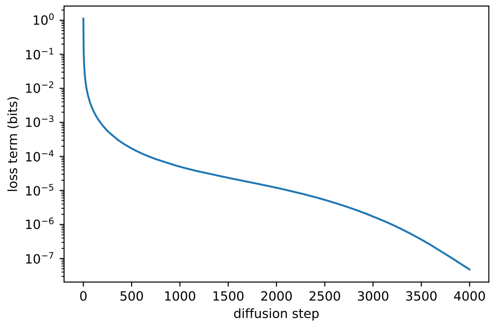


Figure 2. Terms of the VLB vs diffusion step. The first few terms contribute most to NLL.

Considering that β_t and $\tilde{\beta}_t$ represent two opposite extremes, it is reasonable to ask why this choice doesn't affect samples. One clue is given by Figure 1, which shows that β_t and $\tilde{\beta}_t$ are almost equal except near $t = 0$, i.e. where the model is dealing with imperceptible details. Furthermore, as we increase the number of diffusion steps, β_t and $\tilde{\beta}_t$ seem to remain close to one another for more of the diffusion process. This suggests that, in the limit of infinite diffusion steps, the choice of σ_t might not matter at all for sample quality. In other words, as we add more diffusion steps, the model mean $\mu_\theta(x_t, t)$ determines the distribution much more than $\Sigma_\theta(x_t, t)$.

While the above argument suggests that fixing σ_t is a reasonable choice for the sake of sample quality, it says nothing about log-likelihood. In fact, Figure 2 shows that the first few steps of the diffusion process contribute the most to the variational lower bound. Thus, it seems likely that we could improve log-likelihood by using a better choice of $\Sigma_\theta(x_t, t)$. To achieve this, we must learn $\Sigma_\theta(x_t, t)$ without the instabilities encountered by Ho et al. (2020).

Since Figure 1 shows that the reasonable range for $\Sigma_\theta(x_t, t)$ is very small, it would be hard for a neural network to predict $\Sigma_\theta(x_t, t)$ directly, even in the log domain, as observed by Ho et al. (2020). Instead, we found it better to parameterize the variance as an interpolation between β_t and $\tilde{\beta}_t$ in the

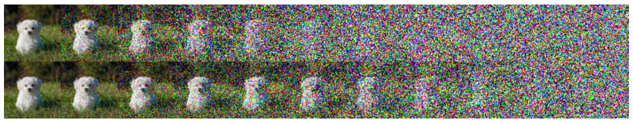


Figure 3. Latent samples from linear (top) and cosine (bottom) schedules respectively at linearly spaced values of t from 0 to T. The latents in the last quarter of the linear schedule are almost purely noise, whereas the cosine schedule adds noise more slowly

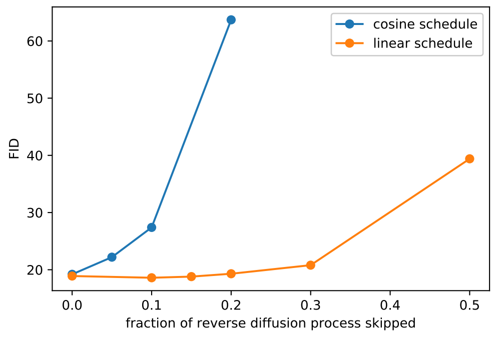


Figure 4. FID when skipping a prefix of the reverse diffusion process on ImageNet 64×64.

log domain. In particular, our model outputs a vector v containing one component per dimension, and we turn this output into variances as follows:

$$\Sigma_\theta(x_t, t) = \exp(v \log \beta_t + (1 - v) \log \tilde{\beta}_t) \tag{15}$$

We did not apply any constraints on v, theoretically allowing the model to predict variances outside of the interpolated range. However, we did not observe the network doing this in practice, suggesting that the bounds for $\Sigma_\theta(x_t, t)$ are indeed expressive enough.

Since L_{simple} doesn't depend on $\Sigma_\theta(x_t, t)$, we define a new hybrid objective:

$$L_{\text{hybrid}} = L_{\text{simple}} + \lambda L_{\text{vlb}} \tag{16}$$

For our experiments, we set $\lambda = 0.001$ to prevent L_{vlb} from overwhelming L_{simple}. Along this same line of reasoning, we also apply a stop-gradient to the $\mu_\theta(x_t, t)$ output for the L_{vlb} term. This way, L_{vlb} can guide $\Sigma_\theta(x_t, t)$ while L_{simple} is still the main source of influence over $\mu_\theta(x_t, t)$.

3.2. Improving the Noise Schedule

We found that while the linear noise schedule used in Ho et al. (2020) worked well for high resolution images, it was sub-optimal for images of resolution 64×64 and 32×32. In particular, the end of the forward noising process is too

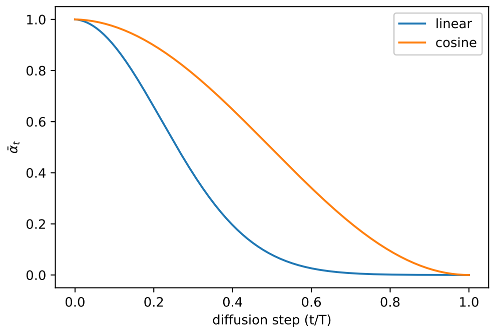


Figure 5. $\bar{\alpha}_t$ throughout diffusion in the linear schedule and our proposed cosine schedule.

noisy, and so doesn't contribute very much to sample quality. This can be seen visually in Figure 3. The result of this effect is studied in Figure 4, where we see that a model trained with the linear schedule does not get much worse (as measured by FID) when we skip up to 20% of the reverse diffusion process.

To address this problem, we construct a different noise schedule in terms of $\bar{\alpha}_t$:

$$\bar{\alpha}_t = \frac{f(t)}{f(0)}, \quad f(t) = \cos\left(\frac{t/T + s}{1 + s} \cdot \frac{\pi}{2}\right)^2 \tag{17}$$

To go from this definition to variances β_t, we note that $\beta_t = 1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}$. In practice, we clip β_t to be no larger than 0.999 to prevent singularities at the end of the diffusion process near $t = T$.

Our cosine schedule is designed to have a linear drop-off of $\bar{\alpha}_t$ in the middle of the process, while changing very little near the extremes of $t = 0$ and $t = T$ to prevent abrupt changes in noise level. Figure 5 shows how $\bar{\alpha}_t$ progresses for both schedules. We can see that the linear schedule from Ho et al. (2020) falls towards zero much faster, destroying information more quickly than necessary.

We use a small offset s to prevent β_t from being too small near $t = 0$, since we found that having tiny amounts of noise at the beginning of the process made it hard for the network to predict ϵ accurately enough. In particular, we selected s such that $\sqrt{\beta_0}$ was slightly smaller than the pixel bin size $1/127.5$, which gives $s = 0.008$. We chose to use cos^2 in particular because it is a common mathematical function with the shape we were looking for. This choice was arbitrary, and we expect that many other functions with similar shapes would work as well.

3.3. Reducing Gradient Noise

We expected to achieve the best log-likelihoods by optimizing L_{vlb} directly, rather than by optimizing L_{hybrid}. However,

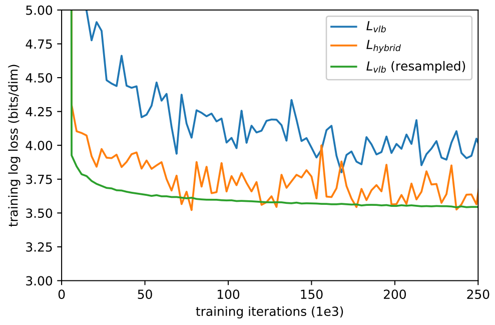


Figure 6. Learning curves comparing the log-likelihoods achieved by different objectives on ImageNet 64×64.

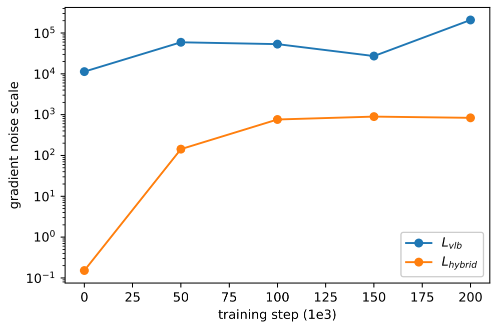


Figure 7. Gradient noise scales for the L_{vlb} and L_{hybrid} objectives on ImageNet 64×64.

we were surprised to find that L_{vlb} was actually quite difficult to optimize in practice, at least on the diverse ImageNet 64×64 dataset. Figure 6 shows the learning curves for both L_{vlb} and L_{hybrid}. Both curves are noisy, but the hybrid objective clearly achieves better log-likelihoods on the training set given the same amount of training time.

We hypothesized that the gradient of L_{vlb} was much noisier than that of L_{hybrid}. We confirmed this by evaluating the gradient noise scales (McCandlish et al., 2018) for models trained with both objectives, as shown in Figure 7. Thus, we sought out a way to reduce the variance of L_{vlb} in order to optimize directly for log-likelihood.

Noting that different terms of L_{vlb} have greatly different magnitudes (Figure 2), we hypothesized that sampling t uniformly causes unnecessary noise in the L_{vlb} objective. To address this, we employ importance sampling:

$$L_{\text{vlb}} = E_{t \sim p_t}\left[\frac{L_t}{p_t}\right], \text{ where } p_t \propto \sqrt{E[L_t^2]} \text{ and } \sum p_t = 1 \tag{18}$$

Since $E[L_t^2]$ is unknown beforehand and may change throughout training, we maintain a history of the previous 10 values for each loss term, and update this dynamically during training. At the beginning of training, we sample t uniformly until we draw 10 samples for every $t \in [0, T-1]$.

With this importance sampled objective, we are able to achieve our best log-likelihoods by optimizing L_{vlb}. This can be seen in Figure 6 as the L_{vlb} (resampled) curve. The figure also shows that the importance sampled objective is considerably less noisy than the original, uniformly sampled objective. We found that the importance sampling technique was not helpful when optimizing the less-noisy L_{hybrid} objective directly.

Table 1. Ablating schedule and objective on ImageNet 64×64.

Iters	T	Schedule	Objective	NLL	FID
200K	1K	linear	L_{simple}	3.99	32.5
200K	4K	linear	L_{simple}	3.77	31.3
200K	4K	linear	L_{hybrid}	3.66	32.2
200K	4K	cosine	L_{simple}	3.68	**27.0**
200K	4K	cosine	L_{hybrid}	3.62	28.0
200K	4K	cosine	L_{vlb}	**3.57**	56.7
1.5M	4K	cosine	L_{hybrid}	3.57	**19.2**
1.5M	4K	cosine	L_{vlb}	**3.53**	40.1

Table 2. Ablating schedule and objective on CIFAR-10.

Iters	T	Schedule	Objective	NLL	FID
500K	1K	linear	L_{simple}	3.73	3.29
500K	4K	linear	L_{simple}	3.37	**2.90**
500K	4K	linear	L_{hybrid}	3.26	3.07
500K	4K	cosine	L_{simple}	3.26	3.05
500K	4K	cosine	L_{hybrid}	3.17	3.19
500K	4K	cosine	L_{vlb}	**2.94**	11.47

3.4. Results and Ablations

In this section, we ablate the changes we have made to achieve better log-likelihoods. Table 1 summarizes the results of our ablations on ImageNet 64×64, and Table 2 shows them for CIFAR-10. We also trained our best ImageNet 64×64 models for 1.5M iterations, and report these results as well. L_{vlb} and L_{hybrid} were trained with learned sigmas using the parameterization from Section 3.1. For L_{vlb}, we used the resampling scheme from Section 3.3.

Based on our ablations, using L_{hybrid} and our cosine schedule improves log-likelihood while keeping similar FID as the baseline from Ho et al. (2020). Optimizing L_{vlb} further improves log-likelihood at the cost of a higher FID. We generally prefer to use L_{hybrid} over L_{vlb} as it gives a boost in likelihood without sacrificing sample quality.

In Table 3 we compare our best likelihood models against prior work, showing that these models are competitive with the best conventional methods in terms of log-likelihood.

Table 3. Comparison of DDPMs to other likelihood-based models on CIFAR-10 and Unconditional ImageNet 64×64. NLL is reported in bits/dim. On ImageNet 64×64, our model is competitive with the best convolutional models, but is worse than fully transformer-based architectures.

Model	ImageNet	CIFAR
Glow (Kingma & Dhariwal, 2018)	3.81	3.35
Flow++ (Ho et al., 2019)	3.69	3.08
PixelCNN (van den Oord et al., 2016c)	3.57	3.14
SPN (Menick & Kalchbrenner, 2018)	3.52	-
NVAE (Vahdat & Kautz, 2020)	-	2.91
Very Deep VAE (Child, 2020)	3.52	2.87
PixelSNAIL (Chen et al., 2018)	3.52	2.85
Image Transformer (Parmar et al., 2018)	3.48	2.90
Sparse Transformer (Child et al., 2019)	3.44	**2.80**
Routing Transformer (Roy et al., 2020)	**3.43**	-
DDPM (Ho et al., 2020)	3.77	3.70
DDPM (cont flow) (Song et al., 2020b)	-	2.99
Improved DDPM (ours)	**3.53**	**2.94**

4. Improving Sampling Speed

All of our models were trained with 4000 diffusion steps, and thus producing a single sample takes several minutes on a modern GPU. In this section, we explore how performance scales if we reduce the steps used during sampling, and find that our pre-trained L_{hybrid} models can produce high-quality samples with many fewer diffusion steps than they were trained with (without any fine-tuning). Reducing the steps in this way makes it possible to sample from our models in a number of seconds rather than minutes, and greatly improves the practical applicability of image DDPMs.

For a model trained with T diffusion steps, we would typically sample using the same sequence of t values $(1, 2, ..., T)$ as used during training. However, it is also possible to sample using an arbitrary subsequence S of t values. Given the training noise schedule $\bar{\alpha}_t$, for a given sequence S we can obtain the sampling noise schedule $\bar{\alpha}_{S_t}$, which can be then used to obtain corresponding sampling variances

$$\beta_{S_t} = 1 - \frac{\bar{\alpha}_{S_t}}{\bar{\alpha}_{S_{t-1}}}, \quad \tilde{\beta}_{S_t} = \frac{1 - \bar{\alpha}_{S_{t-1}}}{1 - \bar{\alpha}_{S_t}} \beta_{S_t} \tag{19}$$

Now, since $\Sigma_\theta(x_{S_t}, S_t)$ is parameterized as a range between β_{S_t} and $\tilde{\beta}_{S_t}$, it will automatically be rescaled for the shorter diffusion process. We can thus compute $p(x_{S_{t-1}}|x_{S_t})$ as $\mathcal{N}(\mu_\theta(x_{S_t}, S_t), \Sigma_\theta(x_{S_t}, S_t))$.

To reduce the number of sampling steps from T to K, we use K evenly spaced real numbers between 1 and T (inclusive), and then round each resulting number to the nearest integer. In Figure 8, we evaluate FIDs for an L_{hybrid} model and an L_{simple} model that were trained with 4000 diffusion steps, using 25, 50, 100, 200, 400, 1000, and 4000 sampling steps. We do this for both a fully-trained checkpoint, and a checkpoint mid-way through training. For CIFAR-10 we used 200K and 500K training iterations, and for ImageNet-64 we used 500K and 1500K training iterations. We find that the L_{simple} models with fixed sigmas (with both the larger $\sigma_t^2 = \beta_t$ and the smaller $\sigma_t^2 = \tilde{\beta}_t$) suffer much more in sample quality when using a reduced number of sampling steps, whereas our L_{hybrid} model with learnt sigmas maintains high sample quality. With this model, 100 sampling steps is sufficient to achieve near-optimal FIDs for our fully trained models.

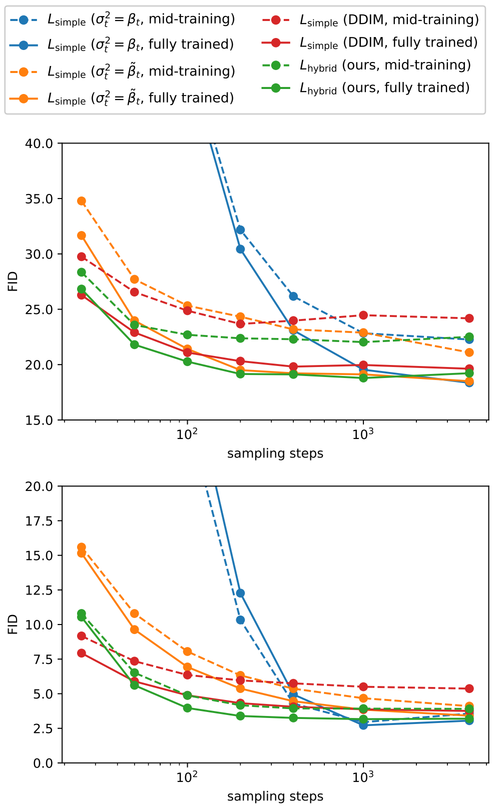


Figure 8. FID versus number of sampling steps, for models trained on ImageNet 64×64 (top) and CIFAR-10 (bottom). All models were trained with 4000 diffusion steps.

Parallel to our work, Song et al. (2020a) propose a fast sampling algorithm for DDPMs by producing a new implicit model that has the same marginal noise distributions, but deterministically maps noise to images. We include their algorithm, DDIM, in Figure 8, finding that DDIM produces better samples with fewer than 50 sampling steps, but worse samples when using 50 or more steps. Interestingly, DDIM performs worse at the start of training, but closes the gap to other samplers as training continues. We found that our striding technique drastically reduced performance of DDIM, so our DDIM results instead use the constant strid-

Table 4. Sample quality comparison on class-conditional ImageNet 64×64. Precision and recall (Kynkäänniemi et al., 2019) are measured using Inception-V3 features and $K = 5$. We trained BigGAN-deep for 125K iterations, and did not use truncation for sampling to maximize recall for the GAN.

Model	FID	Prec.	Recall
BigGAN-deep (Brock et al., 2018)	4.06	**0.86**	0.59
Improved Diffusion (small)	6.92	0.77	**0.72**
Improved Diffusion (large)	**2.92**	0.82	**0.71**

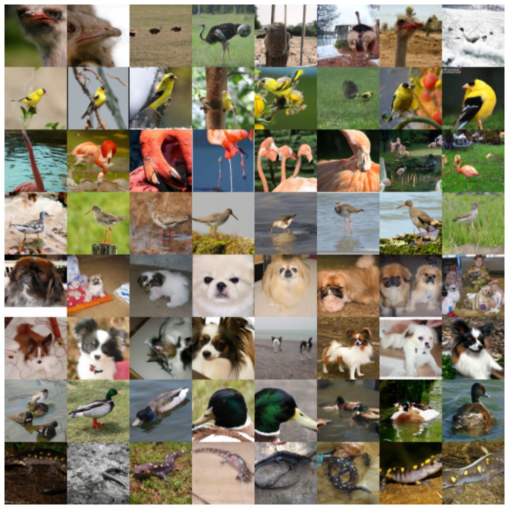


Figure 9. Class-conditional ImageNet 64×64 samples generated using 250 sampling steps from L_{hybrid} model (FID 2.92). The classes are 9: ostrich, 11: goldfinch, 130: flamingo, 141: redshank, 154: pekinese, 157: papillon, 97: drake and 28: spotted salamander. We see that there is a high diversity in each class, suggesting good coverage of the target distribution

ing[1] from Song et al. (2020a), wherein the final timestep is $T - T/K + 1$ rather than T. The other samplers performed slightly better with our striding.

5. Comparison to GANs

While likelihood is a good proxy for mode-coverage, it is difficult to compare to GANs with this metric. Instead, we turn to precision and recall (Kynkäänniemi et al., 2019). Since it is common in the GAN literature to train class-conditional models, we do the same for this experiment. To make our models class-conditional, we inject class information through the same pathway as the timestep t. In particular, we add a class embedding v_i to the timestep embedding e_t, and pass this embedding to residual blocks throughout the model. We train using the L_{hybrid} objective and use 250 sampling steps. We train two models: a "small" model with 100M parameters for 1.7M training steps, and a larger model with 270 million parameters for 250K iterations. We train one BigGAN-deep model with 100M parameters across the generator and discriminator.

When computing metrics for this task, we generated 50K samples (rather than 10K) to be directly comparable to other works.[2] This is the only ImageNet 64×64 FID we report that was computed using 50K samples. For FID, the reference distribution features were computed over the full training set, following (Brock et al., 2018).

Figure 9 shows our samples from the larger model, and Table 4 summarizes our results. We find that BigGAN-deep outperforms our smaller model in terms of FID, but struggles in terms of recall. This suggests that diffusion models are better at covering the modes of the distribution than comparable GANs.

6. Scaling Model Size

In the previous sections, we showed algorithmic changes that improved log-likelihood and FID without changing the amount of training compute. However, a trend in modern machine learning is that larger models and more training time tend to improve model performance (Kaplan et al., 2020; Chen et al., 2020a; Brown et al., 2020). Given this observation, we investigate how FID and NLL scale as a function of training compute. Our results, while preliminary, suggest that DDPMs improve in a predictable way as training compute increases.

To measure how performance scales with training compute, we train four different models on ImageNet 64×64 with the L_{hybrid} objective described in Section 3.1. To change model capacity, we apply a depth multiplier across all layers, such that the first layer has either 64, 96, 128, or 192 channels. Note that our previous experiments used 128 channels in the first layer. Since the depth of each layer affects the scale of the initial weights, we scale the Adam (Kingma & Ba, 2014) learning rate for each model by $1/\sqrt{\text{channel multiplier}}$, such that the 128 channel model has a learning rate of 0.0001 (as in our other experiments).

Figure 10 shows how FID and NLL improve relative to theoretical training compute.[3] The FID curve looks approximately linear on a log-log plot, suggesting that FID scales according to a power law (plotted as the black dashed line). The NLL curve does not fit a power law as cleanly, suggesting that validation NLL scales in a less-favorable manner

[1]We additionally tried the quadratic stride from Song et al. (2020a), but found that it hurt sample quality when combined with our cosine schedule.

[2]We found that using more samples led to a decrease in estimated FID of roughly 2 points.

[3]The x-axis assumes full hardware utilization

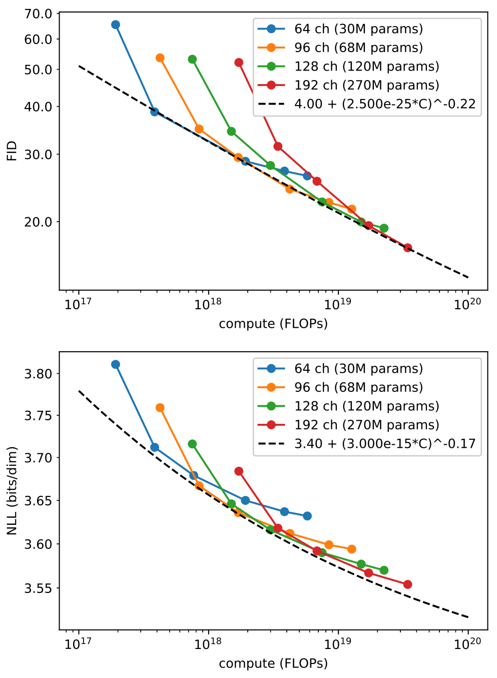


Figure 10. FID and validation NLL throughout training on ImageNet 64 × 64 for different model sizes. The constant for the FID trend line was approximated using the FID of in-distribution data. For the NLL trend line, the constant was approximated by rounding down the current state-of-the-art NLL (Roy et al., 2020) on this dataset.

than FID. This could be caused by a variety of factors, such as 1) an unexpectedly high irreducible loss (Henighan et al., 2020) for this type of diffusion model, or 2) the model overfitting to the training distribution. We also note that these models do not achieve optimal log-likelihoods in general because they were trained with our L_{hybrid} objective and not directly with L_{vlb} to keep both good log-likelihoods and sample quality.

7. Related Work

Chen et al. (2020b) and Kong et al. (2020) are two recent works that use DDPMs to produce high fidelity audio conditioned on mel-spectrograms. Concurrent to our work, Chen et al. (2020b) use a combination of improved schedule and L_1 loss to allow sampling with fewer steps with very little reduction in sample quality. However, compared to our unconditional image generation task, their generative task has a strong input conditioning signal provided by the mel-spectrograms, and we hypothesize that this makes it easier to sample with fewer diffusion steps.

Jolicoeur-Martineau et al. (2020) explored score matching in the image domain, and constructed an adversarial training objective to produce better x_0 predictions. However, they found that choosing a better network architecture removed the need for this adversarial objective, suggesting that the adversarial objective is not necessary for powerful generative modeling.

Parallel to our work, Song et al. (2020a) and Song et al. (2020b) propose fast sampling algorithms for models trained with the DDPM objective by leveraging different sampling processes. Song et al. (2020a) does this by deriving an implicit generative model that has the same marginal noise distributions as DDPMs while deterministically mapping noise to images. Song et al. (2020b) model the diffusion process as the discretization of a continuous SDE, and observe that there exists an ODE that corresponds to sampling from the reverse SDE. By varying the numerical precision of an ODE solver, they can sample with fewer function evaluations. However, they note that this technique obtains worse samples than ancestral sampling when used directly, and only achieves better FID when combined with Langevin corrector steps. This in turn requires hand-tuning of a signal-to-noise ratio for the Langevin steps. Our method allows fast sampling directly from the ancestral process, which removes the need for extra hyperparameters.

Also in parallel, Gao et al. (2020) develops a diffusion model with reverse diffusion steps modeled by an energy-based model. A potential implication of this approach is that fewer diffusion steps should be needed to achieve good samples.

8. Conclusion

We have shown that, with a few modifications, DDPMs can sample much faster and achieve better log-likelihoods with little impact on sample quality. The likelihood is improved by learning Σ_θ using our parameterization and L_{hybrid} objective. This brings the likelihood of these models much closer to other likelihood-based models. We surprisingly discover that this change also allows sampling from these models with many fewer steps.

We have also found that DDPMs can match the sample quality of GANs while achieving much better mode coverage as measured by recall. Furthermore, we have investigated how DDPMs scale with the amount of available training compute, and found that more training compute trivially leads to better sample quality and log-likelihood.

The combination of these results makes DDPMs an attractive choice for generative modeling, since they combine good log-likelihoods, high-quality samples, and reasonably fast sampling with a well-grounded, stationary training objective that scales easily with training compute. These results indicate that DDPMs are a promising direction for future research.

References

Brock, A., Donahue, J., and Simonyan, K. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020.

Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Dhariwal, P., Luan, D., and Sutskever, I. Generative pretraining from pixels, 2020a. URL `https://cdn.openai.com/papers/Generative_Pretraining_from_Pixels_V2.pdf`.

Chen, N., Zhang, Y., Zen, H., Weiss, R. J., Norouzi, M., and Chan, W. Wavegrad: Estimating gradients for waveform generation, 2020b.

Chen, X., Mishra, N., Rohaninejad, M., and Abbeel, P. Pixelsnail: An improved autoregressive generative model. In *International Conference on Machine Learning*, pp. 864–872. PMLR, 2018.

Child, R. Very deep vaes generalize autoregressive models and can outperform them on images. *arXiv preprint arXiv:2011.10650*, 2020.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers, 2019.

Gao, R., Song, Y., Poole, B., Wu, Y. N., and Kingma, D. P. Learning energy-based models by diffusion recovery likelihood, 2020.

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

Henighan, T., Kaplan, J., Katz, M., Chen, M., Hesse, C., Jackson, J., Jun, H., Brown, T. B., Dhariwal, P., Gray, S., Hallacy, C., Mann, B., Radford, A., Ramesh, A., Ryder, N., Ziegler, D. M., Schulman, J., Amodei, D., and McCandlish, S. Scaling laws for autoregressive generative modeling, 2020.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. 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.

Ho, J., Chen, X., Srinivas, A., Duan, Y., and Abbeel, P. Flow++: Improving flow-based generative models with variational dequantization and architecture design. *arXiv preprint arXiv:1902.00275*, 2019.

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

Hyvärinen, A. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(Apr):695–709, 2005.

Jolicoeur-Martineau, A., Piché-Taillefer, R., des Combes, R. T., and Mitliagkas, I. Adversarial score matching and improved sampling for image generation, 2020.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2014.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pp. 10215–10224, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes, 2013.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. Diffwave: A versatile diffusion model for audio synthesis, 2020.

Krizhevsky, A. Learning multiple layers of features from tiny images, 2009. URL `http://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

Kynkäänniemi, T., Karras, T., Laine, S., Lehtinen, J., and Aila, T. Improved precision and recall metric for assessing generative models, 2019.

McCandlish, S., Kaplan, J., Amodei, D., and Team, O. D. An empirical model of large-batch training, 2018.

Menick, J. and Kalchbrenner, N. Generating high fidelity images with subscale pixel networks and multidimensional upscaling, 2018.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, Ł., Shazeer, N., Ku, A., and Tran, D. Image transformer. *arXiv preprint arXiv:1802.05751*, 2018.

Ravuri, S. and Vinyals, O. Classification accuracy score for conditional generative models. *arXiv preprint arXiv:1905.10887*, 2019.

Razavi, A., van den Oord, A., and Vinyals, O. Generating diverse high-fidelity images with vq-vae-2, 2019.

Roy, A., Saffar, M., Vaswani, A., and Grangier, D. Efficient content-based sparse attention with routing transformers, 2020.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans, 2016.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications, 2017.

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

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models, 2020a.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pp. 11918–11930, 2019.

Song, Y. and Ermon, S. Improved techniques for training score-based generative models. *arXiv preprint arXiv:2006.09011*, 2020.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations, 2020b.

Vahdat, A. and Kautz, J. Nvae: A deep hierarchical variational autoencoder. *arXiv preprint arXiv:2007.03898*, 2020.

van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks, 2016a.

van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. Conditional image generation with pixelcnn decoders, 2016b. URL http://image-net.org/small/download.php.

van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. Conditional image generation with pixelcnn decoders, 2016c.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2017.

Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., and Xiao, J. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop, 2015.

A. Hyperparameters

For all of our experiments, we use a UNet model architecture[4] similar to that used by Ho et al. (2020). We changed the attention layers to use multi-head attention (Vaswani et al., 2017), and opted to use four attention heads rather than one (while keeping the same total number of channels). We employ attention not only at the 16x16 resolution, but also at the 8x8 resolution. Additionally, we changed the way the model conditions on t. In particular, instead of computing a conditioning vector v and injecting it into hidden state h as GroupNorm$(h + v)$, we compute conditioning vectors w and b and inject them into the hidden state as GroupNorm$(h)(w + 1) + b$. We found in preliminary experiments on ImageNet 64×64 that these modifications slightly improved FID.

For ImageNet 64×64 the architecture we use is described as follows. The downsampling stack performs four steps of downsampling, each with three residual blocks (He et al., 2015). The upsampling stack is setup as a mirror image of the downsampling stack. From highest to lowest resolution, the UNet stages use $[C, 2C, 3C, 4C]$ channels, respectively. In our ImageNet 64×64 ablations, we set $C = 128$, but we experiment with scaling C in a later section. We estimate that, with $C = 128$, our model is comprised of 120M parameters and requires roughly 39 billion FLOPs in the forward pass.

For our CIFAR-10 experiments, we use a smaller model with three resblocks per downsampling stage and layer widths $[C, 2C, 2C, 2C]$ with $C = 128$. We swept over dropout values $\{0.1, 0.2, 0.3\}$ and found that 0.1 worked best for the linear schedule while 0.3 worked best for our cosine schedule. We expand upon this in Section F.

We use Adam (Kingma & Ba, 2014) for all of our experiments. For most experiments, we use a batch size of 128, a learning rate of 10^{-4}, and an exponential moving average (EMA) over model parameters with a rate of 0.9999. For our scaling experiments, we vary the learning rate to accomodate for different model sizes. For our larger class-conditional ImageNet 64×64 experiments, we scaled up the batch size to 2048 for faster training on more GPUs.

When using the linear noise schedule from Ho et al. (2020), we linearly interpolate from $\beta_1 = 0.0001/4$ to $\beta_{4000} = 0.02/4$ to preserve the shape of $\bar{\alpha}_t$ for the $T = 4000$ schedule.

When computing FID we produce 50K samples from our models, except for unconditional ImageNet 64×64 where we produce 10K samples. Using only 10K samples biases the FID to be higher, but requires much less compute for sampling and helps do large ablations. Since we mainly use FID for relative comparisons on unconditional ImageNet 64×64, this bias is acceptable. For computing the reference distribution statistics we follow prior work (Ho et al., 2020; Brock et al., 2018) and use the full training set for CIFAR-10 and ImageNet, and 50K training samples for LSUN. Note that unconditional ImageNet 64×64 models are trained and evaluated using the official ImageNet-64 dataset (van den Oord et al., 2016a), whereas for class conditional ImageNet 64×64 and 256×256 we center crop and area downsample images (Brock et al., 2018).

[4]In initial experiments, we found that a ResNet-style architecture with no downsampling achieved better log-likelihoods but worse FIDs than the UNet architecture.

B. Fast Sampling on LSUN 256×256

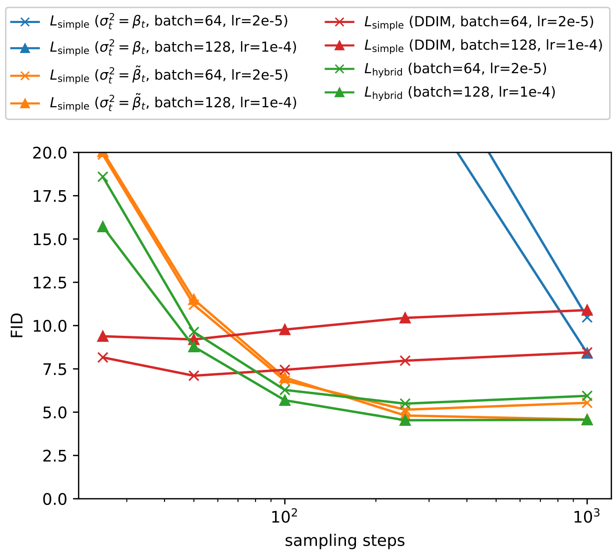


Figure 11. FID vs. number of sampling steps from an LSUN 256×256 bedroom model.

To test the effectiveness of our L_{hybrid} models on a high-resolution domain, we trained both L_{hybrid} and L_{simple} models on the LSUN bedroom (Yu et al., 2015) dataset. We train two models: one with batch size 64 and learning rate 2×10^{-5} as in Ho et al. (2020), and another with a larger batch size 128 and learning rate 10^{-4}. All models were trained with 153.6M examples, which is 2.4M training iterations with batch size 64.

Our results are displayed in Figure 11. We find that DDIM outperforms our L_{hybrid} model when using fewer than 50 diffusion steps, while our L_{hybrid} model outperforms DDIM with more than 50 diffusion steps. Interestingly, we note that DDIM benefits from a smaller learning rate and batch size, whereas our method is able to take advantage of a larger learning rate and batch size.

C. Sample Quality on ImageNet 256×256

We trained two models on class conditional ImageNet 256×256. The first is a usual diffusion model that directly models the 256×256 images. The second model reduces compute by chaining a pretrained 64×64 model $p(x_{64}|y)$ with another upsampling diffusion model $p(x_{256}|x_{64}, y)$ to upsample images to 256×256. For the upsampling model, the downsampled image x_{64} is passed as extra conditioning input to the UNet. This is similar to VQ-VAE-2 (Razavi et al., 2019), which uses two stages of priors at different latent resolutions to more efficiently learn global and local features. The linear schedule worked better for 256×256 images, so we used that for these results. Table 5 summarizes our results. For VQ-VAE-2, we use the FIDs reported in (Ravuri & Vinyals, 2019). Diffusion models still obtain the best FIDs for a likelihood-based model, and close the gap to GANs considerably.

MODEL	**FID**
VQ-VAE-2 ((Razavi et al., 2019), two-stage)	38.1
Improved Diffusion (ours, single-stage)	31.5
Improved Diffusion (ours, two-stage)	**12.3**
BigGAN (Brock et al., 2018)	7.7
BigGAN-deep (Brock et al., 2018)	**7.0**

Table 5. Sample quality comparison on class conditional ImageNet 256×256. BigGAN FIDs are reported for the truncation that results in the best FID.

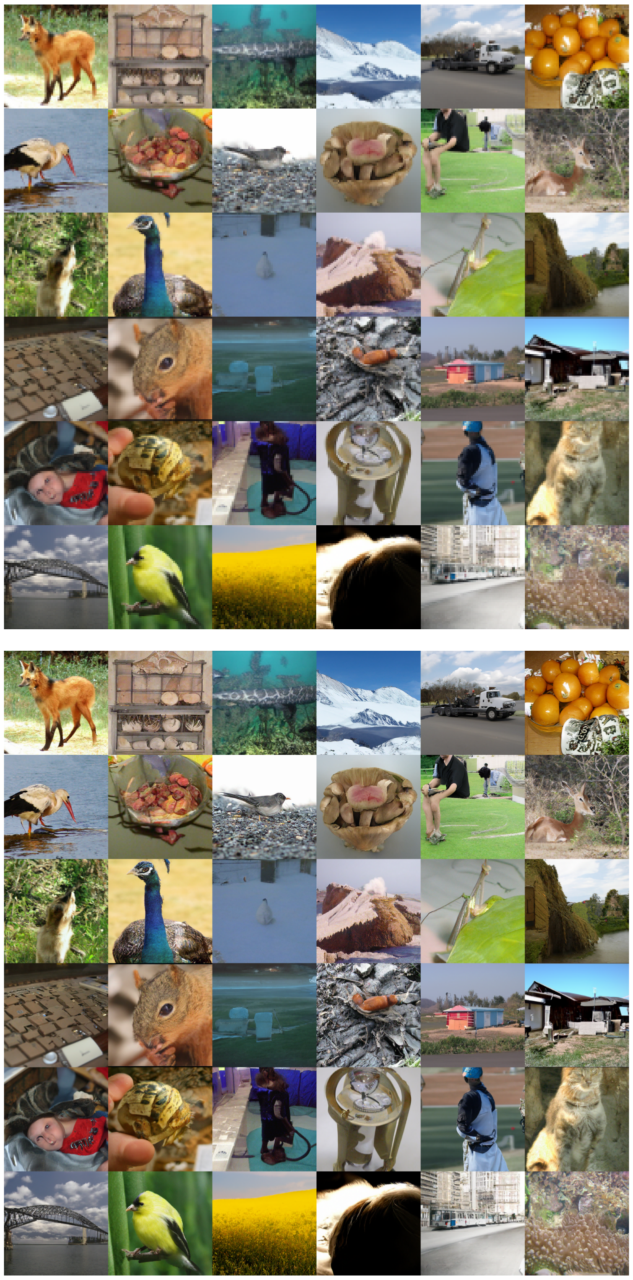


Figure 12. Random samples from two-stage class conditional ImageNet 256×256 model. On top are random samples from the 64×64 model (FID 2.92), whereas on bottom are the results after upsampling them to 256×256 (FID 12.3). Each model uses 250 sampling steps.

D. Combining L_{hybrid} and L_{vlb} Models

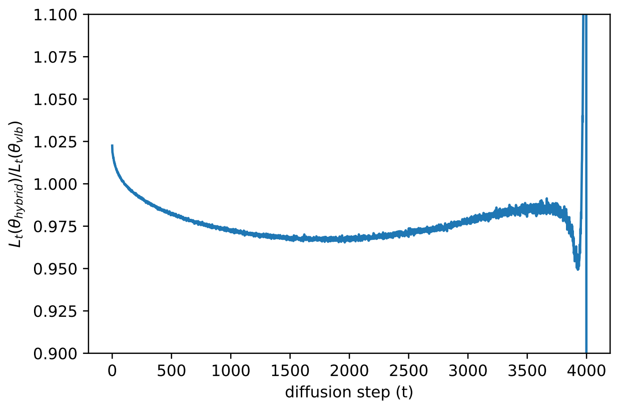


Figure 13. The ratio between VLB terms for each diffusion step of θ_{hybrid} and θ_{vlb}. Values less than 1.0 indicate that θ_{hybrid} is "better" than θ_{vlb} for that timestep of the diffusion process.

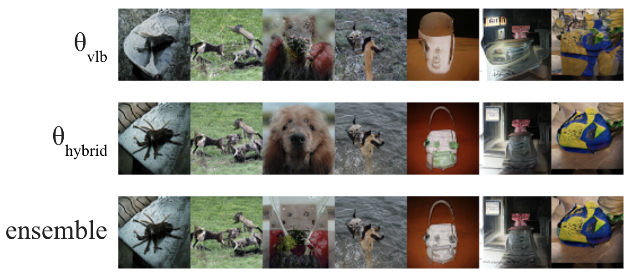


Figure 14. Samples from θ_{vlb} and θ_{hybrid}, as well as an ensemble produced by using θ_{vlb} for the first and last 100 diffusion steps. For these samples, the seed was fixed, allowing a direct comparison between models.

To understand the trade-off between L_{hybrid} and L_{vlb}, we show in Figure 13 that the model resulting from L_{vlb} (referred to as θ_{vlb}) is better at the start and end of the diffusion process, while the model resulting from L_{hybrid} (referred to as θ_{hybrid}) is better throughout the middle of the diffusion process. This suggests that θ_{vlb} is focusing more on imperceptible details, hence the lower sample quality.

Given the above observation, we performed an experiment on ImageNet 64×64 to combine the two models by constructing an ensemble that uses θ_{hybrid} for $t \in [100, T-100)$ and θ_{vlb} elsewhere. We found that this model achieved an FID of **19.9** and an NLL of **3.52 bits/dim**. This is only slightly worse than θ_{hybrid} in terms of FID, while being better than both models in terms of NLL.

E. Log-likelihood with Fewer Diffusion Steps

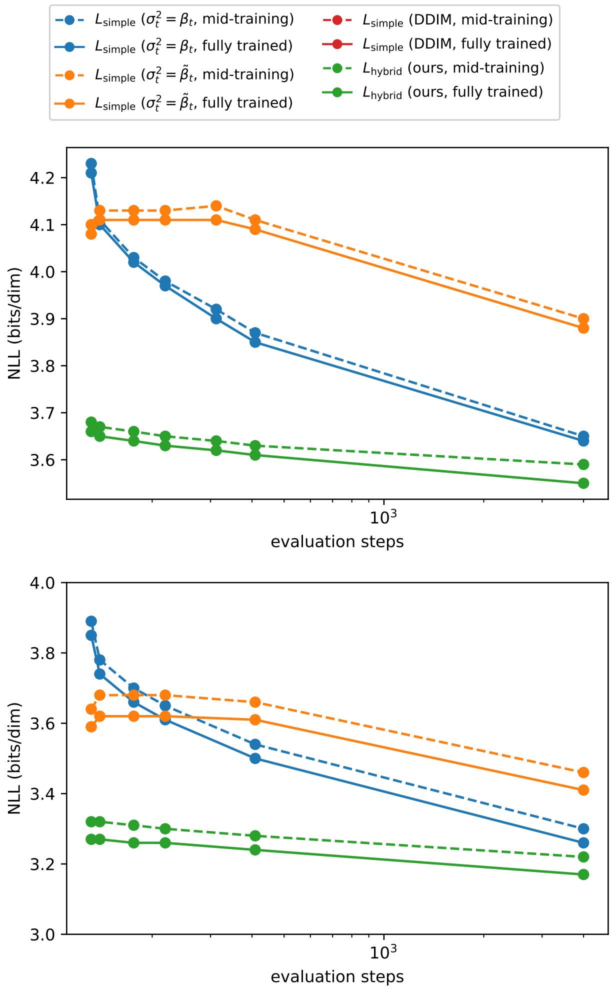
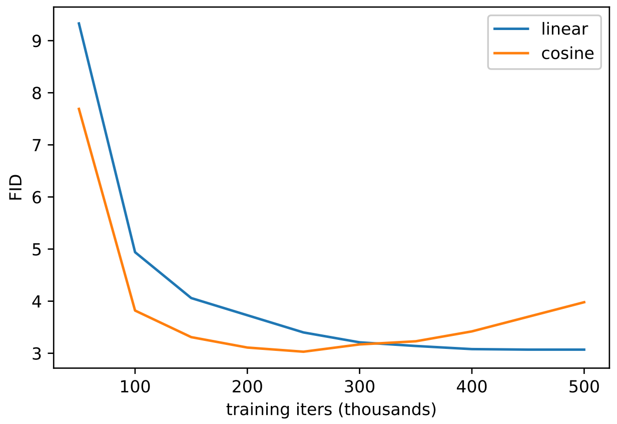


Figure 15. NLL versus number of evaluation steps, for models trained on ImageNet 64×64 (top) and CIFAR-10 (bottom). All models were trained with 4000 diffusion steps.

Figures 15 plots negative log-likelihood as a function of number of sampling steps for both ImageNet 64×64 and CIFAR-10. In initial experiments, we found that although constant striding did not significantly affect FID, it drastically reduced log-likelihood. To address this, we use a strided subset of timesteps as for FID, but we also include every t from 1 to T/K. This requires T/K extra evaluation steps, but greatly improves log-likelihood compared to the uniformly strided schedule. We did not attempt to calculate NLL using DDIM, since Song et al. (2020a) does not present NLL results or a simple way of estimating likelihood under DDIM.

F. Overfitting on CIFAR-10



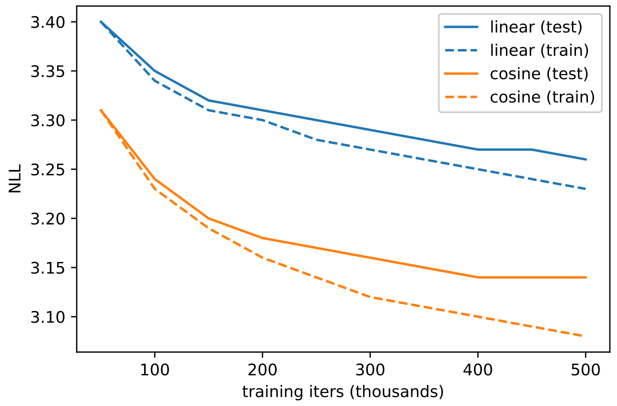


Figure 16. FID (top) and NLL (bottom) over the course of training for two CIFAR-10 models, both with dropout 0.1. The model trained with the linear schedule learns more slowly, but does not overfit as quickly. When too much overfitting occurs, we observed overfitting artifacts similar to those from Salimans et al. (2017), which is reflected by increasing FID.

On CIFAR-10, we noticed that all models overfit, but tended to reach similar optimal FID at some point during training. Holding dropout constant, we found that models trained with our cosine schedule tended to reach optimal performance (and then overfit) more quickly than those trained with the linear schedule (Figure 16). In our experiments, we corrected for this difference by using more dropout for our cosine models than the linear models. We suspect that the overfitting from the cosine schedule is either due to 1) less noise in the cosine schedule providing less regularization, or 2) the cosine schedule making optimization, and thus overfitting, easier.

G. Early stopping for FID

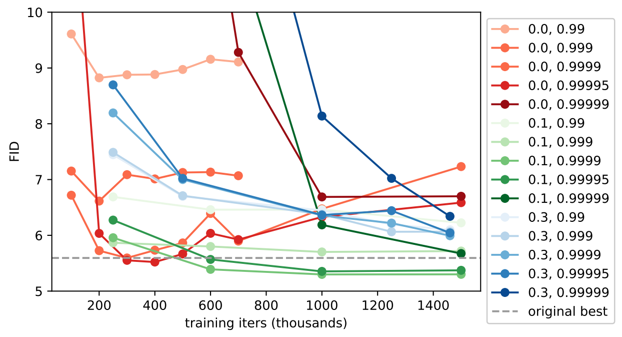


Figure 17. A sweep of dropout and EMA hyperparameters on class conditional ImageNet-64.

Like on CIFAR-10, we surprisingly observed overfitting on class-conditional ImageNet 64×64, despite it being a much larger and more diverse dataset. The main observable result of this overfitting was that FID started becoming worse over the course of training. We initially tried a sweep (Figure 17) over the EMA hyperparameter to make sure it was well tuned, and found that 0.9999 and 0.99995 worked best. We then tried runs with dropout 0.1 and 0.3, and found that models with a small amount of dropout improved the best attainable FID but took longer to get to the same performance and still eventually overfit. We concluded that the best way to train, given what we know, is to early stop and instead increase model size if we want to use additional training compute.

H. Samples with Varying Steps and Objectives

Figures 18 through 23 show unconditional ImageNet 64×64 samples as we reduce number of sampling steps for an L_{hybrid} model with $4K$ diffusion steps trained for 1.5M training iterations.

Figures 24 through 29 show unconditional CIFAR-10 samples as we reduce number of sampling steps for an L_{hybrid} model with $4K$ diffusion steps trained for 500K training iterations.

Figures 30 and 31 highlight the difference in sample quality between models trained with L_{hybrid} and L_{vlb}.

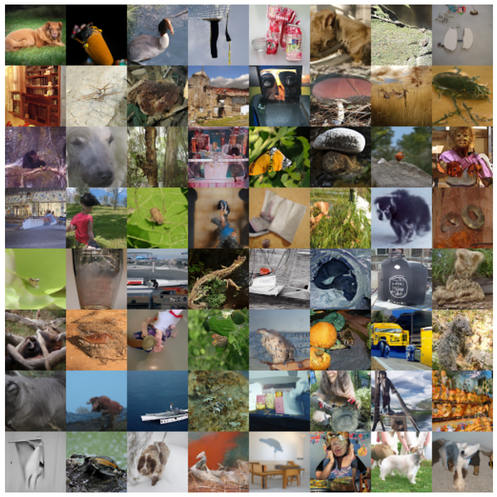


Figure 18. 50 sampling steps on unconditional ImageNet 64 × 64

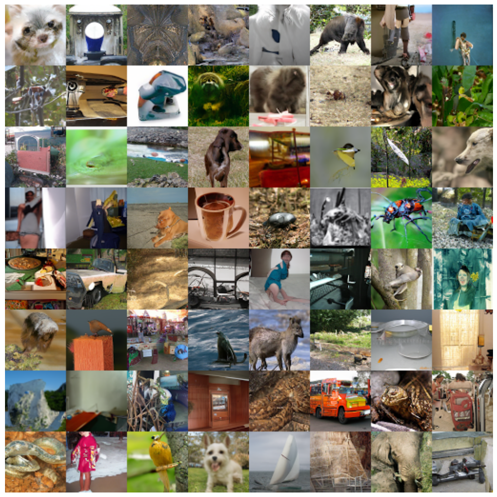


Figure 19. 100 sampling steps on unconditional ImageNet 64 × 64

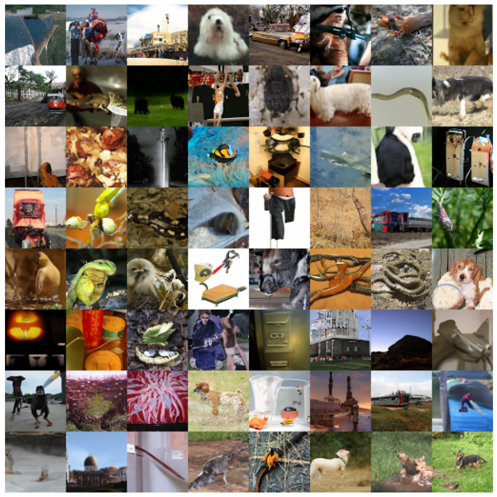


Figure 20. 200 sampling steps on unconditional ImageNet 64 × 64

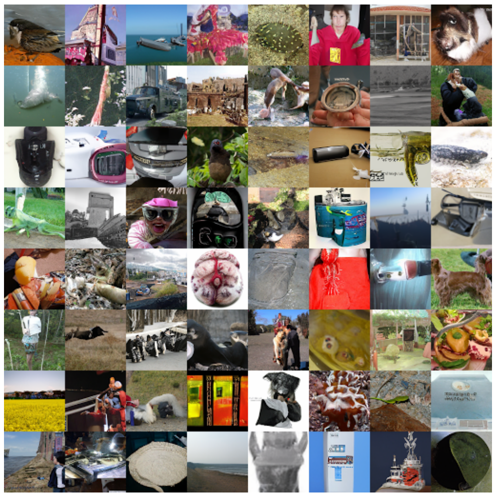


Figure 21. 400 sampling steps on unconditional ImageNet 64 × 64

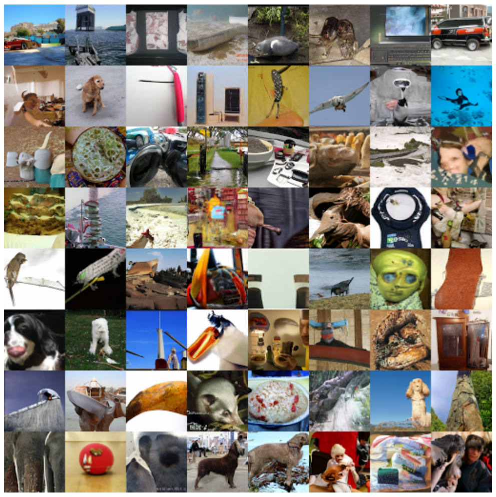


Figure 22. 1000 sampling steps on unconditional ImageNet 64×64

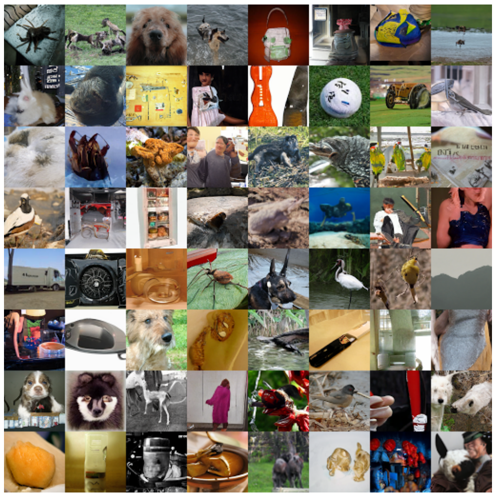


Figure 23. 4K sampling steps on unconditional ImageNet 64 × 64.

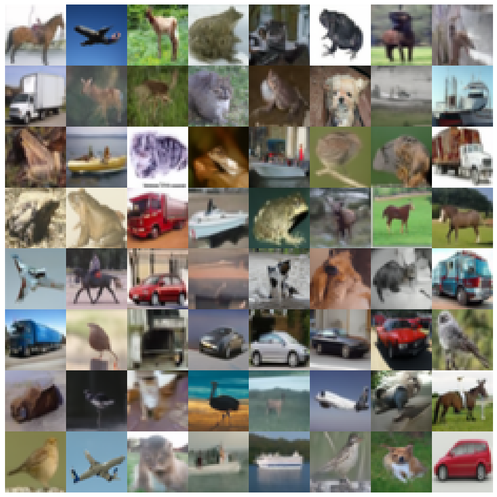


Figure 24. 50 sampling steps on unconditional CIFAR-10

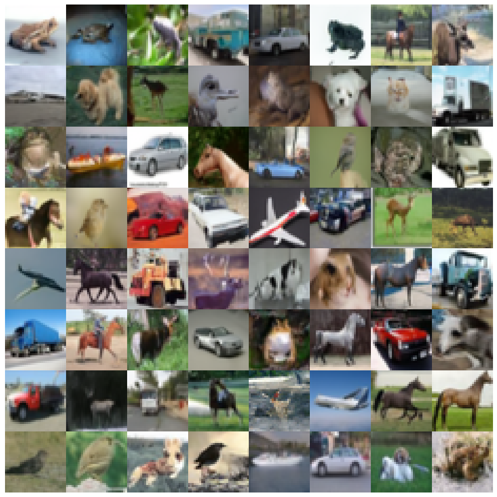


Figure 25. 100 sampling steps on unconditional CIFAR-10

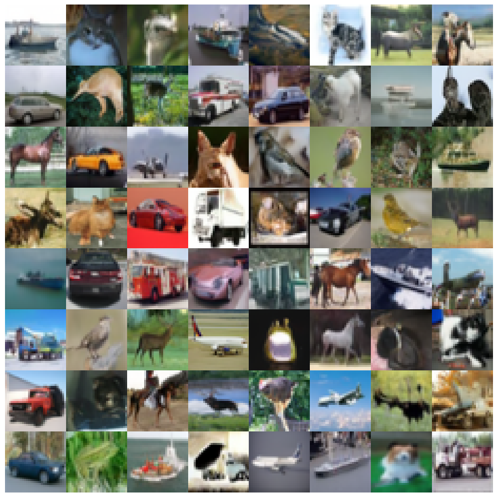


Figure 26. 200 sampling steps on unconditional CIFAR-10

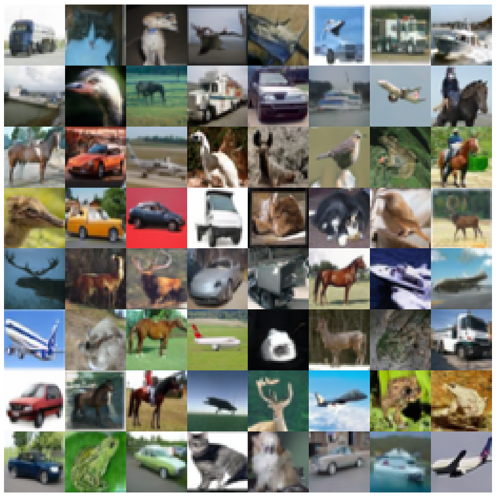


Figure 27. 400 sampling steps on unconditional CIFAR-10

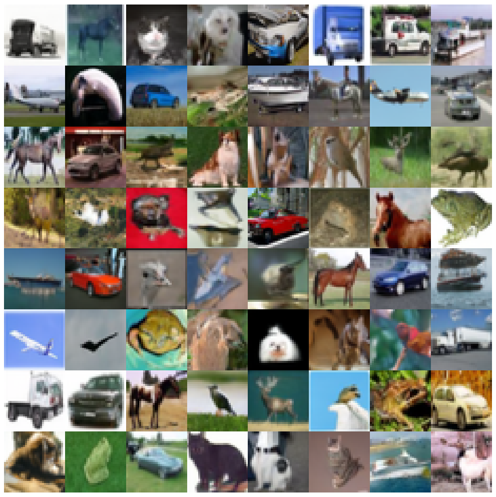


Figure 28. 1000 sampling steps on unconditional CIFAR-10

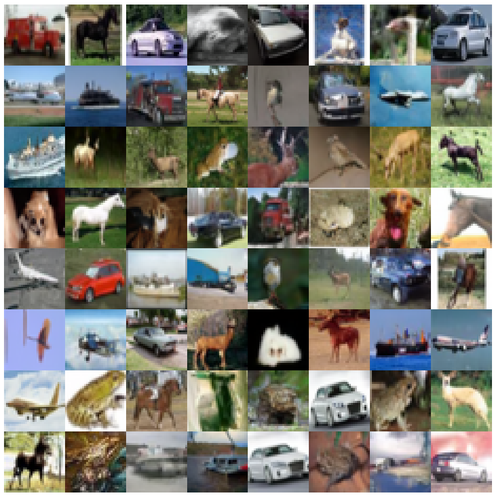


Figure 29. 4000 sampling steps on unconditional CIFAR-10

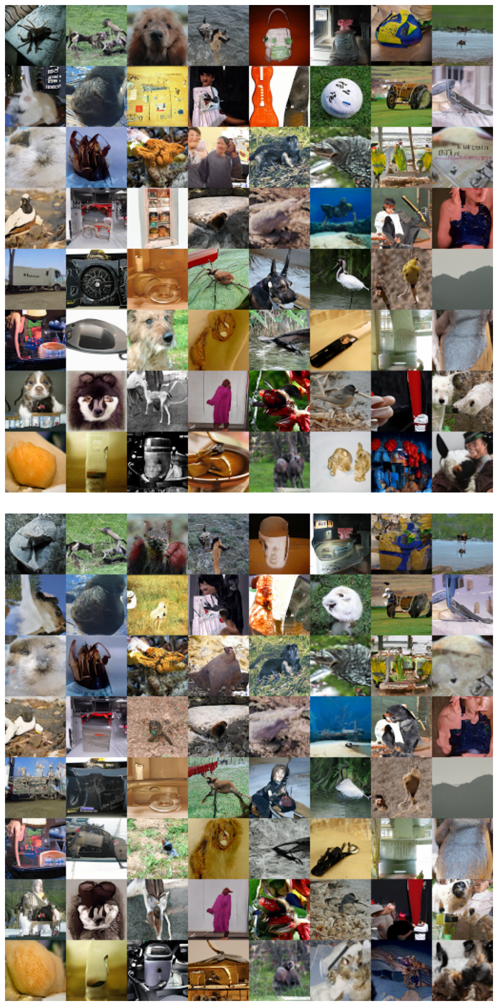


Figure 30. Unconditional ImageNet 64×64 samples generated from L_{hybrid} (top) and L_{vlb} (bottom) models using the exact same random noise. Both models were trained for 1.5M iterations.

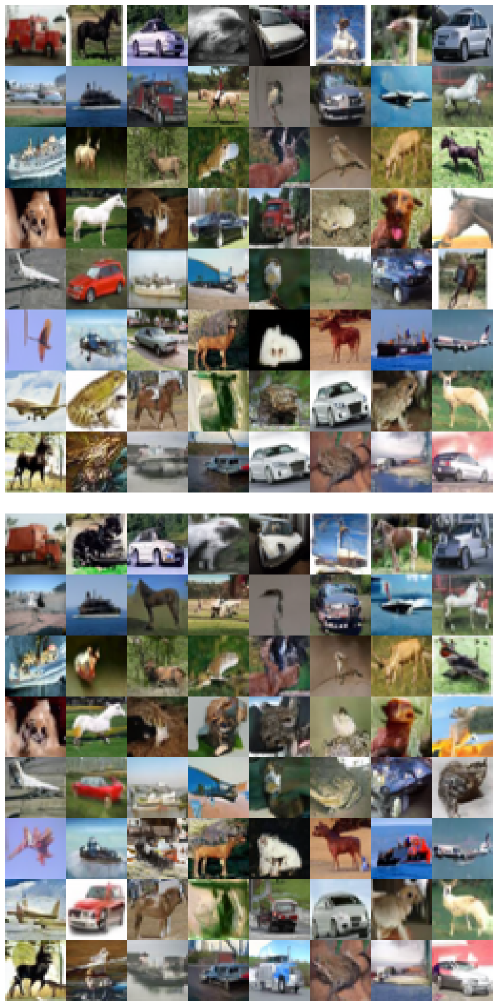


Figure 31. Unconditional CIFAR-10 samples generated from L_{hybrid} (top) and L_{vlb} (bottom) models using the exact same random noise. Both models were trained for 500K iterations.