Boosting Latent Diffusion with Perceptual Objectives

Tariq Berrada, Pietro Astolfi, Jakob Verbeek, Melissa Hall, Marton Havasi, Michal Drozdzal, Yohann Benchetrit, Adriana Romero-Soriano, Karteek Alahari

Boosting Latent Diffusion with Perceptual Objectives

Tariq Berrada Ifriqi[1,2], **Pietro Astolfi**[1], **Jakob Verbeek**[1], **Melissa Hall**[1], **Marton Havasi**[1], **Michal Drozdzal**[1], **Yohann Benchetrit**[1], **Adriana Romero-Soriano**[1,3,4,5], **Karteek Alahari**[2]

[1]FAIR at Meta, [2]Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, France, [3]McGill University, [4]Mila, Quebec AI institute, [5]Canada CIFAR AI chair

Latent diffusion models (LDMs) power state-of-the-art high-resolution generative image models. LDMs learn the data distribution in the latent space of an autoencoder (AE) and produce images by mapping the generated latents into RGB image space using the AE decoder. While this approach allows for efficient model training and sampling, it induces a disconnect between the training of the diffusion model and the decoder, resulting in a loss of detail in the generated images. To remediate this disconnect, we propose to leverage the internal features of the decoder to define a *latent perceptual loss* (LPL). This loss encourages the models to create sharper and more realistic images. Our loss can be seamlessly integrated with common autoencoders used in latent diffusion models, and can be applied to different generative modeling paradigms such as DDPM with epsilon and velocity prediction, as well as flow matching. Extensive experiments with models trained on three datasets at 256 and 512 resolution show improved quantitative – with boosts between 6% and 20% in FID – and qualitative results when using our perceptual loss.

Date: November 8, 2024
Correspondence: tariqberrada@meta.com

Meta

Figure 1 Samples from models trained with and without our latent perceptual loss on CC12M. Samples from our model with latent perceptual loss (bottom) have more detail and realistic textures.

1 Introduction

Latent diffusion models (LDMs) (Rombach et al., 2022) have enabled considerable advances in image generation, and elevated the problem of generative image modeling to a level where it has become available as a technology to the public. A critical part to this success is to define the generative model in the latent space of an autoencoder (AE), which reduces the resolution of the representation over which the model is defined, thereby making it possible to scale diffusion methods to larger datasets, resolutions, and architectures than original pixel-based diffusion models (Dhariwal and Nichol, 2021; Sohl-Dickstein et al., 2015).

To train an LDM, all images are first projected into a latent space with the encoder of a pre-trained autoencoder, and then, the diffusion model is optimized directly in the latent space. Note that when learning the diffusion model the AE decoder is not used – the diffusion model does not receive any training feedback that would ensure that all latent values reachable by the diffusion process decode to a high quality image. This training procedure leads to a disconnect between the diffusion model and the AE decoder, prompting the LDM to produce low quality images that oftentimes lack high frequency image components. Moreover, we note that the latent spaces of pre-trained LDM's autoencoders tend to be highly irregular, in the sense that small changes in the latent space can lead to large changes in the generated images, further exacerbating the autoencoder-diffusion disconnect problem.

In this work, we propose to alleviate this autoencoder-diffusion disconnect and propose to include the AE decoder in the training objective of LDM. In particular, we introduce latent perceptual loss (LPL) that acts on the decoder's intermediate features to enrich the training signal of LDM. This is similar to the use of perceptual losses for image-to-image translation tasks (Johnson et al., 2016; Zhang et al., 2018), but we apply this idea in the context of generative modeling and use the feature space of the pre-trained AE decoder rather than that of an external pre-trained discriminative network. Our latent perceptual loss results in sharper and more realistic images, and leads to better structural consistency than the baseline – see Figure 1. We validate LPL on three datasets of different sizes – the commonly used datasets ImageNet-1k (1M data points) and CC12M (12M data points), and additionally a private dataset S320M (320M data points) – as well as three generative models formulation – DDPM (Ho et al., 2020) with velocity and epsilon prediction, and conditional flow matching model (Lipman et al., 2023). In our experiments, we report standard image generative model metrics – such as FID (Heusel et al., 2017), CLIPScore (Hessel et al., 2021), as well as Precision and Recall (Sajjadi et al., 2018; Kynkäänniemi et al., 2019). Our experiments show that the use of LPL leads to consistent performance boosts between 6% and 20% in terms of FID. Our qualitative analysis further highlights the benefits of LPL, showing images that are sharp and contain high-frequency image details.

In summary, our contributions are:

- We propose the *latent perceptual loss (LPL)*, a perceptual loss variant leveraging the intermediate feature representation of the autoencoder's decoder.
- We present extensive experimental results on the ImageNet-1k, CC12M, and S320M datasets, demonstrating the benefits of LPL in boosting the model's quality by 6% to 20% in terms of FID.
- We show that LPL is effective for a variety of generative model formulations including DDPM and conditional flow matching approaches.

2 Related work

Diffusion models. The generative modeling landscape has been significantly impacted by diffusion models, surpassing previous state-of-the-art GAN-based methods (Brock et al., 2019; Karras et al., 2019, 2020, 2021). Diffusion models offer advantages such as more stable training and better scalability, and were successfully applied to a wide range of applications, including image generation (Chen et al., 2024; Ho et al., 2020), video generation (Ho et al., 2022b; Singer et al., 2023), music generation (Levy et al., 2023; San Roman et al., 2023), and text generation (Wu et al., 2023). Various improvements of the framework have been proposed, including different schedulers (Lin et al., 2024; Hang and Gu, 2024), loss weights (Choi et al., 2022; Hang et al., 2023), and more recently generalizations of the framework with flow matching (Lipman et al., 2023). In our work we evaluate the use of our latent perceptual loss in three different training paradigms: DDPM under noise and velocity prediction, as well as flow-based training with the optimal transport path.

Latent diffusion. Due to the iterative nature of the reverse diffusion process, training and sampling diffusion models is computationally demanding, in particular at high resolution. Different approaches have been explored to generate high-resolution content. For example, Ho et al. (2022a) used a cascaded approach to progressively add high-resolution details, by conditioning on previously generated lower resolution images. A more widely adopted approach is to define the generative model in the latent space induced by a pretrained autoencoder (Rombach et al., 2022), as previously explored for discrete autoregressive generative models (Esser et al., 2021). Different architectures have been explored to implement diffusion models in the latent space, including convolutional UNet-based architectures (Rombach et al., 2022; Podell et al., 2024), and more recently

transformer-based ones (Peebles and Xie, 2023; Chen et al., 2024; Gao et al., 2023; Esser et al., 2024) which show better scaling performance. Working in a lower-resolution latent space accelerates training and inference, but training models using a loss defined in the latent space also deprives them from matching high-frequency details from the training data distribution. Earlier approaches to address this problem include the use of a refiner model (Podell et al., 2024), which consists of a second diffusion model trained on high-resolution high-quality data that is used to noise and denoise the initial latents, similar to how SDEdit works for image editing (Meng et al., 2022). Our latent perceptual loss addresses this issue in an orthogonal manner by introducing a loss defined across different layers of the AE decoder in the latter stages of the training process. Our approach avoids the necessity of training on specialized curated data (Dai et al., 2023), and does not increase the computational cost of inference.

Perceptual losses. The use of internal features of a fixed, pre-trained deep neural network to compare images or image distributions has become common practice as they have been found to correlate to some extent with human judgement of similarity (Johnson et al., 2016; Zhang et al., 2018). An example of this is the widely used Fréchet Inception Distance (FID) to assess generative image models (Heusel et al., 2017). Such "perceptual" distances have also been found to be effective as a loss to train networks for image-to-image tasks and boost image quality as compared to using simple ℓ_1 or ℓ_2 reconstruction losses. They have been used to train autoencoders (Esser et al., 2021), models for semantic image synthesis (Isola et al., 2017; Berrada et al., 2024b) and super-resolution (Suvorov et al., 2022; Jo et al., 2020), and to assess the sample diversity of generative image models (Schönfeld et al., 2021; Astolfi et al., 2024). In addition, recent works propose variants that do not require pretrained image backbones (Amir and Weiss, 2021; Czolbe et al., 2020; Veeramacheneni et al., 2023). An et al. (2024); Song et al. (2023) employed LPIPS as metric function in pixel space to train cascaded diffusion models and consistency models, respectively. Most closely related to our work, Kang et al. (2024) used a perceptual loss defined over latents to distill LDMs to conditional GANs, but used a separate image classification network trained over latents rather than the autoencoder's decoder to obtain the features for this loss. In summary, compared to prior work on perceptual losses, our work is different in that (i) LPL is defined over the features of the decoder – that maps from latent space to RGB pixel space, rather than using a network that takes RGB images as input – and (ii) we use LPL to train latent diffusion models.

3 Using the latent decoder to define a perceptual loss

In this section, we analyze the impact of the decoder-diffusion disconnect on the LDM training, and then, we follow with the definition of our latent perceptual loss.

3.1 Latent diffusion and the MSE objective

We use F_β to refer to an autoencoder that consists of two modules. The encoder, F_β^e, maps RGB images $x_0 \in \mathbb{R}^{H\times W\times 3}$ to a latent representation $z_0 \in \mathbb{R}^{H/d\times W/d\times C}$, where d is the spatial downscaling factor, and C the channel dimension of the autoencoder. The decoder, F_β^d, maps from the latent space to the RGB image space. In LDM, the diffusion model, D_Θ, with parameters Θ is defined over the latent representation of the autoencoder. We follow a typical setting, see *e.g.* (Peebles and Xie, 2023; Chen et al., 2024; Rombach et al., 2022), where we use a fixed pre-trained autoencoder with a downsampling factor of $d = 8$ and a channel capacity of $C = 4$.

Training Objective. The diffusion formulation results in an objective function that is a lower-bound on the log-likelihood of the data. In the DDPM paradigm (Ho et al., 2020), the variational lower bound can be expressed as a sum of denoising score matching losses (Vincent, 2011), and the objective function can be written as $\mathcal{L} = \sum_t \mathcal{L}_t$, where $\mathcal{L}_t = \mathcal{D}_{\mathrm{KL}}\left[q\left(x_{t-1}|x_t, x_0\right)||p_\theta\left(x_{t-1}|x_t\right)\right] = \mathbb{E}_{x_0,\epsilon,t}\left[\frac{\beta_t}{(1-\beta_t)(1-\alpha_t)}\cdot\|\epsilon-\epsilon_\theta(x_t,t)\|^2\right]$. Ho et al. (2020) observed that disregarding the time step specific weighting resulted in improved sample quality, and introduced a simplified noise reconstruction objective, known as epsilon prediction, where the objective is the average of the MSE loss between the predicted noise and the noise vector added to the image, $\mathcal{L}_{\text{simple}} = \sum_t \mathbb{E}_{x_0,\epsilon,t}\left[\|\epsilon-\epsilon_\theta(x_t,t)\|^2\right]$. The underlying idea is that the better the noise estimation, the better the final sample quality. An equivalent way to interpret this objective is through reparameterization of the target to the original latents, $\mathcal{L}_{\text{simple}} = \sum_t \lambda_t \cdot \mathbb{E}_{x_0,\epsilon,t}\left[\|x_0-\hat{x}_0(x_t,t;\theta)\|^2\right]$, where $\hat{x}_0(x_t,t;\theta) = \sqrt{1+\sigma_t^2}x_t - \sigma_t\hat{\epsilon}_t$ and $\lambda_t = 1/\sigma_t^2$.

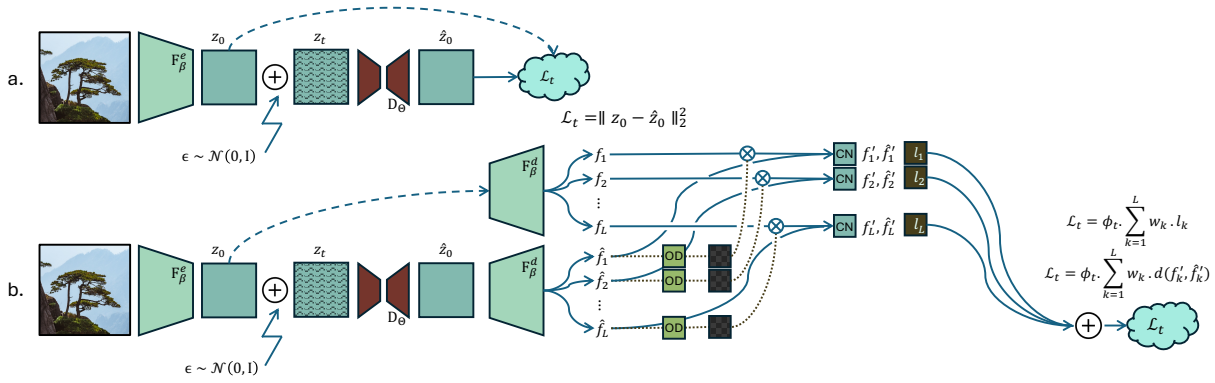


Figure 2 Overview of our approach. (a) Latent diffusion models compare clean latents and the predicted latents. (b) Our LPL acts in the features of the autoencoder's decoder effectively aligning the diffusion process with the decoder. F^e_β, F^d_β*: autoencoder encoder and decoder,* D_Θ*: denoiser network, CN: cross normalization layer, OD: outlier detection.*

We note that the presence of ℓ_2 in the LDM objective has some important implications. First, the ℓ_2 norm treats all pixels in the latents as equally important and disregards the downstream structure induced by the decoder whose objective is to reconstruct the image from its latents. This is problematic because the autoencoder's latent space has a highly irregular structure and is not equally influenced by the different pixels in the latent code. Thus, optimizing the ℓ_2 distance in the diffusion model latent space could be different from optimizing the perceptual distance between images. Second, while an ℓ_2 objective is theoretically justified in the original DDPM formulation, generative models trained with an ℓ_2 reconstruction objective have been observed to produce blurry images, as is the case *e.g.* for VAE models (Kingma and Welling, 2014).

The problem of blurry images due to ℓ_2 reconstruction losses has been addressed through the use of perceptual losses such as LPIPS (Zhang et al., 2018), which provide a significant boost to the image quality in settings such as autoencoding (Esser et al., 2021), super-resolution Ledig et al. (2017) and image-to-image generative models (Isola et al., 2017; Park et al., 2019). However, in the case of LDM, a perceptual loss cannot be used directly on the predicted latents, instead the latents would need to be decoded to an RGB image and then entered into a feature-extraction network – introducing significant overhead in terms of memory and computation. To avoid such overheads, we develop an alternative perceptual loss that operates directly on the feature space of the decoder.

3.2 Latent perceptual loss

We propose a perceptual loss that operates on the features of the latent-to-RGB decoder, F^d_β, extracted at different depths in the decoder. Let $z_t = a_t z_0 + b_t \epsilon_t$ be a noisy sample in the diffusion model latent space at time t, and $\hat{z}_0 = (z_t - b_t D(z_t, \sigma_t; \Theta))/a_t$ the corresponding estimated noise-free latent at time $t = 0$. We compute two sets of hierarchies of L decoder features, $\{\phi_l\}_{l=1}^L$, and, $\{\hat{\phi}_l\}_{l=1}^L$, by decoding both the original, z_0, and estimated latents, $\hat{z}_0$ (where for brevity we drop the dependence of $\hat{z}_0$ on t):

$$\begin{cases} \phi_1, ..., \phi_L = \left(F^{d,l}_\beta(z_0)\right)_{l \in [\![1,L]\!]}, \\ \hat{\phi}_1, ..., \hat{\phi}_L = \left(F^{d,l}_\beta(\hat{z}_0)\right)_{l \in [\![1,L]\!]}. \end{cases} \tag{1}$$

Using these intermediate features, we can define our training objective. Our LPL, $\mathcal{L}_{LPL}$, is a weighted sum of the quadratic distances between the feature representations at the different decoding scales, obtained after normalization:

$$\mathcal{L}_{\mathrm{LPL}} = \mathbb{E}_{t \in \mathcal{T}, \epsilon \sim \mathcal{N}(0,I), x_0 \in D_{\mathcal{X}}} \left[\delta_{\sigma_t \leq \tau_\sigma} \sum_{l=1}^{L} \frac{\omega_l}{C_l} \sum_{c=1}^{C_l} \left\| \rho_{l,c}(\hat{\phi}_{l,c}) \odot \left(\phi'_{l,c} - \hat{\phi}'_{l,c}\right) \right\|_2^2 \right]. \tag{2}$$

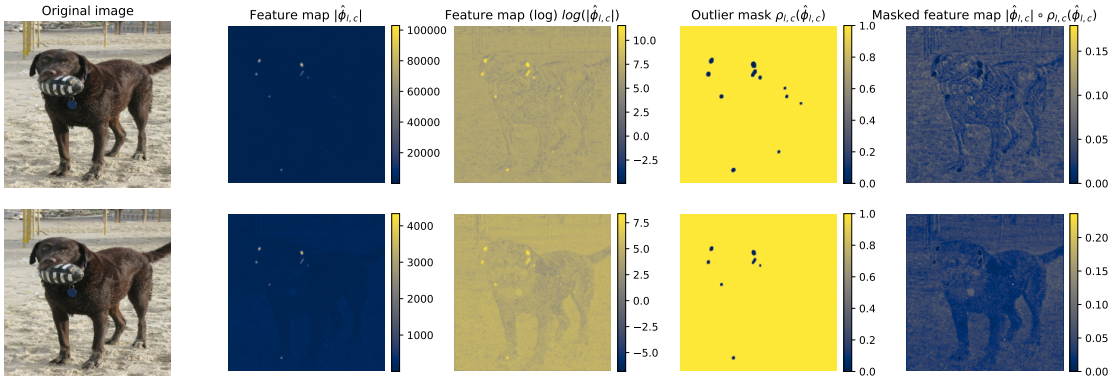


Figure 3 Example of feature maps from the autoencoder's decoder. The presence of outliers makes the underlying feature representation difficult to exploit. l refers to the block index, while c is the channel index within the block. Top row: $l = 4, c = 2$, bottom row: $l = 4, c = 8$.

Framework	a_t	b_t	$\hat{x}_0$
DDPM-ϵ_t	$1/\sqrt{1+\sigma_t^2}$	$\sigma_t/\sqrt{1+\sigma_t^2}$	$\sqrt{1+\sigma_t^2}x_t - \sigma_t D(x_t, t; \Theta)$
DDPM-v_θ	$1/\sqrt{1+\sigma_t^2}$	$\sigma_t/\sqrt{1+\sigma_t^2}$	$\frac{1}{\sqrt{\sigma_t^2+1}}(x_t - \sigma_t D(x_t, t; \Theta))$
Flow-OT	$1-t$	t	$(x_t - tD(x_t, t; \Theta))/(1-t)$

Table 1 Summary of the formula for the estimate of the clean image corresponding to the different formulations. Using the following parameterization, $\forall t, x_t = a_t x_0 + b_t \epsilon_t$.

where ϕ'_l is the standardized version of ϕ_l across the channel dimension, $\rho_{l,c}(\hat{\phi}_{l,c})$ is a binary mask masking the detected outliers in the feature map $\hat{\phi}_{l,c}$, ω_l is a depth-specific weighting and C_l the channel dimensionality of the feature tensor. Note that we better explain these terms later in this section. Moreover, to reduce both the LPL computational complexity and memory overhead, we only apply our loss for high signal-to-noise ratios (SNR). In particular, we impose a hard threshold τ_σ and only apply the loss if the SNR is higher than it $\delta_{\sigma_t \leq \tau_\sigma}(\sigma_t)$.
The LPL loss is applied in conjunction with the standard diffusion loss, resulting in the following training objective.

$$\mathcal{L}_{\text{tot}} = \mathcal{L}_{\text{Diff}} + w_{\text{LPL}} \cdot \mathcal{L}_{\text{LPL}} \tag{3}$$

Depth-specific weighting. Empirically, we find the loss amplitude at different decoder layers to differ significantly – it grows with a factor of two when considering layers with a factor two increase resolution. To balance the contributions from different decoder layers, we therefore weight them by the inverse of the upscaling factor *w.r.t.* the first layer, *i.e.* $\omega_l = 2^{-r_l/r_1}$ where r_l is the resolution of the l-th layer.

Outlier detection. When inspecting the decoder features we find artefacts at decoder's deeper layers. Particularly, in some cases a small number of decoder activations have very high absolute values, see Figure 3. This is undesirable, as such outliers can dominate the perceptual loss, reducing its effectiveness. To prevent this, we use a simple outlier detection algorithm to mask them when computing the perceptual loss. See the supplementary material for details.

Normalization. Since the features in the decoder can have significantly varying statistics from each other, we follow Zhang et al. (2018) and normalize them per channel so that the features in every channel in every layer are zero mean and have unit variance. However, normalizing the feature maps corresponding to the original and denoised latents with different statistics can induce nonzero gradients even when the absolute value has been correctly predicted. To obtain a coherent normalization, we use the feature statistics from the denoised latents to normalize both tensors.

3.3 Generalization for Latent Generative Modeling

While the bulk of our experiments have been conducted on models trained under DDPM (Ho et al., 2020) for noise prediction, we can generalize our method to different frameworks such as diffusion with velocity

prediction (Salimans and Ho, 2022) and flow matching (Lipman et al., 2023). To do this, the only requirement is to be able to estimate the original latents from the model predictions. Under general frameworks such as DDPM and flows, we can write the forward equation in the form $\forall t, x_t = a_t x_0 + b_t \epsilon_t$, where the different paradigms only differ in terms of the parameterization of a_t and b_t. In Table 1, we provide a summary for these different formulations.

4 Experimental evaluation

In this section, we first present our experimental setup, and then go on to present our main results, as well as qualitative results and a number of ablation studies.

4.1 Experimental setup

Datasets. We conduct an extensive evaluation on three datasets of different scales and distributions: ImageNet-1k (Deng et al., 2009), CC12M (Changpinyo et al., 2021), and S320M: a large internal dataset of 320M stock images. We note that for both ImageNet-1k and CC12M, human faces were blurred to avoid training models on identifiable personal data. For both CC12M and S320M, we recaption the images using Florence-2 (Xiao et al., 2024) to obtain captions that more accurately describe the image content. For each of these datasets, we conduct evaluations at both 256×256 and 512×512 image resolution.

Architectures. All experiments are performed using the Multi-modal DiT architecture from Esser et al. (2024). We downscale the model size to be similar to Pixart-α (Chen et al., 2024) and DiT-XL/2 (Peebles and Xie, 2023), which corresponds to 28 blocks with a hidden size of 1,536, amounting to a total of 796M parameters. For ImageNet-1k models we condition on class labels, while for the other datasets we condition on text prompts. For our main results, we perform our experiments using the asymetric autoencoder from Zhu et al. (2023). For ablation studies, we revert to the lighter autoencoder from SDXL (Podell et al., 2024).

Training and sampling. Our training and sampling methodologies are largely based on Berrada et al. (2024a). Unless specified otherwise, we follow the DDPM-ϵ training paradigm (Ho et al., 2020), using the DDIM (Song et al., 2021) algorithm with 50 steps for sampling and a classifier-free guidance scale of $\lambda = 2.0$ (Ho and Salimans, 2021). Following Podell et al. (2024), we use a quadratic scheduler with $\beta_{\text{start}} = 0.00085$ and $\beta_{\text{end}} = 0.012$. For the flow experiments, we use the conditional OT probability path (Lipman et al., 2023) with the mode sampling with heavy tails paradigm from Esser et al. (2024). Under this paradigm, the model is trained for velocity prediction and evaluated using DDIM sampler.

Similar to Chen et al. (2024), we pre-train all models at 256 resolution on the dataset of interest for 600k iterations. We then enter a second phase of training, in which we optionally apply our perceptual loss, which lasts for 200k iterations for 256 resolution models and for 120k iterations for models at 512 resolution.

When changing the resolution of the images, the resolution of the latents changes by the same factor, keeping the same noise threshold τ_σ yields inconsistent results across resolutions. To ensure consistent behavior, we follow Esser et al. (2024), and scale the noise threshold similarly to how the noise schedule is scaled in order to keep the same uncertainty per patch. In practice, this amounts to scaling the threshold by the upscaling factor. The kernel sizes for the morphological operations in the outlier detection algorithm are also scaled to cover the same proportion of the image.

Metrics. To evaluate our models, we report results in terms of FID (Heusel et al., 2017) to assess image quality and to what extent the generated images match the distribution of training images, and CLIPScore (Hessel et al., 2021) to assess the alignment between the prompt and the generated image for text-conditioned models. In addition, we report distributional metrics precision and recall (Sajjadi et al., 2018) as well as density and coverage (Naeem et al., 2020) to better understand effects on image quality (precision/density) and diversity (recall/coverage). We evaluate metrics with respect to ImageNet-1k and, for models trained on CC12M and S320M, the validation set of CC12M. For FID and other distributional metrics, we use the evaluation datasets as the reference datasets and compare an equal number of synthetic samples. For CLIPScore, we use the prompts of the evaluation datasets and the corresponding synthetic samples. Following previous works (Rombach et al., 2022; Peebles and Xie, 2023), we use a guidance scale of 1.5 for resolutions of 256 and 2.0 for resolutions of 512, which we also found to be optimal for our baseline models trained without LPL.

Figure 4 Samples from models trained with and without our latent perceptual loss on S320M. Samples from the model with perceptual loss (bottom row) show more realistic textures and details.

	Pre-training		Post-training			Results	
	Res.	*Iters*	*Res.*	*Iters*	*LPL*	*FID* (↓)	*CLIP* (↑)
ImageNet-1k	256	600k	256	200k	✗	2.98	—
					✓	**2.72**	—
			512	120k	✗	4.88	—
					✓	**3.79**	—
CC12M	256	600k	256	200k	✗	7.81	25.06
					✓	**6.22**	**25.12**
			512	120k	✗	8.79	24.88
					✓	**7.27**	**25.12**
S320M	256	600k	512	120k	✗	8.81	24.39
					✓	**8.30**	**24.41**

Table 2 Impact of our perceptual loss for models trained on different datasets and resolutions for DDPM-ϵ model. Using LPL boosts FID and CLIP score for all datasets and resolutions considered.

4.2 Main results

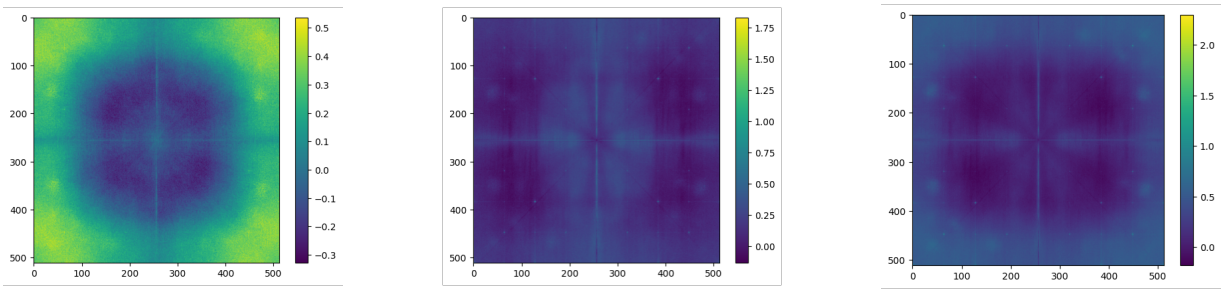
LPL applied across different datasets. In Table 2 we consider the impact of the LPL on the FID and CLIPScore for models trained on the three datasets and two resolutions. We observe that the LPL loss consistently improves both metrics across all three datasets. Most notably, FID is improved by 0.91 points on ImageNet-1k at 512 resolution and by 1.52 points on CC12M at 512 resolution. The CLIPScore is also improved for both resolutions on CC12M, by 0.06 and 0.24 points respectively. Similarly, for the S320M dataset, we observe that FID is improved by 0.51 points while CLIP score improves (marginally) by 0.02 points. Samples of models trained with and without LPL on CC12M and S320M are shown in Figure 1 and Figure 4, respectively.

Generalization to other frameworks. We showcase the generality of the LPL by applying it to different generative models, experimenting with DDPM for both epsilon and velocity prediction, and flow matching with optimal-transport (OT) path similar to Esser et al. (2024). In Table 3 we report experimental results for models trained on ImageNet-1k at 512 resolution. The DDPM-based models perform very similar (except perhaps for FID, where they differ by 0.16 points), and we we find significant improvements across all metrics other than density when using LPL. Density remains similar to the baseline for DDPM models, but improves from 1.14 to 1.29 for the Flow-OT model, where all metrics are improved relative to the DDPM trained ones. We posit that this is due to the mode sampling scheme in (Esser et al., 2024), which emphasizes middle timesteps that could better control the trajectories of the flow path towards having more diversity and not improving the quality (precision/density). Hence, applying LPL to Flow-OT solve this by considerably boosting quality. Notably, considering DDPM baselines, LPL provides a boost as significant as the one provided by using flow matching (scores of DDPM w/ LPL in 2nd and 4th columns are on par or better than Flow-OT w/o LPL in 5th column). Moreover, the provided boost is orthogonal to the training paradigm, leading to overall best results when using LPL with the flow model.

Frequency analysis. While the metrics above provide useful information on model performance, they do not specifically provide insights in terms of frequencies at which using LPL is more effective at modeling data

Paradigm	DDPM-ϵ		DDPM-v_Θ		Flow-OT	
LPL	✗	✓	✗	✓	✗	✓
FID (↓)	4.88	**3.79**	4.72	**3.84**	4.54	**3.61**
Coverage (↑)	0.80	**0.82**	0.80	**0.83**	0.82	**0.85**
Density (↑)	**1.14**	1.13	**1.15**	1.14	1.14	**1.29**
Precision (↑)	0.74	**0.77**	0.73	**0.78**	0.75	**0.79**
Recall (↑)	0.49	**0.51**	0.49	**0.50**	0.52	**0.54**

Table 3 Effect of the LPL on ImageNet-1k models at 512 resolution trained with different methods. We observe consistent improvements on all metrics when incorporating the LPL, except for density metric for which we observe a very slight degradation when using DDPM training.



(a) $\mathcal{F}(D_{\text{LPL}}) - \mathcal{F}(D_{\text{base}})$ **(b)** $\mathcal{F}(D_{\text{base}}) - \mathcal{F}(D_{\text{real}})$ **(c)** $\mathcal{F}(D_{\text{LPL}}) - \mathcal{F}(D_{\text{real}})$

Figure 5 Power spectrum of real and generated images. Difference in (log) power spectrum between image generated with and without LPL. Using LPL strenghtens frequencies at the extremes (very low and very high).

than the baseline. To provide insight on the effect of our perceptual loss *w.r.t.* the frequency content of the generated images, we compare the power spectrum profile of images generated with a model trained with and without LPL on CC12M at 512 resolution as well as a set of real images from the validation set.

In Figure 5, we plot the difference between log-power spectra between the three image sets. The left-most panel clearly shows the presence of more high frequency signal in the generated images when using LPL to train the model, confirming what has been observed in the qualitative examples of Figure 1 and Figure 4. Moreover, the very lowest frequencies are also strengthened in the samples of the model with LPL. We posit that using the LPL makes it easier to match very low frequencies as they tend be encoded separately in certain channels of the decoder. In Figure 6, we report the error when comparing the power-spectrum of synthetic images and real images, averaged across the validation set of CC12M. For this, we compute the average of the power spectrum across a set of 10k synthetic images from each model and the reference images for the validation set of CC12M. Our experiments indicate that the model trained with LPL is consistently more accurate in modeling high frequencies ($f > 150$), at the expense of a somewhat larger error at middle frequencies ($75 < f < 150$).

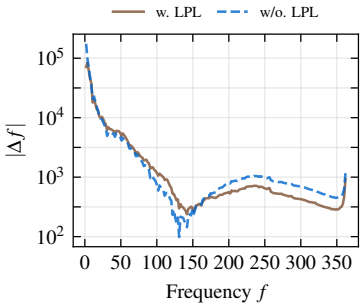


Figure 6 Frequency comparison. We compare the power spectrum of the images obtained with or without LPL with real reference images from the validation set of CC12M.

4.3 Ablation results

LPL depth. Using decoder layers to compute our perceptual loss comes with increased computational and memory costs. We therefore study the effect of computing our perceptual loss using only a subset of the decoder layers, as well as a baseline using the RGB pixel output of the decoder. We progressively add more decoder features, so the model with five blocks contains features from the first up to the fifth block. From the results in Figure 8, we find that earlier blocks do not significantly improve FID - and can even negatively impact performance. Deeper layers, on the other hand, significantly improve the performance. The most significant gains are obtained when incorporating the third and fourth decoder layers which both improve the FID by more than one point *w.r.t.* the model incorporating one block less. Finally, the last decoder layer improves the FID only marginally and could be omitted to reduce resource consumption. We perform an additional ablation where the loss operates directly on the RGB image space (without using internal decoder

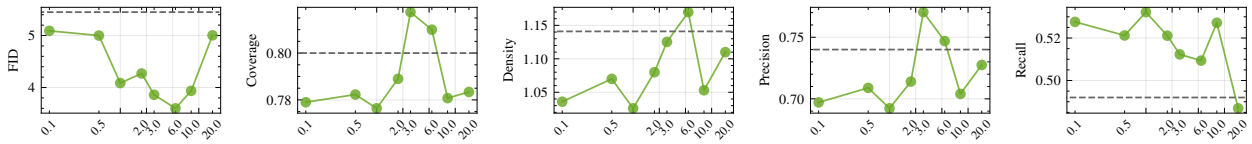


Figure 7 Ablation study on the impact of the noise threshold τ_σ. We report FID, coverage, density, precision and recall. The dashed line corresponds to the baseline without LPL, note the logarithmic scaling of the noise threshold on the horizontal axis.

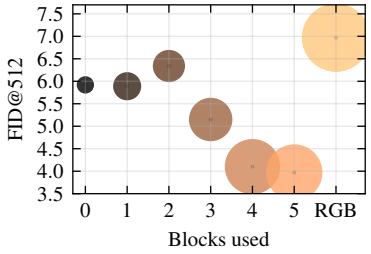


Figure 8 Exploration of LPL depth. Influence of decoder blocks used in LPL on FID, zero corresponds to not using LPL. Disk radius shows GPU memory usage: w/o LPL=64.9 GB, LPL 5 blocks=83.4 GB.

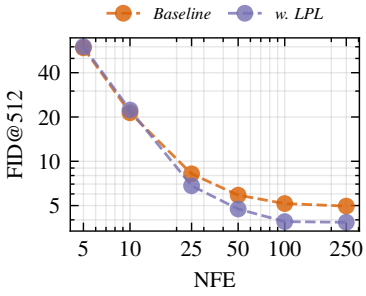


Figure 9 Impact of LPL for different number of sampling steps. With higher numbers of sampling steps, the difference between the baseline and the model trained with LPL increases.

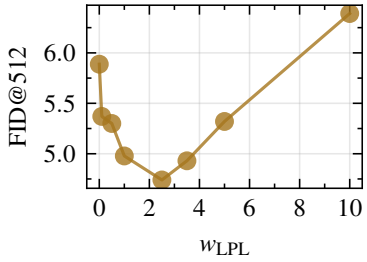


Figure 10 Influence of the LPL loss weight on model performance. The curve shows a sharp decrease in FID before going back up for larger weights.

features), which results in degraded performance compared to the baseline while inducing a considerable memory overhead.

Feature normalization. Before computing our perceptual loss, we normalize the decoder features. We compared normalizing the features of the original latent and the predicted one separately, or normalizing both using the statistics from the predicted latent. Our experiment is conducted on ImageNet-1k at 512 resolution. While the model trained with separately normalized latents results in a slight boost of FID (4.79 *vs*. 4.88 for the baseline w/o LPL), the model trained with shared normalization statistics leads to a much more significant improvement and obtains an FID of 3.79.

SNR threshold value. We conduct an experiment on the influence of the SNR threshold which determines at which time steps our perceptual loss is used for training. Lower threshold values correspond to using LPL for fewer iterations that are closest to the noise-free targets. We report results across several metrics in Figure 7 and illustrated with qualitative examples in the supplementary material in Figure 13. We find improved performance over the baseline without LPL for all metrics and that the best values for each metric are obtained for a threshold between three and six, except for the recall which is very stable (and better than the baseline) for all threshold values under 20.

Reweighting strategy. We compare the performance when using uniform or depth-specific weights to combine the contributions from different decoder layers in the LPL. We find that using depth-specific weights results in significant improvements in terms of image quality *w.r.t.* using uniform weights. While the depth-specific weights achieve an FID of 3.79, the FID obtained using uniform weights is 4.38. Hence, while both strategies improve image quality over the baseline (which achieves an FID of 4.88), reweighting the layer contributions to be approximately similar further boosts performance and improves FID by 0.59 points.

LPL and convergence. As the LPL loss adds a non-negligible memory overhead, by having to evaluate and backpropagate through the latent decoder, it is interesting to explore at which point in training it should be introduced. We train models on ImageNet-1k at 512 resolution with different durations of the post-training stage. We use an initial post-training phase — of zero, 50k, or 400k iterations — in which LPL is not used, followed by another 120k iterations in which we either apply LPL or not.

Initial post-train iters	0	50k	400k
Δ FID (↓)	−0.58	−0.78	−0.97
Δ coverage (↑)	+4.29	+3.51	+3.99
Δ density (↑)	+0.14	+0.12	+0.21
Δ precision (↑)	+4.01	+4.55	+5.89
Δ recall (↑)	+1.99	+2.32	+4.22

Table 4 Effect of our perceptual loss on models pre-trained without LPL for a set number of iterations. In each column, we report the difference in metrics after post-training for 120k iterations with or without LPL. All metrics improve when adding LPL in the post-training phase.

The results in Table 4 indicate that in each case LPL improves all metrics and that the improvements are larger when the model has been trained longer and is closer to convergence (except for the coverage metric where we see the largest improvement when post-training for only 120k iterations). This suggests that better models (ones trained for longer) benefit more from our perceptual loss.

Influence on sampling efficiency. We conduct an experiment to assess the influence of the perceptual loss on the sampling efficiency. To this end, we sample the ImageNet@512 model with different numbers of function evaluations (NFE) then check the trends for the baseline and the model trained with our method. For this experiment, we use DDIM algorithm. Results are reported on Figure 9, where we find that for very low numbers of function evaluations, both models perform similarly. The improvement gains from the LPL loss start becoming considerable after 25 NFEs, where we observe a steady increase in performance gains with respect to the number of function evaluations up to 100, afterwards both models stabilize at a point where the model trained using LPL achieves an improvement of approximately 1.1 points over the baseline.

Impact on EMA. Since the LPL has the effect of increasing the accuracy of the estimated latent during every timestep, it reduces fluctuations between successive iterations of the model during training. Consequently, when training with LPL the EMA momentum can be reduced to obtain optimal performance. In Figure 11 we report the results of a grid search over the momentum parameter γ_{EMA}. We find that the model trained with LPL achieves better results when using a slightly lower momentum than the baseline. From the graph, it's clear that better FID is obtained closer to $\gamma_{\text{EMA}} = 0.99975$ for the LPL model, which corresponds to a half-life of approximately 2750 iterations, while the non-LPL model achieves its optimal score at $\gamma_{\text{EMA}} = 0.9999$ corresponding to a half life of 6930 iterations, more than twice as much as the LPL model, thereby validating our hypothesis.

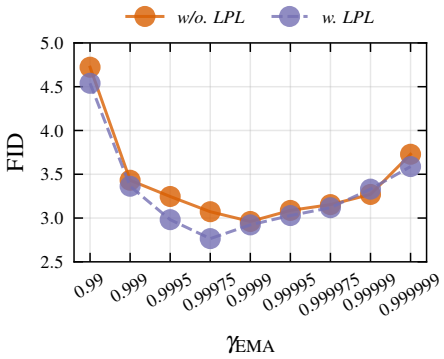


Figure 11 Impact of EMA decay rate. Training with LPL is more stable, and allows for a smaller decay parameter.

Relative weight. We conduct a grid search over different values for the weight of the LPL loss w_{LPL}. We report FID after training for 120k iterations at 512 resolution, all models are initialized from the same 256 pretrained checkpoint. Our results are reported in Figure 10. Introducing LPL sharply decreases FID for lower weights before going back up at higher weights. We find the model to achieve the best FID for $w_{LPL} \approx 3.0$ which roughly corresponds to a fifth of the relative contribution to the total loss.

5 Conclusion

In this work, we identified a disconnect between the decoder and the training of latent diffusion models, where the diffusion model loss does not receive any feedback from the decoder resulting in perceptually non-optimal generations that oftentimes lack high frequency details. To alleviate this disconnect we introduced a latent perceptual loss (LPL) that provides perceptual feedback from the autoencoder's decoder when training the generative model. Our quantitative results showed that the LPL is generalizable and improves performance for models trained on a variety of datasets, image resolutions, as well as generative model formulations. We observe that our loss leads to improvements from 6% up to 20% in terms of FID. Our qualitative analysis show that the introduction of LPL leads to models that produce images with better structural consistency and sharper details compared to the baseline training. Given its generality, we hope that our work will play an important role in improving the quality of future latent generative models.

References

Dan Amir and Yair Weiss. Understanding and simplifying perceptual distances. In *CVPR*, 2021.

Jie An, Zhengyuan Yang, Jianfeng Wang, Linjie Li, Zicheng Liu, Lijuan Wang, and Jiebo Luo. Bring metric functions into diffusion models. In *IJCAI*, 2024.

Pietro Astolfi, Marlene Careil, Melissa Hall, Oscar Mañas, Matthew Muckley, Jakob Verbeek, Adriana Romero Soriano, and Michal Drozdzal. Consistency-diversity-realism Pareto fronts of conditional image generative models. *arXiv preprint*, 2406.10429, 2024.

Tariq Berrada, Pietro Astolfi, Melissa Hall, Reyhane Askari-Hemmat, Yohann Benchetrit, Marton Havasi, Matthew Muckley, Karteek Alahari, Adriana Romero-Soriano, Jakob Verbeek, and Michal Drozdzal. On improved conditioning mechanisms and pre-training strategies for diffusion models, 2024a. https://arxiv.org/abs/2411.03177.

Tariq Berrada, Jakob Verbeek, Camille Couprie, and Karteek Alahari. Unlocking pre-trained image backbones for semantic image synthesis. In *CVPR*, 2024b.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019.

Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut. Conceptual 12M: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In *CVPR*, 2021.

Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In *ICLR*, 2024.

Lenaic Chizat and Francis Bach. A Note on Lazy Training in Supervised Differentiable Programming. working paper or preprint, December 2018. https://inria.hal.science/hal-01945578.

Jooyoung Choi, Jungbeom Lee, Chaehun Shin, Sungwon Kim, Hyunwoo Kim, and Sungroh Yoon. Perception prioritized training of diffusion models. In *CVPR*, 2022.

Steffen Czolbe, Oswin Krause, Ingemar J. Cox, and Christian Igel. A loss function for generative neural networks based on Watson's perceptual model. In *NeurIPS*, 2020.

Xiaoliang Dai, Ji Hou, Chih-Yao Ma, Sam Tsai, Jialiang Wang, Rui Wang, Peizhao Zhang, Simon Vandenhende, Xiaofang Wang, Abhimanyu Dubey, Matthew Yu, Abhishek Kadian, Filip Radenovic, Dhruv Mahajan, Kunpeng Li, Yue Zhao, Vladan Petrovic, Mitesh Kumar Singh, Simran Motwani, Yi Wen, Yiwen Song, Roshan Sumbaly, Vignesh Ramanathan, Zijian He, Peter Vajda, and Devi Parikh. Emu: Enhancing image generation models using photogenic needles in a haystack. *arXiv preprint*, 2309.15807, 2023.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat GANs on image synthesis. In *NeurIPS*, 2021.

Patrick Esser, Robin Rombach, and B. Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024.

Shanghua Gao, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. Masked diffusion transformer is a strong image synthesizer. In *ICCV*, 2023.

Tiankai Hang and Shuyang Gu. Improved noise schedule for diffusion training. *arXiv preprint*, 2407.03297, 2024.

Tiankai Hang, Shuyang Gu, Chen Li, Jianmin Bao, Dong Chen, Han Hu, Xin Geng, and Baining Guo. Efficient diffusion training via Min-SNR weighting strategy. In *ICCV*, 2023.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: A reference-free evaluation metric for image captioning. In *EMNLP*, 2021.

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. In *NeurIPS*, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

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

Jonathan Ho, Chitwan Saharia, William Chan, David J. Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47), 2022a.

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

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*, 2018.

Younghyun Jo, Sejong Yang, and Seon Joo Kim. Investigating loss functions for extreme super-resolution. In *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

Minguk Kang, Richard Zhang, Connelly Barnes, Sylvain Paris, Suha Kwak, Jaesik Park, Eli Shechtman, Jun-Yan Zhu, and Taesung Park. Distilling diffusion models into conditional GANs. In *ECCV*, 2024.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *CVPR*, 2020.

Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *NeurIPS*, 2021.

Diederik Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *NeurIPS*, 2019.

Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.

Mark Levy, Bruno Di Giorgi, Floris Weers, Angelos Katharopoulos, and Tom Nickson. Controllable music production with diffusion models and guidance gradients. In *NeurIPS Workshop on Diffusion Models*, 2023.

Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed. In *WACV*, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICML*, 2023.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *ICLR*, 2022.

Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *ICML*, 2020.

Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *ICLR*, 2024.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lučić, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. In *NeurIPS*, 2018.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *ICLR*, 2022.

Robin San Roman, Yossi Adi, Antoine Deleforge, Romain Serizel, Gabriel Synnaeve, and Alexandre Defossez. From discrete tokens to high-fidelity audio using multi-band diffusion. In *NeurIPS*, 2023.

Edgar Schönfeld, Vadim Sushko, Dan Zhang, Juergen Gall, Bernt Schiele, and Anna Khoreva. You only need adversarial supervision for semantic image synthesis. In *ICLR*, 2021.

Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-a-video: Text-to-video generation without text-video data. In *ICLR*, 2023.

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

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *NeurIPS*, 2019.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *ICML*, 2023.

Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *WACV*, 2022.

Lokesh Veeramacheneni, Moritz Wolter, Hildegard Kuehne, and Juergen Gall. Fréchet wavelet distance: A domain-agnostic metric for image generation. *arXiv preprint*, 2312.15289, 2023.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7): 1661–1674, 2011.

Tong Wu, Zhihao Fan, Xiao Liu, Yeyun Gong, Yelong Shen, Jian Jiao, Hai-Tao Zheng, Juntao Li, Zhongyu Wei, Jian Guo, Nan Duan, and Weizhu Chen. AR-Diffusion: Auto-regressive diffusion model for text generation. In *NeurIPS*, 2023.

Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu Yuan. Florence-2: Advancing a unified representation for a variety of vision tasks. In *CVPR*, 2024.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Zixin Zhu, Xuelu Feng, Dongdong Chen, Jianmin Bao, Le Wang, Yinpeng Chen, Lu Yuan, and Gang Hua. Designing a better asymmetric VQGAN for StableDiffusion. *arXiv preprint*, 2306.04632, 2023.

Appendix

A Relevance of the LPL loss

Interpreting LPL. Under the DDPM paradigm, for latent diffusion, a neural network is trained to model the reverse process $q(z_{t-1}|z_t)$. Under this setting, training is conducted by optimizing the KL divergence between the true reverse process and the predictor that is modeled using a neural network:

$$\mathcal{L}_{t-1} = \mathbb{E}_q \left[D_{\mathrm{KL}} \left(q(z_{t-1}|z_t, z_0) \parallel p_\Theta(z_{t-1}|z_t) \right) \right]. \tag{4}$$

After simplification Ho et al. (2020), the training loss resembles the denoising score matching objective Song and Ermon (2019) over multiple noise scales indexed by timestep t:

$$\mathbb{E}_{x_0,\epsilon,t} \left[\gamma_t \left\| D(z_t, \sigma_t; \Theta) - z_0 \right\|^2 \right], \tag{5}$$

where γ_t is a time-dependent weighting factor. Taking into account the global objective which is image generation means *putting more emphasis on obtaining ℓ_2-optimal reconstructions in image space rather than in latent space*. Such a constraint can be imposed in the form of a penalty term that is added to the training objective:

$$\mathcal{L}_{t-1}^{\mathrm{pen}} = \mathbb{E}_q \left[D_{\mathrm{KL}} \left(p_F(q(x_{t-1}|x_t, x_0)) \parallel p_F(F_\beta^d(p_\Theta(z_{t-1}|z_t))) \right) \right], \tag{6}$$

where p_F is a projector that maps from image space to a suitable embedding space in which to compare the images. We can assume that both $p_F\left(q\left(z_{t-1}|z_t, z_0\right)\right)$ and $p_F\left(F_\beta^d\left(p_\Theta\left(z_{t-1}|z_t\right)\right)\right)$ map to Gaussian distributions with constant variance — such an approximation can be considered reasonable when considering small enough time discretization, assuming that the projector is locally linear around z_t (the local linearity assumption has been studied previously in the litterature Jacot et al. (2018); Chizat and Bach (2018)) — Under these conditions, we can approximate the divergence term in the penalty as $\propto \mathbb{E}_{x_0,t,\epsilon} \left[\left\| F_\beta^{d+} \circ D(z_t, \sigma_t; \Theta) - F_\beta^{d+} \circ F_\beta^e(x_0) \right\|^2 \right]$ where F_β^{d+} is the feature projector of the decoder which outputs the intermediate features from each block of the autoencoder's decoder. This shows that under certain conditions, taking into account the structure of the latent space is akin to matching intermediate feature representations in the process of image decoding.

B Latent Structure

Because of the underlying structure of the latent space, certain errors can have much more detrimental effects to the quality of the decoded image than others. We illustrate this in Figure 12 by comparing the generated image after interpolating the encoded latents to different resolutions then back to its original resolution before decoding them. While these different transformations yield similar errors in terms of MSE, especially in RGB space, the interpolation algorithm becomes crucial when working in the latent space.

An illustration of this effect is presented in Figure 12 where we degrade the quality of the latents by performing an interpolation operation to downsize the latents (when $s < 1$) followed by the reverse operation to recover latents at the original size, such a transformation can be seen as a form of lossy compression where different interpolation methods induce different biases in the information lost.

By examining the reconstructions from the latents, we cannot conclude that there is a direct relationship between the MSE with respect to the original latent and the decoded image quality. While nearest interpolation results in the highest MSE, the reconstructed images are more perceptually similar to the target than the ones obtained with bilinear interpolation. Similarly, while the bicubic interpolation with $s = 1.3$ achieves an MSE of 2.38, it still results in better reconstruction than the bilinear interpolation where $s = 2.0$ which achieves a lower MSE error of 1.88.

From this analysis, we see that certain kinds of errors can have more or less detrimental effects on the image generation, which go beyond simple MSE in the latent space.

Figure 12 Influence of interpolation artefacts on latent reconstruction. We downscale the image by a factor of $1/s$ before upscaling back to recover the original resolution. *From top to bottom:* bilinear interpolation in pixel space, nearest in latent space, bilinear in latent space and bicubic interpolation in latent space.

C Outlier Detection

At deeper layers of the autoencoder, some layers have aretfacts where small patches in the feature maps have a norm orders of magnitude higher than the rest of the feature map. These aretefacts have been detected consistently when testing the different opensource autoencoders available online, which include the ones used in our experiments[1], as well as others.[2]

To ensure easy adaptability to different models, we propose a simple detection algorithm for these patches and mask them when computing the loss and normalizing the feature maps. Our algorithm is based on simple heuristics and is not meant to provide a state-of-the-art solution for outlier detection. Rather, it is proposed as a temporary patch for the observed issues, while the long-term solution would be to train better autoencoders that do not suffer from these outliers.

Detection algorithm. We empirically observe that the activations for every feature map approximately follow a normal distribution, while the outliers can be identified as a small subset of out-of-distribution points. To identify them, we threshold the points with the corresponding percentile at δ_o and $1 - \delta_o$ percentiles. Since computing the quantiles can be computationally expensive during training, we do it using nearest interpolation, which amounts to finding the k-th largest value in every feature map where $k = \delta_o \times H_f \times W_f$ (or $k = (1 - \delta_o) \times H_f \times W_f$ for the maximal values). To remove small false positives that persist in the outlier mask, we apply a morphological opening, which can be seen as an erosion followed by a dilation of the feature map. Pseudo-code for the outlier detection algorithm is provided in Alg. 1.

[1] https://huggingface.co/stabilityai/sdxl-vae, and https://huggingface.co/cross-attention/asymmetric-autoencoder-kl-x-1-5

[2] https://huggingface.co/CompVis/stable-diffusion-v1-4, and https://huggingface.co/cross-attention/asymmetric-autoencoder-kl-x-2.

```
def remove_outliers(features, down_f=1, opening=5, closing=3, m=100, quant=0.02):
    opening = int(ceil(opening/down_f))
    closing = int(ceil(closing/down_f))

    if opening == 2:
        opening = 3
    if closing ==2:
        closing = 1

    # replace quantile with kth (nearest interpolation).
    feat_flat = features.flatten(-2, -1)

    k1 = int(feat_flat.shape[-1]*quant)
    k2 = int(feat_flat.shape[-1]*(1-quant))

    q1 = feat_flat.kthvalue(k1, dim=-1).values[..., None, None]
    q2 = feat_flat.kthvalue(k2, dim=-1).values[..., None, None]

    # Mask obtained by thresholding at the upper quantiles.
    m = 2*feat_flat.std(-1)[..., None, None].detach()
    mask = (q1-m < features)*(features < q2+m)

    # dilate the mask.
    mask=MaxPool2d(
            kernel_size=closing,
            stride=1,
            padding=(closing-1)//2
        )(mask.float()) # closing

    mask=(-MaxPool2d(
            kernel_size=opening,
            stride=1,
            padding=(opening-1)//2
        )(-mask)).bool() # opening

    features = features * mask
    return mask, features
```

Algorithm 1 Outlier detection algorithm. The algorithm works by setting a threshold according to the upper 0.02 quantile of the activations in the feature map. Because the outliers are orders of magnitude away from the rest, we shift the threshold by an offset m that guarantees that only the outliers are thresholded while no activations are masked when no outliers are present. Subsequently, we smooth out the predicted mask using a dilation operation that eliminates small noise in the mask.

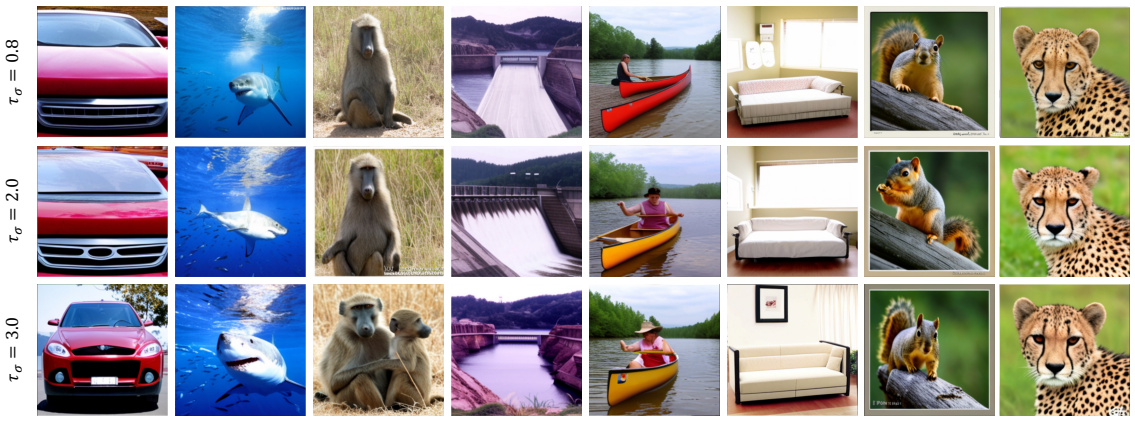


Figure 13 Influence of noise threshold. Higher thresholds allow for more detailed and coherent images. Samples obtained from a model trained on ImageNet@256.

D Additional qualitative results

Noise threshold. In Figure 13, we illustrate the impact of using a higher noise threshold (which amounts to using LPL for longer in the diffusion chain) on the image quality. A higher noise threshold yields better structures in the images and exacerbates semantic features that distinguish objects.

Vanilla diffusion. In Figure 14, we qualitatively investigate the influence of LPL on a baseline model, without classifier-free guidance and without EMA. We can see that LPL significantly improves the structure of objects as compared to the model that was trained without it.

Figure 14 Qualitative comparison of the effect of the latent peceptual loss. Models trained on ImageNet-1k at 256 resolution with (bottom) and without (top) our perceptual loss. Without the perceptual loss, the model frequently fails to generate coherent structures, using the perceptual loss, the model generates more plausible objects with sharper details. The models are finteuned for 100k iterations from a checkpoint that was trained for 200k iterations. The samples are generated *without classifier-free guidance or EMA*, using 50 DDIM steps.

Samples of ImageNet-1k models.[h] In Figure 15 we show samples of models trained with or without LPL on ImageNet-1k at 512 resolution. At higher resolutions, we also observe that the model trained with LPL generates images that are sharper and present more fine-grained details compared to the baseline.

Samples on T2I models.[h] We provide additional qualitative comparisons regarding our LPL loss. Figure 17 showcases results on a model trained on CC12M at 512 resolution, Figure 16 showcases results on a model trained on S320M at 256 resolution.

Figure 15 Influence of finetuning a class-conditional model of ImageNet-1k at 512 resolution using our perceptual loss. Our perceptual loss (bottom row) leads to more realistic textures and more detailed images.

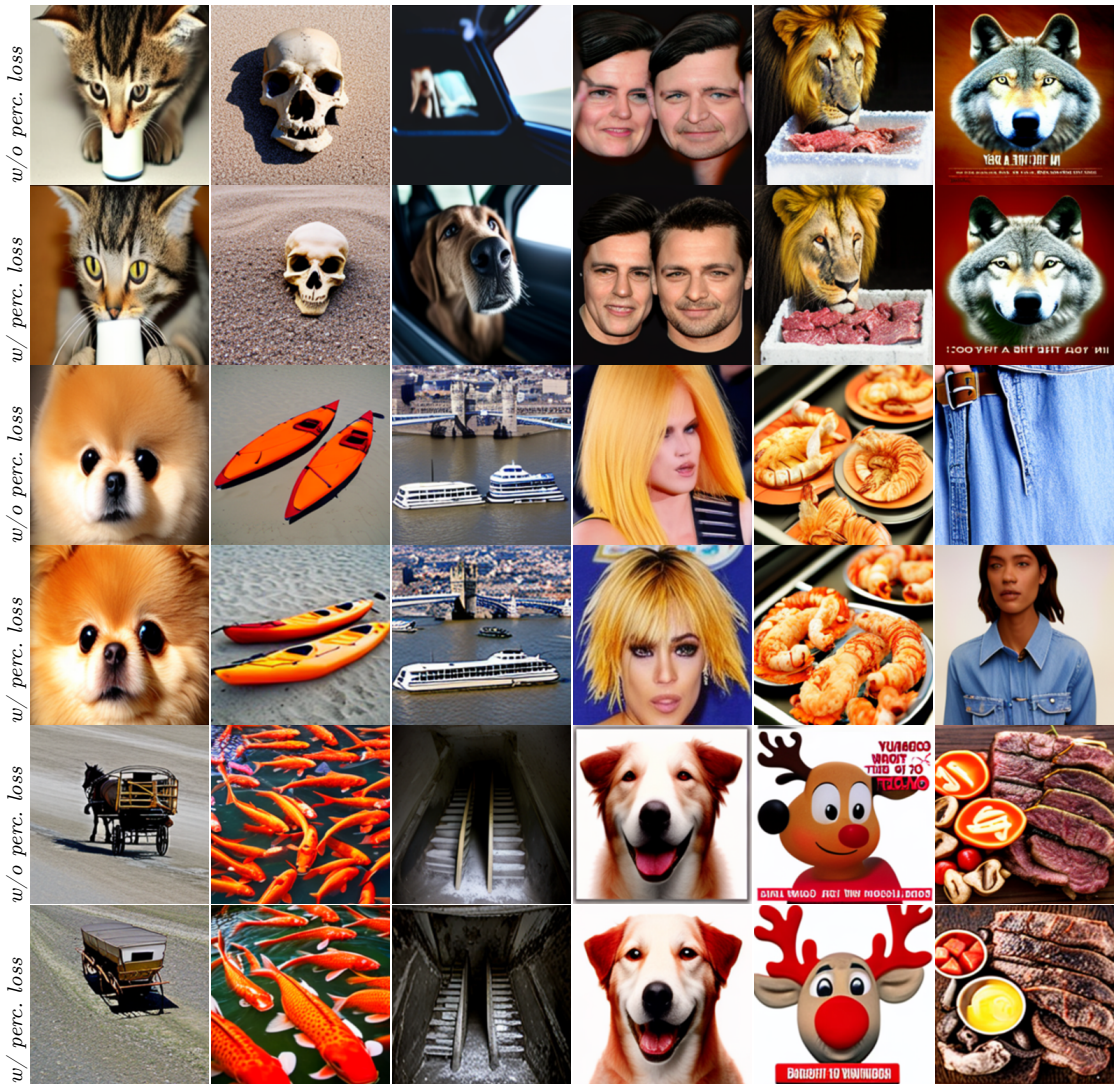
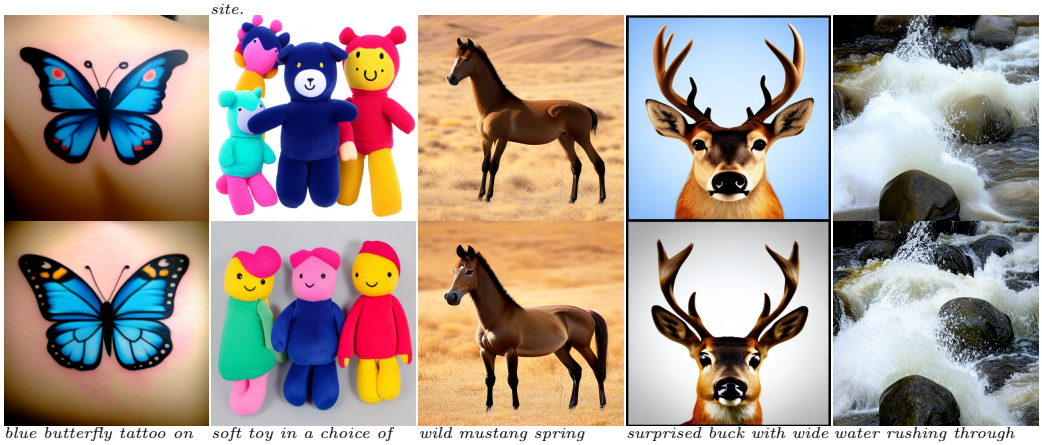


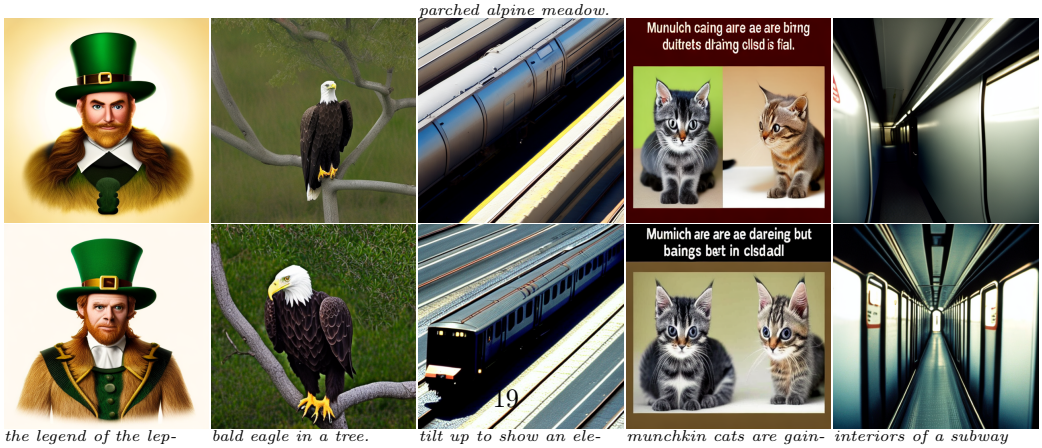
Figure 16 Qualitative comparison. of samples from models trained with and without our LPL on S320M at 256 resolution.

this makes me miss my short hair. | *person weathered , cracked purple leather chairs sitting outside of a building.* | *human skull on the sand.* | *gingerbread little men on the beach.* | *my dog , person making friends.*

a bee gathering nectar from a wild yellow flower. | *two - headed statue in an ancient city of unesco world heritage site.* | *the - cat breeds in photographs.* | *a beautiful shot of the flowers.* | *photo of rescue dog , posted on the page on facebook.*



blue butterfly tattoo on back of the shoulder. | *soft toy in a choice of colours.* | *wild mustang spring foal with its mare in a parched alpine meadow.* | *surprised buck with wide eyes.* | *water rushing through rocks in a river.*



the legend of the leprechaun. | *bald eagle in a tree.* | *tilt up to show an elevated train riding down the track.* | *munchkin cats are gaining in popularity , but is breeding these cats cruel?* | *interiors of a subway train.*