# ADJUSTING THE INDUCTIVE BIAS OF DIFFUSION MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

It has been found empirically that diffusion-based generative models strongly benefit from weighting the score-matching objective in the training process and from redirecting trajectories in the sampling process to closer match the training distribution. Here we show that a beneficial loss weight arises naturally when the training objective is derived from first principles by enforcing detailed balance between the forward and the reverse diffusion trajectories. We find that deterministic sampling by diffusion models induces a strong bias, favoring features of some training examples while ignoring others. To correct for the strong sampling bias, we introduce an efficient and controllable rejection sampling approach. We achieve a new state-of-the-art FID of 1.42 for CIFAR-10 in a class-conditional setting.

## 1 INTRODUCTION

Diffusion models have emerged as powerful generators of images (Sohl-Dickstein et al., 2015; Ho et al., 2020; Karras et al., 2022; Song et al., 2021; Kingma et al., 2023), audio signals (Kong et al., 2021), video sequences (Ho et al., 2022), and text (Popov et al., 2021). Despite their longer training times and slower sampling rates in comparison to generative adversarial networks (GANs), they often show superior results with respect to image quality and sample distribution, at least when using FID (Heusel et al., 2018) as performance measure, which is a debatable performance score (Chong & Forsyth, 2020; Kynkäänniemi et al., 2023; Betzalel et al., 2022). Recent advances to improve the performance of diffusion-based generative models have often been based on extensive empirical optimizations, such as finding a suitable weighting of the score-matching objective as a function of the noise level (Karras et al., 2022) or forcing the denoising trajectories to more closely follow the training set examples (Kim et al., 2023). Although empirical optimization is central to the rapid advances in deep learning, a better theoretical understanding of the limitations of an approach often helps to find strategies for further improvements.

Our first contribution is to show that a loss weight for diffusion models – which results significant performance gain – arises naturally when the denoising diffusion objective is derived from first principles. The starting point for our derivation is the detailed balance condition, which ensures that the noising process and the denoising process take the same trajectory with the same probability density during training. That a loss weight was not an integral part of previously derived objectives is a consequence of matching the score instead of matching noising and denoising trajectories, where the latter implies score-matching.

Our second contribution focuses on the well-known fact of diffusion models – and perhaps all generative models – that the distribution of generated images differs significantly from the training set in feature space. As a consequence, the higher-level features for a significant fraction of training examples are almost never sampled by diffusion models (Fig. 1).

## 2 DENOISING SCORE-MATCHING OBJECTIVE SHOULD BE WEIGHTED

Diffusion-based generative models learn to follow trajectories that connect noise samples, $\boldsymbol{x} \sim p_{noise}(\boldsymbol{x})$, with data points, $\boldsymbol{y} \sim p_{data}(\boldsymbol{x})$. This is achieved by continuously adding noise to data-

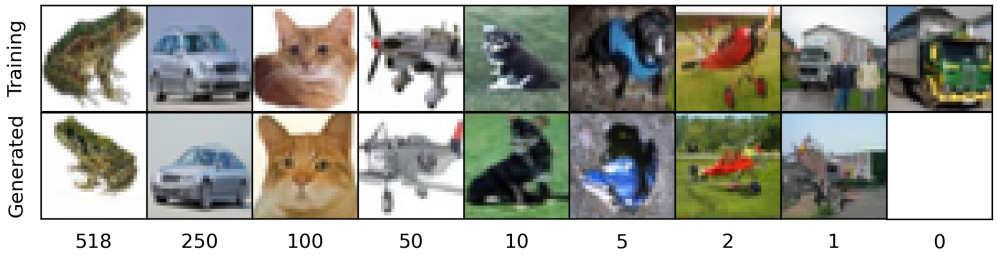



Figure 1: 50k-means clustering of 1M generated samples for CIFAR-10 in Inception feature space, with training set images as cluster centers. Examples of training set images (first row) with random images from the corresponding clusters (second row) and cluster sizes (third row). Diffusion models favor the generation of samples with simple structured backgrounds. More than 7% of the training set images have a cluster size of zero.

points, using a Wiener process, $W(t)$,

$$dX(t) = \sqrt{2D(t)}dW(t) \qquad ; \quad X(0) \sim p_{\text{data}}(\boldsymbol{y})$$

Here, $X(t), W(t)$ denote random variables[1] that are indexed by real number $t \in [0, T]$ – with $t \geq 0$ interpreted as time – and $D(t)$ is a time-dependent diffusion constant. A reverse process $X_R(t)$ can be learned from realisations of the random variable $X(t)$,

$$dX_R(t) = \mathcal{F}(X_R(t), t)dt + \sqrt{2D_R(t)}dW(t) \qquad ; \quad X_R(0) \sim p_{\text{noise}}(\boldsymbol{x})$$

with $\mathcal{F}$ a parameterized denoising flow field that drives noisy data points towards the data distribution. The corresponding conditional probability density function for sampling noisy data points is normal distributed as a consequence of the Wiener process, $X(t) \sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{y}, \sigma(t)^2) = p_X(\boldsymbol{x}, t|\boldsymbol{y}, 0)$. The noise scale $\sigma(t)$ increases over time in the forward process and is related to the diffusion constant by $D(t) = \sigma(t)d\sigma(t)/dt$. The highest noise scale $\sigma(T)$ is chosen such that $p_X(\boldsymbol{x}, T|\boldsymbol{y}, 0) \approx \mathcal{N}(\boldsymbol{x}|0, \sigma(T)^2) =: p_{\text{noise}}(\boldsymbol{x})$. Training of $\mathcal{F}$ at sufficiently high noise levels is necessary to allow the generation of new data points from $X_R(0) \sim p_{\text{noise}}(\boldsymbol{x})$ by the reverse process.

### 2.1 SCORE-MATCHING BY DETAILED BALANCE

For a given forward process with marginal $p(\boldsymbol{x}, t) = \mathbb{E}_{\boldsymbol{x}_0 \sim p_{\text{data}}}[\mathcal{N}(\boldsymbol{x}|\boldsymbol{x}_0, \sigma_t^2)]$, a unique reverse process can be found from the detailed balance condition

$$p_X(\boldsymbol{x}_t, t|\boldsymbol{x}_\tau, \tau)p(\boldsymbol{x}_\tau, \tau) = p_{X_R}(\boldsymbol{x}_\tau, \tau|\boldsymbol{x}_t, t)p(\boldsymbol{x}_t, t) \qquad \forall\, t, \tau \in [0, T], t \geq \tau$$

with $X_R(T - \tau) \sim p_{X_R}(\boldsymbol{x}_\tau, \tau|\boldsymbol{x}_t, t)$ under the initial condition $X_R(T - t) = \boldsymbol{x}_t$. The marginals at the boundaries are given by $p(\boldsymbol{x}, 0) = p_{\text{data}}(\boldsymbol{x})$ and $p(\boldsymbol{x}, T) \approx p_{\text{noise}}(\boldsymbol{x})$. Using the time discretization $t = k\Delta t$, with $k \in \{0, .., N\}$ and $\Delta t := T/N$, together with iterative application of the detailed balance condition, gives the relation

$$\begin{aligned} p_{\text{noise}}(\boldsymbol{x}_N) = p(\boldsymbol{x}_N, t_N) &= \frac{p_X(\boldsymbol{x}_N, t_N|\boldsymbol{x}_{N-1}, t_{N-1})}{p_{X_R}(\boldsymbol{x}_{N-1}, t_{N-1}|\boldsymbol{x}_N, t_N)} p(\boldsymbol{x}_{N-1}, t_{N-1}) \\ &= \prod_{k=1}^{N} \frac{p_X(\boldsymbol{x}_k, t_k|\boldsymbol{x}_{k-1}, t_{k-1})}{p_{X_R}(\boldsymbol{x}_{k-1}, t_{k-1}|\boldsymbol{x}_k, t_k)} p_{\text{data}}(\boldsymbol{x}_0) \end{aligned}$$

The continuous flow field $\mathcal{F}$ is chosen such that the probability of moving along a given trajectory under the forward process equals the probability of moving the same trajectory backwards under the reverse process. In the limit $N \to \infty$, $\mathcal{F}$ can be uniquely determined (Appendix A) from minimizing

[1] If $X(t)$ is a random variable indexed by real number then the set $\{X(t)\}_{t \in [a,b]}$ is a stochastic process on the interval $[a, b]$.

the right-hand-side of the inequality

$$0 \leq \lim_{N\to\infty} \mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}} \mathbb{E}_{\{\boldsymbol{x}_k \sim p_X(\boldsymbol{x}_k, t_k|\boldsymbol{x}_{k-1}, t_{k-1})\}_{k=1}^N} \left[ \ln \frac{p_{\text{data}}(\boldsymbol{y}) \prod_{k=1}^N p_X(\boldsymbol{x}_k, t_k|\boldsymbol{x}_{k-1}, t_{k-1})}{p_{\text{noise}}(\boldsymbol{x}_N) \prod_{k=1}^N p_{X_R}(\boldsymbol{x}_{k-1}, t_{k-1}|\boldsymbol{x}_k, t_k)} \right]$$

$$= \int_0^T \mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}} \mathbb{E}_{\boldsymbol{x}\sim p_X(\boldsymbol{x},t|\boldsymbol{y})} \left[ \delta\mathcal{F}(\boldsymbol{x},t)^T D(t)^{-1} \delta\mathcal{F}(\boldsymbol{x},t) \right] dt \tag{1}$$

$$\overset{\nabla_{\boldsymbol{\theta}}}{=} \int_0^T \mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}} \mathbb{E}_{\boldsymbol{x}\sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{y},\sigma(t)^2)} \left[ \frac{\|N(\boldsymbol{x},t) - (\boldsymbol{x}-\boldsymbol{y})\|^2}{\sigma(t)^2} \right] \mu(t) dt \tag{2}$$

$$= \int_0^{\sigma(T)} \mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}} \mathbb{E}_{\boldsymbol{x}\sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{y},\sigma^2)} \left[ \frac{\|N(\boldsymbol{x},\sigma) - (\boldsymbol{x}-\boldsymbol{y})\|^2}{\sigma^2} \right] \frac{1}{\sigma} d\sigma \tag{3}$$

with $N(\boldsymbol{x}, t)$ a parameterised predictor for the noise $\boldsymbol{n} := \boldsymbol{x} - \boldsymbol{y}$ and $\mu(t) = d\ln\sigma(t)/dt$ the entropy production rate (Table 1). Note that equality holds between Eq. (1) and Eq. (2) if the gradient with respect to parameters is taken, which we denote by $\nabla_{\boldsymbol{\theta}}$ above the equality sign. In this case, the relation between the noise predictor and the flow field is given by

$$\begin{aligned} \delta\mathcal{F}(\boldsymbol{x},t) &:= -\mathcal{F}(\boldsymbol{x},t)/2 + D(t)\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x},t) \\ \mathcal{F}(\boldsymbol{x},t) &= -2\mu(t) N(\boldsymbol{x},t) \end{aligned} \tag{4}$$

The right-hand-side of Eq. (1) is minimal for $\delta\mathcal{F} = 0$, which implies that the optimal noise predictor is given by $N^*(\boldsymbol{x}, t) := -\sigma(t)^2 \nabla \log p(\boldsymbol{x}, t)$. From Eq. (3) follows that $\sigma$ can be interpreted as alternative time scale (Karras et al., 2022). From a physical viewpoint, the denoising function $-N(\boldsymbol{x}, t)$ should be interpreted as force, $\sigma^2(t)$ as temperature, and $\mu(t)$ as mobility coefficient (Table 1). As a consequence of detailed balance, $D(t)$ and $\mu(t)$ are related by the fluctuation-dissipation theorem. The training objective Eq.(3), differs from previous denoising score-matching objectives (Karras et al., 2022; Kingma et al., 2023; Song et al., 2021) by the functional form of the loss weight. Our loss weight is the consequence of matching the probability densities for the forward and reverse processes using the Kullback-Leibler divergence, which is different from learning $\mathcal{F}$ directly by a denoising score-matching objective (Vincent, 2011).

Table 1: Relation of diffusion models to stochastic thermodynamics

| **Physical Systems** | **Diffusion Models** | **Comment** |
|---|---|---|
| Energy scale | $\sigma(t)^2$ | temperature ($k_B T$) |
| Energy function | $E := \|\boldsymbol{x} - \boldsymbol{y}\|^2/2$ | $\boldsymbol{x} \in \mathbb{R}^d$ |
| Internal Energy (per dim) | $U/d = \langle E\rangle/d = \sigma(t)^2/2$ | $\langle . \rangle :=$ equilibrium average |
| Entropy (per dim) | $S/d = 1/2 \ln \sigma(t)^2 + const$ | $\partial S/\partial U = (k_B T)^{-1}$ |
| Force | $F = -(\boldsymbol{x} - \boldsymbol{y})$ | $F := -\nabla_{\boldsymbol{x}} E$ |
| Diffusion coefficient (per dim) | $D(t) = (1/2) d\sigma(t)^2/dt$ | energy absorption rate |
| Mobility coefficient (per dim) | $\mu(t) = (1/2) d\ln\sigma^2/dt$ | entropy production rate |
| Fluct. Diss. Theorem | $\mu(t) = D(t)/\sigma(t)^2$ | conseq. of detailed balance |
| Equilibrium distribution | $\mathcal{N}(\boldsymbol{x}\|\boldsymbol{y}, \sigma(t)^2)$ | quasi-static equilibrium |
| Free Energy (rel. to equilibrium) | $\sigma^2 D_{KL}(P(\boldsymbol{x})\|\mathcal{N}(\boldsymbol{x}\|\boldsymbol{y}, \sigma(t)^2))$ | $P(\boldsymbol{x})$ : some distribution |

### 2.2 Asymptotic behavior of noise predictor

In can be shown that the optimal noise predictor has the asymptotic behavior (Appendix B)

$$\begin{aligned} N^*(\boldsymbol{x},\sigma) &= \mathcal{O}(\sigma) && \text{outside the data distribution, } p_{\text{data}}(\boldsymbol{x}) = 0 \\ N^*(\boldsymbol{x},\sigma) &= \mathcal{O}(\sigma^2) && \text{inside the data distribution, } \sigma \to 0 \end{aligned}$$

As the underlying data distribution from which the training set was sampled is unknown, it is unclear how to incorporate the correct asymptotic behavior for $N(\boldsymbol{x}, \sigma)$ into the training objective. Our current objective, Eq. (3), implies that $N(\boldsymbol{x}, \sigma) = \mathcal{O}(\sigma)$ as a consequence of $\boldsymbol{x} - \boldsymbol{y} = \mathcal{O}(\sigma)$ for all noise scales $\sigma > 0$. Consequently, the objective Eq. (3) is only valid outside the data distribution.

We therefore introduce with $\sigma_\epsilon$ a cutoff value, above which Eq. (3) remains valid. We emphasize that characterizing the thickness of the data hyperplane by a single scalar value $\sigma_\epsilon$ is a strong over-simplification that leaves room for further improvements. To realize the cutoff we multiply the score-matching loss by a sigmoid function, using the substitution

$$\frac{1}{\sigma} d\sigma \to \frac{1}{\sigma} \text{sigmoid}(\gamma \ln(\sigma/\sigma_\epsilon))\, d\sigma = \frac{\sigma^{\gamma-1}}{\sigma^\gamma + \sigma_\epsilon^\gamma}\, d\sigma$$

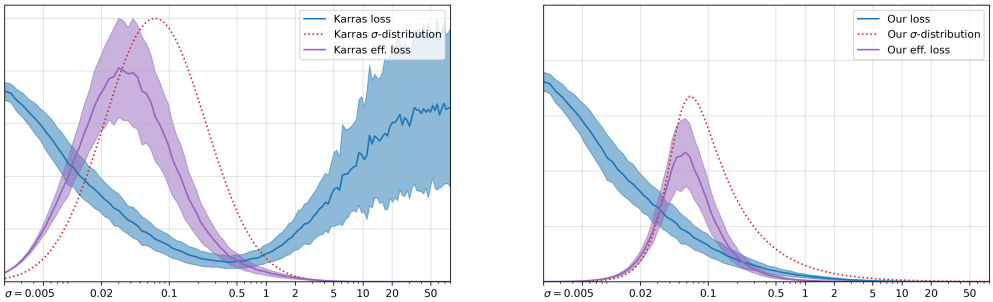
where the hyperparameter $\gamma$ characterizes the sharpness of the cutoff. We set $\gamma = 6$, as lower values showed reduced performance and higher values resulted in training instabilities. For computational efficiency, we normalize the loss weight and introduce the probability distribution

$$q(\sigma) := \left[\frac{1}{\gamma} \ln \frac{\sigma_T^\gamma + \sigma_\epsilon^\gamma}{\sigma_\epsilon^\gamma}\right]^{-1} \frac{\sigma^{\gamma-1}}{\sigma^\gamma + \sigma_\epsilon^\gamma}\, d\sigma$$

which allows us to substitute the loss weight by sampling over noise levels

$$\mathcal{L} = \mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}(\boldsymbol{y})} \mathbb{E}_{\sigma\sim q(\sigma)} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{N}(\boldsymbol{x}|\boldsymbol{y},\sigma^2)} \left[\frac{\|\sigma f(\boldsymbol{x},\sigma) - (\boldsymbol{x}-\boldsymbol{y})\|^2}{\sigma^2}\right] \tag{5}$$

We enforce the correct noise scaling for noise predictor by normalizing the neural network output, $f(\boldsymbol{x},\sigma) = \mathcal{O}(1)$ and defining $N(\boldsymbol{x},\sigma) := \sigma f(\boldsymbol{x},\sigma)$, using the preconditioning pipeline of Karras et al. (2022). It has been found empirically that training the score-matching objective with higher loss weight on intermediate noise levels improves the FID score (Kynkäänniemi et al., 2023). Our analysis shows that the loss weight is determined by the choice of how to quantify the statistical differences between forward and reverse trajectories and by accounting for the denoising asymptotics near $\sigma \to 0$. After coarse-grained optimization of the hyperparameter $\sigma_\epsilon$ (Fig. 3a) our loss weight achieves comparatively good results as the empirically found loss weights (Table 2), despite significant differences in their functional forms (Fig. 2).




(a) EDM (b) Ours

Figure 2: Comparison of the score-matching loss terms (square bracket of Eq.(5), blue lines) and loss weight $q(\sigma)$ (dashed lines) between our approach with $\sigma_\epsilon = 0.05$ (right) and (Karras et al., 2022) (left) for CIFAR-10. The loss function seen by the optimizer is shown in purple. The distributions (dotted lines) are normalized but ours appears smaller due to the heavier tail for large $\sigma$. We have taken the hyperparameters for the EDM loss weight (log-normal distribution) from the original code repository and used these to reproduce the reported FID scores.

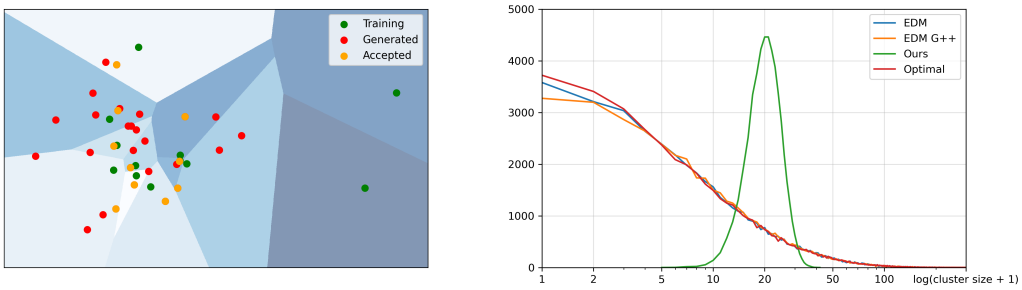
| Model | EDM | Ours | Ours |
|---|---|---|---|
| Sampler | Vanilla EDM | Vanilla EDM | Tuned EDM |
| FID | 1.80[2] | 1.88 | 1.83 |

Table 2: Our training results on conditional CIFAR-10, compared to Karras et al. (2022) EDM Model. Vanilla EDM refers to using the EDM sampling settings which were optimized for the EDM model on CIFAR-10. The second FID shows our result when using this sampler out of the box. Tuning $\rho$ and $n_{steps}$, as shown in Fig. 5, results in an FID score comparable to the EDM model.

[2]Karras et al. (2022) reported the minimum of three FID runs, resulting in a score of 1.79, which we reproduced. We show the mean of the three runs reported, which included the reproduction.

## 3 SAMPLING BIAS IN FEATURE SPACE

An optimal generative model samples from the same data distribution as used to build the training set. Practically, we expect from a generative model that it produces samples that (i) generalize well across the training set, (ii) show similar high variation in feature space as the training examples, and (iii) are not recognizable as out-of-distribution (OOD). However, the trade-off (i)-(iii) is hard to achieve as increasing the variation almost certainly increases the chance to generate OOD samples, and generalization is not possible across outliers in the training set. To quantify the bias in the sampling process for diffusion models, we counted the number of generated examples that share the same training data point as nearest neighbour in feature space. If the model generalizes with respect to both higher-level features and lower-level features, we expect the sampled data points to be close to the training set in this space, assigning each training example the same number of nearest neighbor samples up to stochastic fluctuations. However, after sampling $10^6$ data points we found strong differences in the densities of generated samples and training samples in feature space (Fig. 4b), which indicates a strong sampling bias.




(a) Feature space rejection

(b) CIFAR-10 sampling bias

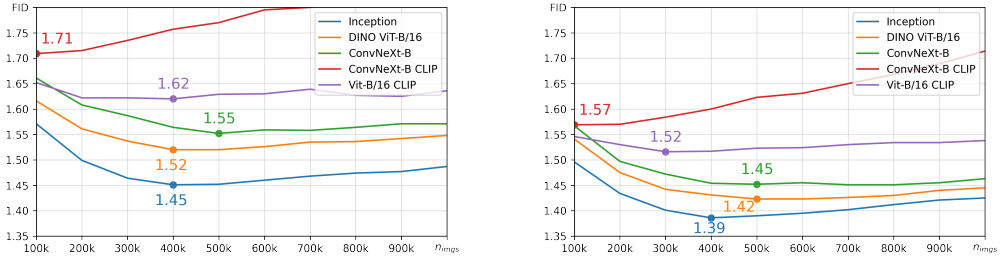
Figure 3: **(a)** Illustration of the rejection rule of the N3-rejection sampler. A generated sample is accepted if it is at least as close to a training example as the corresponding nearest neighbor from a pre-selected set of samples and if it shares the same shaded region. The shaded regions are the result of k-means clustering, with $k$ the size of the training set and training examples taken as cluster centers. Distances are defined by the cosine similarity in feature space. **(b)** Visualization of the sampling bias for diffusion models on CIFAR-10, using DINO ViT-B/16 (Caron et al., 2021) as feature extractor. Shown is the distribution of cluster sizes – how often each training image is targeted by generated samples (number of red points in the shaded regions of (a)). An optimal sampler targets all training examples with the same probability (green). Our model is slightly less biased in comparison to EDM and EDM G++.

### 3.1 BIAS CORRECTING BY REJECTION SAMPLING

To correct for the sampling bias, we assign each training example a non-overlapping, surrounding region in feature space (Fig. 4a). The regions are taken to be small for training examples that have a disproportionally high number of nearest neighbor samples and large in the opposite case. The nearest neighbor regions are determined by Algorithm 1, which finds the next nearest neighbor to each training example across a sufficiently large set of generated examples. The boundaries of the regions are defined by cosine similarity between the next nearest neighbor and its training examples. Samples that fall in these regions are accepted – and rejected if they lie outside these regions. As a consequence, the accepted samples follow more closely the distribution of the training set in feature space. To find a suitable feature space, we tested different feature extractors. As network architectures, we tried ConvNeXt and Vision transformers in combination with class-supervised, language-supervised (CLIP), and self-supervised pretraining. The reason for this choice was the strong OOD prediction performance for these settings (Michels et al., 2023; Ming et al., 2022).

### 3.2 FEATURE EXTRACTOR COMPARISON

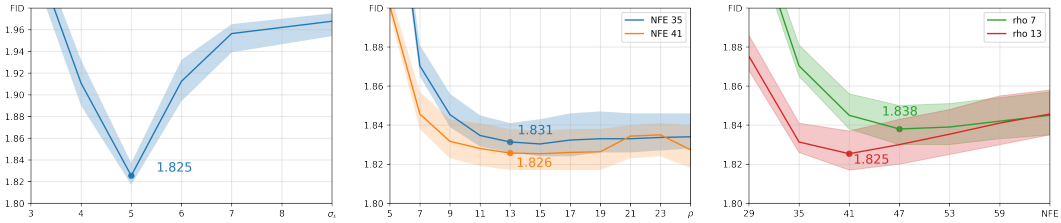
Algorithm 1 assigns only sampling regions around training examples if there is at least one nearest neighbor assigned to them. After the generation of 1 Mio. samples, approximately 7% of the




(a) EDM (b) EDM-G++

Figure 4: Finding the optimal set size $n_{imgs}$ of generated samples to determine the acceptance threshold (nearest neighbor distance) using Algorithm 1. Surprisingly, CLIP features perform worst and DINO features perform best. **(a)** Generated samples from EDM. **(b)** Generated samples from EDM G++.

50k training examples of CIFAR-10 lack a nearest neighbour. Training examples without nearest neighbors frequently look atypical (Fig. 1) and seem to represent outliers over which the model is not generalizing well. If we increase the number of generated examples from which Algorithm 1 can select, we increase the likelihood that training set outliers get assigned a nearest neighbor, which is then accepted as the next nearest neighbor. Consequently, the distribution of accepted samples spreads over the training set but image quality drops by producing samples that are far from every training sample in feature space, using cosine similarity as a distance measure. The combined effect is that FID declines with increasing the number of samples to select the next nearest neighbors but eventually rises again. We take the generated sample set size with minimum FID as an optimal point to determine the regions for rejection sampling (Fig. 4). It is somewhat surprising to see that among the models for feature extraction an unsupervised method (DINO) results in the lowest FID.




(a) $\sigma_\epsilon$ comparison (b) $\rho$ comparison (c) NFE comparison

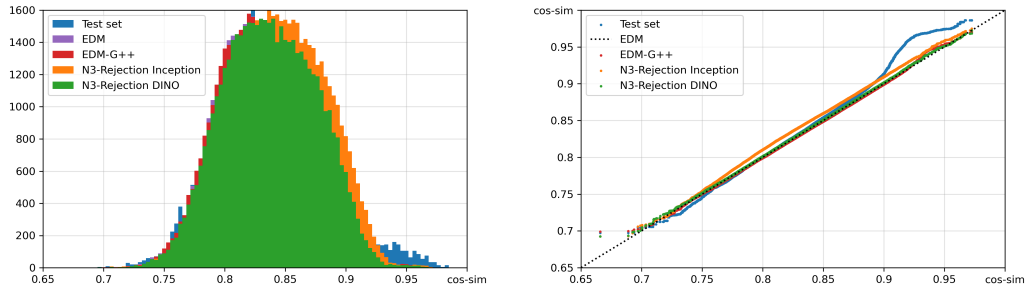
Figure 5: Hyperparameter tuning for our model in the class conditional setting for CIFAR-10. All points are averages of three runs and the shaded area reflects the minimum and maximum values for these runs. The hyperparameter $\rho$ has been introduced by Karras et al. (2022) to allow adjustment of the denoising step size. NFE is the number of network calls needed for sampling. Training details can be found in Appendix D.

Our rejection sampling approach could be criticized for simply selecting generated examples that are closer in cosine similarity to training examples in the Inception feature space, as this would naturally lower the FID. However, the distribution of cosine similarities in this space is unchanged after rejection sampling, even with the best-performing feature extractor with respect to FID (Fig. 6). This shows that reducing the distance between generated samples and training examples in DINO feature space does not change their distance in Inception feature space. Similar behavior is observed for discriminator-guided diffusion, which brings generated examples and training examples closer in real space (Kim et al., 2023). Also here, no significant shift of Inception features is observed (Fig. 6).

**Algorithm 1** PyTorch-like pseudocode of the N3-rejection sampler.

```
# train_features [n, f_dim] - normalized features of the train set
# feature_model             - Feature extraction model, e.g. ConvNeXt
# all_seeds [n_seeds]       - Seeds for reproducible image generation
# best_seeds [n_train]      - Empty tensor. Save the best seed for each training image
# best_scores [n_train]     - Empty tensor. Save the best score for each training image

# Loop through all seeds batch-wise
for batch_seeds in all_seeds:
    # Generate images and extract features
    x = edm_sampler(batch_seeds)  # [bs, C, H, W]
    feats = get_features(x, feature_model)  # [bs, f_dim]
    feats = torch.nn.functional.normalize(feats)

    # Compute cosine similarity to all training images
    score = feats @ train_features.t()  # [bs,n]

    # Only keep score for best matching training image
    best_score, best_matches = score.max(1, keepdim=True)
    scores = torch.zeros_like(score)
    scores = torch.scatter(updates, 1, best_matches, best_score) # [bs,n]

    # Select the closest match for each training image among best matches
    max_scores, max_idx = scores.max(dim=0)  # [n]

    # If there is a better match than before, keep it
    condition = max_scores > best_scores
    best_seeds = torch.where(condition, batch_seeds[max_idx], best_seeds)
return best_seeds
```

The rejection sampling results are summarised in Table 3. The combination of Discriminator Guided Diffusion (EDM-G++) and rejection sampling in DINO feature spaces results in the lowest FID. This result indicates that these two methods for improving FID are to some extent complementary. We admit that have tested only a very limited amount of feature extractors, given the wide spectrum of pre-trained models currently available. It is therefore almost certain that a even better feature extractor can be constructed. However, the question is if FID is the right measure for both the quality and the distribution of generated samples. We share the concerns of using FID for benchmarking generative models (Chong & Forsyth, 2020; Kynkäänniemi et al., 2023; Betzalel et al., 2022), especially because we have no explanation for why the CLIP models behave worse despite the fact that they are excellent OOD detectors.




(a) Histogram of cosine similarities.

(b) QQ-plot of cosine similarities

Figure 6: Statistical evaluation of cosine similarities in Inception v3 (Szegedy et al., 2015) feature space for 50k samples on CIFAR-10 . Each cosine similarity is between a generated image and its closest training image in feature space. (a) Histogram of cosine similarities showing that N3-rejection on Inception features induces the expected distribution shift towards higher overlap with the training set, indicating a direct manipulation of FID. (b) QQ-plot illustrates the absence of a distribution shift if N3-Rejection is carried out in DINO feature space and the resulting accepted images are evaluated in Inception feature space.

.

| Sampling Method | Feature Model | FID | Rel. time[3] |
|---|---|---|---|
| EDM Sampler (Karras et al., 2022) | - | 1.80[4] | 1× |
| EDM-G++ (Kim et al., 2023) | - | 1.64 | 2× |
| N3-rejection | ConvNeXt-B (CLIP) | 1.71 | 2× |
| N3-rejection | ViT-B/16 (CLIP) | 1.62 | 8× |
| N3-rejection | ConvNeXt-B | 1.55 | 10× |
| N3-rejection | ViT-B/16 (DINO) | **1.52** | 8× |
| EDM-G++ & N3-rejection | ConvNeXt-B (CLIP) | 1.57 | 4× |
| EDM-G++ & N3-rejection | ViT-B/16 (CLIP) | 1.52 | 12× |
| EDM-G++ & N3-rejection | ConvNeXt-B | 1.45 | 20× |
| EDM-G++ & N3-rejection | ViT-B/16 (DINO) | **1.42** | 20× |
| N3-rejection | Inception | 1.45 | 8× |
| EDM-G++ & N3-rejection | Inception | 1.39 | 16× |

Table 3: Performance comparison for different sampling methods with Karras et al. (2022) EDM model on class conditional CIFAR-10.

## 4 CONCLUSION

In this work, we show that the bias of diffusion models can be attenuated if the denoising score-matching objective is correctly weighted during training and the generated samples cover more closely the training set in feature space. Our results demonstrate that the proposed rejection sampling approach for reducing the sampling bias achieves new SOTA FIDs on CIFAR-10. In particular, we find that the combination of a discriminator guided diffusion model in combination with our rejection sampling approach leads to FID of 1.42 for class conditional CIFAR-10. The disadvantage of our approach is the slower sampling speed, as roughly one out of ten examples are rejected. Negative societal effects arise from increased energy consumption per generated image as a consequence of rejection sampling and increasing the image quality increases the quality of deepfakes and their potential risks.

[3] Approximations on our hardware. The speed differences for feature models are mainly due to them reaching their optimal FID for different amounts of total samples, not the evaluation speed of the feature models themselves.

[4] Karras et al. (2022) reported the minimum of three FID runs, resulting in a score of 1.79, which we reproduced. We show the mean of the three runs reported, which included the reproduction.

### AUTHOR CONTRIBUTIONS

If you'd like to, you may include a section for author contributions as is done in many journals. This is optional and at the discretion of the authors.

### ACKNOWLEDGMENTS

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

## REFERENCES

Eyal Betzalel, Coby Penso, Aviv Navon, and Ethan Fetaya. A study on the evaluation of generative models, 2022.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021.

Min Jin Chong and David Forsyth. Effectively unbiased fid and inception score and where to find them, 2020.

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, 2018.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models, 2022.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models, 2022.

Dongjun Kim, Yeongmin Kim, Se Jung Kwon, Wanmo Kang, and Il-Chul Moon. Refining generative process with discriminator guidance in score-based diffusion models, 2023.

Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models, 2023.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis, 2021.

Tuomas Kynkäänniemi, Tero Karras, Miika Aittala, Timo Aila, and Jaakko Lehtinen. The role of imagenet classes in fréchet inception distance, 2023.

Felix Michels, Nikolas Adaloglou, Tim Kaiser, and Markus Kollmann. Contrastive language-image pretrained (clip) models are powerful out-of-distribution detectors. *arXiv preprint arXiv:2303.05828*, 2023.

Yifei Ming, Ziyang Cai, Jiuxiang Gu, Yiyou Sun, Wei Li, and Yixuan Li. Delving into out-of-distribution detection with vision-language representations. *Advances in Neural Information Processing Systems*, 35:35087–35102, 2022.

Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail Kudinov. Grad-tts: A diffusion probabilistic model for text-to-speech, 2021.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

Pascal Vincent. A connection between score matching and denoising autoencoders, 2011.

Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training, 2020.

## A FIRST-PRINCIPLES DERIVATION OF THE DIFFUSION OBJECTIVE

The stochastic differential equation (SDE) for the forward (noising) process is given by

$$dX(t) = \sqrt{2D(t)}dW(t) \qquad ; \quad X(0) \sim p_{\text{data}}(\boldsymbol{y})$$

and the SDE for the reverse (denoising) process is given by

$$dX_R(t) = \mathcal{F}(X_R(t), t)dt + \sqrt{2D_R(t)}dW(t) \qquad ; \quad X_R(0) \sim p_{\text{noise}}(\boldsymbol{x})$$

with $0 \leq t \leq T$ and $W(t)$ the standard Wiener process that injects Gaussian noise. If the forward and the reverse process produce identical trajectories, their stochastic variables are related by $X(t) = X_R(T-t)$.

The SDEs defined above are meaningless unless a rule for their integration is defined. Here, we use the midpoint (or Stratonovich or Heun) rule [5]. For discrete update steps, $t_n = n\Delta t$ with $n \in \{0, .., N\}$, application of the midpoint rule results in

$$X_{R,n} = X_{R,n-1} + \frac{1}{2}[\mathcal{F}(X_{R,n}, t_n) + \mathcal{F}(X_{R,n}, t_{n-1})]\Delta t + \sqrt{2D_{R,n}\Delta t}\,\Xi_n$$

with $\Xi_n \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $X_{R,n} := X_R(t_n)$. The midpoint rule has the advantage that rules from ordinary calculus can be applied to transform stochastic variables. Consequently, the joint probability density $p(\Xi_1, .., \Xi_N) = \prod_{n=1}^{N} \mathcal{N}(\Xi_n|\mathbf{0}, \boldsymbol{I})$ can be transformed into the joint probability density for the discrete realizations $\boldsymbol{x}_n$ of the stochastic variable $X_n$ using the Jacobian $J_{nl} = d\Xi_n/dx_l$ and the identity $p_R(\boldsymbol{x}_1, .., \boldsymbol{x}_N) = \det(J)p(\Xi_1, .., \Xi_N)$. The probability density for a trajectory generated by the discrete reverse process that passes through the points $\boldsymbol{x}_0, \boldsymbol{x}_1, .., \boldsymbol{x}_N$ is given up to order $\mathcal{O}(\Delta t^2)$ by

$$p_R(\boldsymbol{x}_N, .., \boldsymbol{x}_1|\boldsymbol{x}_0) = \frac{1}{\prod_{n=1}^{N}\sqrt{4\pi\det(D_{R,n}\Delta t}} \exp\left[-\frac{1}{2}\sum_{n=1}^{N}\boldsymbol{\xi}_n^T\boldsymbol{\xi}_n - \frac{1}{2}\sum_{n=1}^{N}\nabla_{\boldsymbol{x}_n}\cdot\mathcal{F}(\boldsymbol{x}_n, t_n)\Delta t\right]$$

with

$$\boldsymbol{\xi}_n := \frac{\boldsymbol{x}_n - \boldsymbol{x}_{n-1} - \frac{1}{2}[\mathcal{F}(\boldsymbol{x}_n, t_n) + \mathcal{F}(\boldsymbol{x}_{n-1}, t_{n-1})]\Delta t}{\sqrt{2D_{R,n}\Delta t}}$$

In the limit, $\Delta t \to 0, N \to \infty, N\Delta t = T$, we arrive at the path integral representation

$$p_R[\boldsymbol{x}(t)|\boldsymbol{x}(0)] \propto \exp\left[-\frac{1}{4}\int_0^T [\dot{\boldsymbol{x}} - \mathcal{F}(\boldsymbol{x}, t)]^T D_R(t)^{-1}[\dot{\boldsymbol{x}} - \mathcal{F}(\boldsymbol{x}, t)] - \frac{1}{2}\int_0^T \nabla_{\boldsymbol{x}}\cdot\mathcal{F}(\boldsymbol{x}, t)dt\right]$$

where we omitted to write out the normalization. Here, $p_R[.]$ indicates that joint probability density is now a functional of the continuous trajectory $\boldsymbol{x}(t)$. A path integral representation for the forward process, $p[\boldsymbol{x}(t)|\boldsymbol{x}(0)]$, can be obtained by simply setting $\mathcal{F} = 0$ and subsituting $D_R \to D$.

We now set $D_R(t) = D(t)$ and introduce $F_a := -\mathcal{F}/2$. We now express the path integral for reverse process by integrating along the forward trajectories. This can be achieved by reversing the time arrow for the reverse trajectories, which amounts in the substitution $\boldsymbol{x}(t) \to \boldsymbol{x}(T-t)$ and as a consequence $\dot{\boldsymbol{x}}(t) \to -\dot{\boldsymbol{x}}(t)$. The log ratio of the probability densities is then given by [6]

$$\ln\frac{p[\boldsymbol{x}(t)|\boldsymbol{x}_0]}{p_R[\boldsymbol{x}(T-t)|\boldsymbol{x}_T]} = \int_0^T [\dot{\boldsymbol{x}}(t) - F_a(\boldsymbol{x}(t), t)]^T D(t)^{-1}[-F_a(\boldsymbol{x}(t), t)]dt - \int_0^T \nabla_{\boldsymbol{x}}\cdot F_a(\boldsymbol{x}, t)dt$$

The Fokker-Planck (or Smoluchowski) equation $\partial_t p(x, t) = -\nabla_{\boldsymbol{x}}\cdot F_B(\boldsymbol{x}, t)p(\boldsymbol{x}, t)$ describes the time evolution of the probability density, $p(x, t)$, at position $\boldsymbol{x}$ for a forward process with $F = 0$.

[5] For $x(t)$ a stochastic process we have $dx^2 \propto dt$ and get $\int_{t_1}^{t_2}\frac{df(x(t))}{dx}\dot{x}(t)dt = f(x_2) - f(x_1) - \frac{1}{2}\int_{t_1}^{t_2}\frac{d^2f(x(t))}{dx(t)^2}dt$. The second term only vanishes when taking the midpoint rule in the discrete case.

[6] Note that $\int_0^T f(\boldsymbol{x}(T-t))dt = \int_0^T f(\boldsymbol{x}(u))du$.

Here, $F_B(\boldsymbol{x},t) := -D(t)\nabla \ln p(\boldsymbol{x},t)$ is the Brownian force. Together with the total differential $dP(\boldsymbol{x}(t),t) = [\partial_t P(\boldsymbol{x}(t),t) + \nabla_x P(\boldsymbol{x}(t),t)\dot{\boldsymbol{x}}(t)]dt$ we obtain

$$\begin{aligned}
\ln \frac{p(\boldsymbol{x}_T)}{p(\boldsymbol{x}_0)} &= \int_0^T \frac{d\ln p(\boldsymbol{x}(t),t)}{dt}dt \\
&= \int_0^T \frac{-\nabla_{\boldsymbol{x}}\cdot F_B(\boldsymbol{x}(t),t)p(\boldsymbol{x}(t),t) + \nabla_{\boldsymbol{x}} p(\boldsymbol{x}(t),t)\dot{\boldsymbol{x}}(t)}{p(\boldsymbol{x}(t),t)}dt \\
&= \int_0^T [\dot{\boldsymbol{x}}(t) - F_B(\boldsymbol{x}(t),t)]^T D(t)^{-1}[-F_B(\boldsymbol{x}(t),t)]dt - \int_0^T \nabla_{\boldsymbol{x}}\cdot F_B(\boldsymbol{x}(t),t)dt
\end{aligned}$$

Detailed Balance of trajectories can be achieved for $F_a = F_B$. Combining results we get with $\delta\mathcal{F}(\boldsymbol{x}(t),t) := F_a(\boldsymbol{x}(t),t) - F_B(\boldsymbol{x}(t),t)$

$$\begin{aligned}
0 &\leq \mathbb{E}_{\boldsymbol{x}(t)}\left[\ln \frac{p[\boldsymbol{x}(t)|\boldsymbol{x}_0]p(\boldsymbol{x}_0)}{p_R[\boldsymbol{x}(T-t)|\boldsymbol{x}_T]p(\boldsymbol{x}_T)}\right] \\
&= \mathbb{E}_{\boldsymbol{x}(t)}\left[-\int_0^T [\dot{\boldsymbol{x}} - F_B]^T D^{-1}\delta\mathcal{F}dt + \int_0^T \delta\mathcal{F}^T D^{-1}F_B dt + \int_0^T \delta\mathcal{F}^T D^{-1}\delta\mathcal{F}dt - \int_0^T \nabla_{\boldsymbol{x}}\cdot\delta\mathcal{F}dt\right] \\
&= \mathbb{E}_{\boldsymbol{x}(t)}\left[-\int_0^T [\dot{\boldsymbol{x}} - F_B]^T D^{-1}\delta\mathcal{F}dt + \int_0^T \delta\mathcal{F}^T D^{-1}\delta\mathcal{F}dt\right] \\
&= \mathbb{E}_{\boldsymbol{x}(t)}\left[\int_0^T \delta\mathcal{F}^T D^{-1}\delta\mathcal{F}dt\right] \qquad (6)
\end{aligned}$$

where $\mathbb{E}_{\boldsymbol{x}(t)}[.]$ denotes the expectation over the forward paths, $\boldsymbol{x}(t) \sim p[\boldsymbol{x}(t)|\boldsymbol{x}_0]p(\boldsymbol{x}_0)$. In going from the second to the third line we have used the relation

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{x}(t)}[\nabla_{\boldsymbol{x}}\cdot\delta\mathcal{F}(\boldsymbol{x}(t),t)] &= \int_{-\infty}^{\infty} p(\boldsymbol{x}(t),t)\nabla_{\boldsymbol{x}}\cdot\delta\mathcal{F}(\boldsymbol{x}(t),t)d\boldsymbol{x}(t) \\
&= -\int_{-\infty}^{\infty}\delta\mathcal{F}(\boldsymbol{x}(t),t)^T\nabla_{\boldsymbol{x}}p(\boldsymbol{x}(t),t)d\boldsymbol{x}(t) \\
&= \mathbb{E}_{\boldsymbol{x}(t)}\left[\delta\mathcal{F}(\boldsymbol{x}(t),t)^T D^{-1}(t)F_B(\boldsymbol{x}(t),t)\right] \qquad (7)
\end{aligned}$$

In going from the third to the fourth line, we write out the expectation $\mathbb{E}_{\boldsymbol{x}(t)}[.] = \int p(\boldsymbol{x}(t),t)[.]dx(t)$ and use

$$p(\boldsymbol{x}(t),\boldsymbol{x}(t-\Delta t),t|\boldsymbol{x}_0) \propto \exp\left[-\frac{(x(t)-x(t-\Delta t))^2}{4D(t)\Delta t}\right]\exp\left[-\frac{(x(t-\Delta t)-\boldsymbol{x}_0)^2}{2\sigma^2(t-\Delta t)}\right]$$

around the midpoint $\bar{x}(t) = (x(t)+x(t-\Delta t))/2$. Defining $\epsilon = (x(t)-x(t-\Delta t))/2$ we get for any non-stochastic field $G(\boldsymbol{x},t)$ in the limit $\Delta t \to 0$

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{x}(t)}\left[\dot{\boldsymbol{x}}(t)^T G(\boldsymbol{x}(t),t)\right] \\
&\quad= \mathbb{E}_{\boldsymbol{x}_0}\int p(\boldsymbol{x}(t),\boldsymbol{x}(t-\Delta t),t|\boldsymbol{x}_0)\left[\dot{\boldsymbol{x}}(t)^T G(\bar{\boldsymbol{x}}(t),t)\right]d\boldsymbol{x}(t)d\boldsymbol{x}(t-\Delta t) \\
&\quad= \mathbb{E}_{\boldsymbol{x}_0}\int \frac{1}{Z}\exp\left[-\frac{(2\epsilon)^2}{4D(t)\Delta t}\right]\exp\left[-\frac{(\bar{x}(t)-\boldsymbol{x}_0-\epsilon)^2}{2\sigma^2(t)}\right]\left[\frac{2\epsilon}{\Delta t}G(\bar{\boldsymbol{x}}(t),t)\right]d(2\epsilon)d\bar{x}(t) \\
&\quad= \mathbb{E}_{\boldsymbol{x}_0}\int \frac{1}{Z}\exp\left[-\frac{(\bar{x}(t)-\boldsymbol{x}_0)^2}{2\sigma^2(t)}\right]\frac{(\bar{x}(t)-\boldsymbol{x}_0)D(t)}{\sigma^2(t)}G(\bar{\boldsymbol{x}}(t),t)d\bar{x}(t) \\
&\quad= \mathbb{E}_{\bar{\boldsymbol{x}}(t)}\left[F_B(\bar{\boldsymbol{x}}(t),t)G(\bar{\boldsymbol{x}}(t),t)\right]
\end{aligned}$$

with $Z$ the associated normalisation [7].Thus we are left with training objective

$$0 \leq \mathbb{E}_{\boldsymbol{x}(t)}\left[\ln \frac{p[x(t)|\boldsymbol{x}_0]p(\boldsymbol{x}_0)}{p[x(T-t)|\boldsymbol{x}_T]p(\boldsymbol{x}_T)}\right] = \mathbb{E}_{\boldsymbol{x}(t)}\left[\int_0^T \delta\mathcal{F}(\boldsymbol{x}(t),t)^T D(t)^{-1}\delta\mathcal{F}(\boldsymbol{x}(t),t)dt\right]$$

Note that this objective is equivalent to the score-matching objective used for diffusion models by design.

[7]Note that $\mathbb{E}_{\boldsymbol{x}}\left[G(\boldsymbol{x})\nabla_{\boldsymbol{x}}\ln p(\boldsymbol{x},t)\right] = \int p(\boldsymbol{x},t)G(\boldsymbol{x})\nabla_{\boldsymbol{x}}\ln p(\boldsymbol{x},t)d\boldsymbol{x} = \mathbb{E}_{\boldsymbol{x}_0}\left[G(\boldsymbol{x})\nabla_{\boldsymbol{x}}p(\boldsymbol{x},t|\boldsymbol{x}_0)\right]$.

## B NOISE LEVEL ASYMPTOTICS OF THE NOISE PREDICTOR

The starting point of our analysis is Eq. (3 of the main text, from which we first drop all irrelevant multiplicative constants. We apply the gradient with respect to parameters, $\boldsymbol{\theta}$, to simplify the calculation by instantaneously removing all parameter-independent loss contributions. We want to investigate the behavior of term

$$J(\sigma, \theta) := \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{y} \sim p_{\text{data}}(\boldsymbol{y})} \mathbb{E}_{\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{y}, \sigma^2)} \left[ \| N(\boldsymbol{x}, \sigma) - (\boldsymbol{x} - \boldsymbol{y}) \|^2 \right]$$

inside and outside the data distribution. Outside the the data distribution we have $p_{\text{data}}(\boldsymbol{x}) = 0$ and the optimal noise predictor follows the asymptotic $N^*(\boldsymbol{x}, \sigma) = \boldsymbol{x} - \boldsymbol{y} = \sigma \boldsymbol{\eta} = \mathcal{O}(\sigma)$, with $\boldsymbol{\eta}$ standard normal distributed. Inside the data distribution, we have to evaluate the term

$$\lim_{\sigma \to 0} J(\sigma, \theta) = \lim_{\sigma \to 0} \int \int p_{\text{data}}(\boldsymbol{y}) \mathcal{N}(\boldsymbol{x}|\boldsymbol{y}, \sigma^2) \left[ N(\boldsymbol{x}, \sigma) - (\boldsymbol{x} - \boldsymbol{y}) \right]^T \nabla_{\boldsymbol{\theta}} N(\boldsymbol{x}, \sigma) d\boldsymbol{y} d\boldsymbol{x} \tag{8}$$

In the limit $\sigma \to 0$ the distribution $\mathcal{N}(\boldsymbol{x}|\boldsymbol{y}, \sigma^2)$ is more peaked than the data distribution and we can expand the data distribution around $\boldsymbol{x}$

$$p_{\text{data}}(\boldsymbol{x} - (\boldsymbol{x} - \boldsymbol{y})) = p_{\text{data}}(\boldsymbol{x}) - (\boldsymbol{x} - \boldsymbol{y})^T \nabla_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) + \mathcal{O}(\|\boldsymbol{x} - \boldsymbol{y}\|^2)$$

Carrying out the integral over $\boldsymbol{y}$, and collecting terms up to order $\sigma^2$ we arrive at

$$\lim_{\sigma \to 0} J(\sigma, \theta) = \int \left[ p_{\text{data}}(\boldsymbol{x}) N(\boldsymbol{x}, \sigma) + \sigma^2 \nabla_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \right]^T \nabla_{\boldsymbol{\theta}} N(\boldsymbol{x}, \sigma) d\boldsymbol{x} \tag{9}$$

$$= \lim_{\sigma \to 0} \int p_{\text{data}}(\boldsymbol{x}) \left[ N(\boldsymbol{x}, \sigma) + \sigma^2 \nabla_{\boldsymbol{x}} \ln p_{\text{data}}(\boldsymbol{x}) \right]^T \nabla_{\boldsymbol{\theta}} N(\boldsymbol{x}, \sigma) d\boldsymbol{x} \tag{10}$$

$$= \nabla_{\boldsymbol{\theta}} \lim_{\sigma \to 0} \int p_{\text{data}}(\boldsymbol{y}) \left\| N(\boldsymbol{y}, \sigma) + \sigma^2 \nabla_{\boldsymbol{y}} \ln p_{\text{data}}(\boldsymbol{y}) \right\|^2 d\boldsymbol{y} \tag{11}$$

This result shows that the optimal noise predictor is given by

$$N^*(\boldsymbol{x}, \sigma) = -\sigma^2 \nabla_{\boldsymbol{x}} \ln p_{\text{data}}(\boldsymbol{x}) = \mathcal{O}(\sigma^2) \quad ,$$

which drives the denoising trajectories towards local maxima in the data hyperplane.

## C THICKNESS OF THE DATA HYPERPLANE

Loosely speaking, $\sigma_\epsilon$ can be interpreted as the average "thickness" of the data hyperplane. Together with $\gamma$, it is the only hyperparameter of the derived diffusion objective in Eq. (5), determining the sampling distribution $p_{\sigma_\epsilon, \gamma}(\sigma)$ and thus how the noise levels $\sigma$ relate to the data hyperplane during training: $\sigma_\epsilon$ determines at what point the distribution goes towards 0 and $\gamma$ determines how fast this happens.

There are two reasons why we want to avoid learning in the in-distribution domain. Firstly, the asymptotic denoising behavior is unknown inside of the data distribution and secondly, learning in the in-distribution domain promotes overfitting, since we are fitting data points with an accuracy beyond the natural variation of the data distribution. The parameter $\sigma_\epsilon$ functions as a lower bound on the true variation of the data distribution across dimensions here. Changes on pixel level below this bound do not move the image outside of the data distribution. It can be empirically approximated by finding the direction with minimal variation at any point of the data distribution.

Using the exact "thickness" of the data hyperplane for $\sigma_\epsilon$ and $\gamma \to \infty$ allows $\sigma \sim p_{\sigma_\epsilon, \gamma}(\sigma)$ to approach the hyperplane infinitely close, but then drops the probability to 0. Since this is unknowable in practice, without knowing the true data distribution, we settle for smaller values for $\gamma$, yielding a smoother transition from high to low sample probability.

The true data distribution is unknown and we usually have a very limited number of true data samples available to us, which don't tightly cover the entire distribution. To get a better grasp on the full shape of the data distribution, we can employ methods that produce unlimited numbers of samples that still lie approximately within the data distribution, like data augmentation and adversarial examples. We are looking to determine the average "thickness" of the data hyperplane, i.e. its width in the direction of lowest variance in pixel-space.

A straightforward approach to find a good starting point for tuning $\sigma_\epsilon$ is to use the radius of adversarial examples for a data set, e.g. $\epsilon = 8/255$ for CIFAR-10 and $\epsilon = 2/255$ for Image-Net (Wong et al., 2020).

## D IMPLEMENTATIONAL DETAILS

All of our results were produced with our official code, which is based on the implementation of Karras et al. (2022) [8]. The code has been left unaltered in some places, mostly the organization of data and the fundamental setup of training runs, and has been modified in other places, like the training logic, image generation, data processing, experimental analysis, and network conditioning. When we refer to the EDM model for CIFAR-10, we refer to the "SongUNet" in the "ddpmpp" setup with "EDMPrecond" preconditioning. All necessary specifications to reproduce our results with our code can be found in 4. The result of 1.79 FID on conditional CIFAR-10 of Karras et al. (2022) has been reproduced using this code. All experiments were run using Python 3.9 and PyTorch 1.21.1 with CUDA toolkit 11.3.1 on 4 NVIDIA A-100 GPUs.

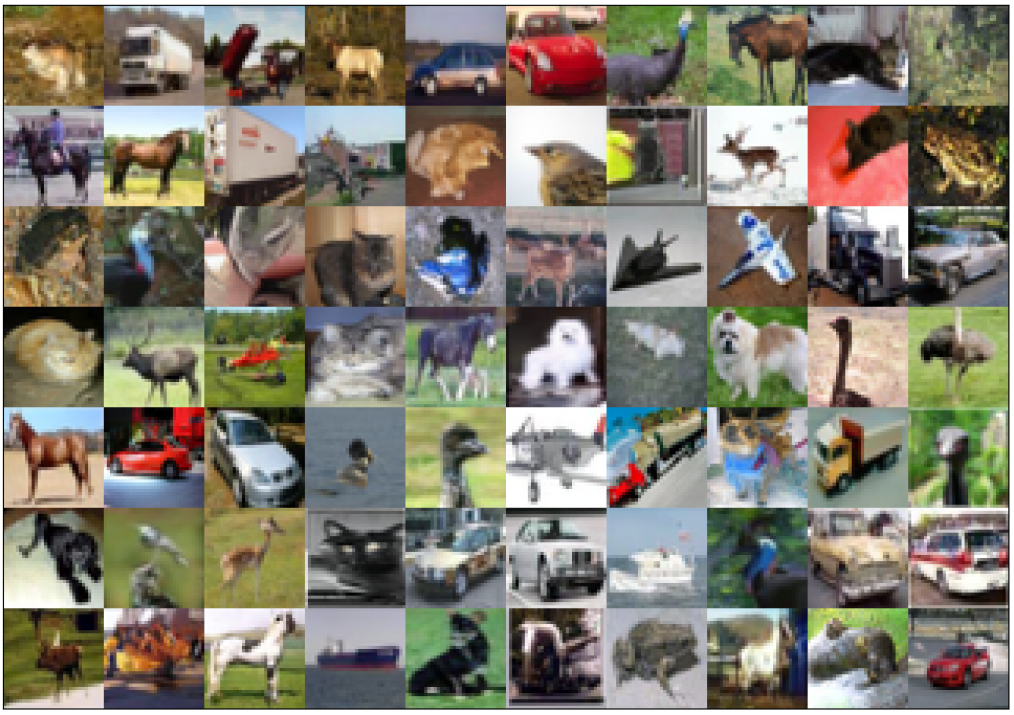
**FID calculation** The FID implementation was left untouched, see Karras et al. (2022) code for details. Fig. 5 used a mean of three FID runs with different random seeds, not the minimum. These random seeds were always chosen to be 0-49999, 50000-99999, and 100000-149999, for reproducibility.

| | | | |
|---|---|---|---|
| Training | Data | Number of GPUs | 4 |
| | | Duration (Mimg) | 200 |
| | | Batch size | 1024 |
| | | Seed | 1948593099 |
| | | Shuffle | False |
| | | Augment probabilities | 12% |
| | Optimizer | Learning Rate $\times 10^{-3}$ | 1 |
| | | LR ramp-up (Mimg) | 10 |
| | | EMA Half-life (Mimg) | 0.5 |
| | Architecture | Model type | DDPM++ |
| | | Channel Multiplier | 128 |
| | | Channels per resolution | 2 2 2 |
| | | Dropout Probability | 13% |
| | Parameters | $n_{steps}$ / NFE | 21 / 41 |
| | | $\rho$ | 13 |
| | | $\sigma_{min}$ | $2.0 \times 10^{-3}$ |
| | | $\sigma_{max}$ | 80 |
| Sampling | Parameters | $\sigma_\epsilon$ | 5.0 $\times 10^{-2}$ |
| | | $\gamma$ | 6 |
| | | $\sigma^{OD}_{min}$ | $1.0 \times 10^{-3}$ |
| | | $\sigma^{OD}_{max}$ | 90 |
| | | seeds | 100000-149999 |

Table 4: Training and sampling specifications for our model to reproduce results in table 2.

[8] https://github.com/NVlabs/edm

## E SAMPLES



(a) Random images that got accepted by the N3-rejection sampler (70/50k)

(b) The corresponding nearest neighbors in the train set (70/50k).

Figure 7: Random 50k subset of CIFAR-10 conditional samples of the N3-rejection sampler, using DINO ViT-B/16 (Caron et al. (2021)) as a feature model, achieving an FID of **1.52**.


Figure 8: Training images with the highest amount of nearest neighbors in feature space from Sthe N3-rejection sampler run, using the EDM model and DINO ViT-B/16 (Caron et al. (2021)) as a feature model, achieving an FID of **1.52**. The left column shows training images with the amount of nearest neighbors denoted next to it. The other columns show random subsets of these nearest neighbors. The model is particularly biased towards these images, generating them far too often.

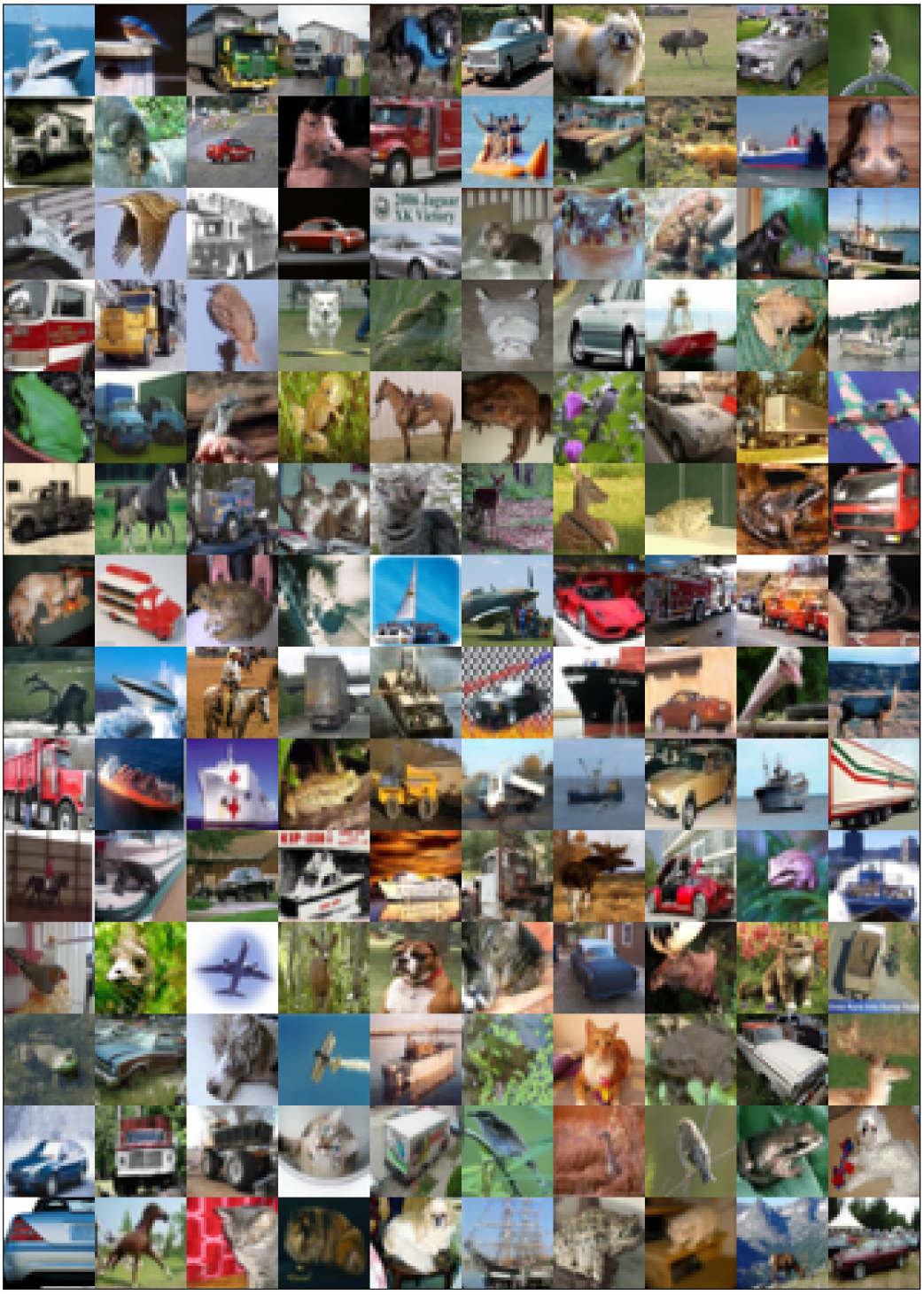


Figure 9: Training images without nearest neighbors in feature space from the N3-rejection sampler run, using the EDM model and DINO ViT-B/16 (Caron et al. (2021)) as a feature model, achieving an FID of **1.52**.

## F CHOICE OF TRAINING PARAMETERS

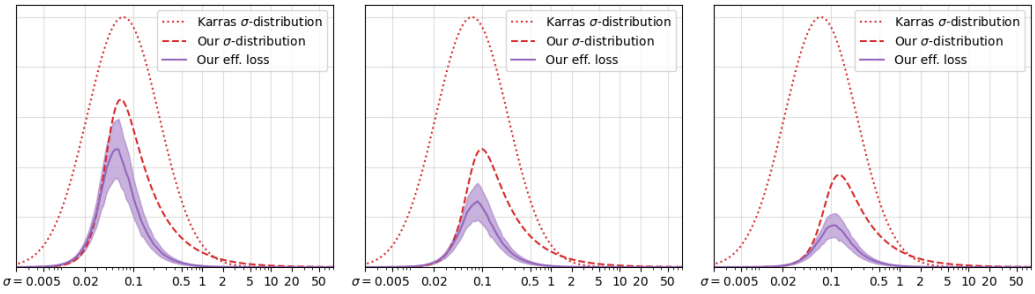



Figure 10: Comparison between different values for $\sigma_{eps}$ with $\gamma = 4$ with Karras et al. (2022) EDM model on CIFAR-10. **(a)** $\sigma_{eps} = 5e-2$, **(b)** $\sigma_{eps} = 7.5e-2$, **(b)** $\sigma_{eps} = 1e-1$.

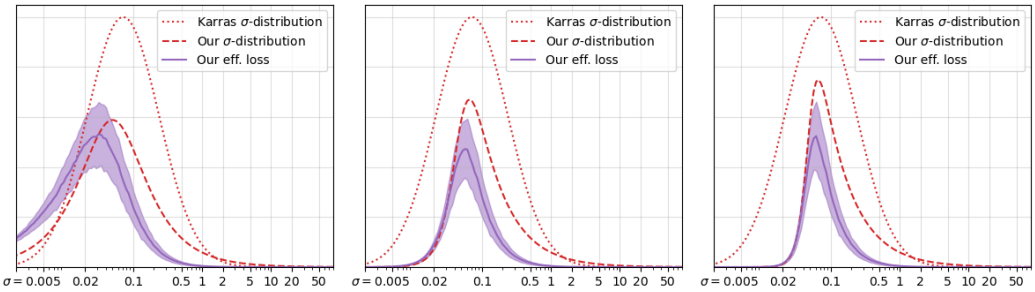



Figure 11: Comparison between different values for $\gamma$ with $\sigma_{eps} = 5e-2$ with Karras et al. (2022) EDM model on CIFAR-10. **(a)** $\gamma = 2$, **(b)** $\gamma = 4$, **(b)** $\gamma = 6$.

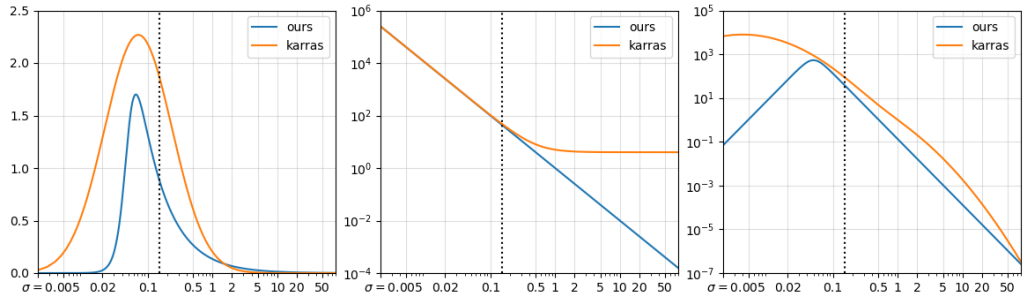



Figure 12: Comparison between Karras et al. (2022) EDM model and our **(a)** $\sigma$-distributions, **(b)** loss weight and **(b)** effective weight for $\gamma = 6$ and $\sigma_{eps} = 5e-2$ with Karras et al. (2022) EDM model on CIFAR-10.