Scale Space Diffusion

Supplementary Material

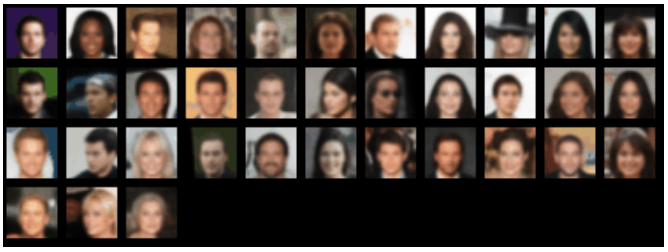


Figure 7. Animation of the predicted clean image $x_{0,\theta}^{r(t-1)}$ over the generation process for gradual downsizing degradation operator in SSD framework. (Best viewed in Adobe Reader).

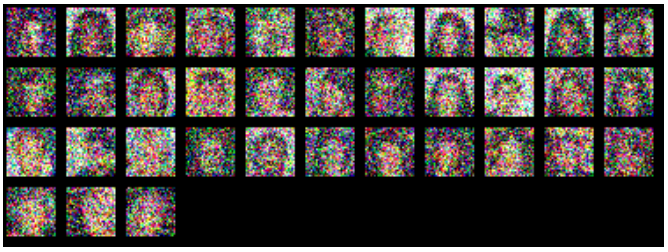


Figure 8. Animation of the noisy intermediate state x_t over the generation process for the gradual downsizing degradation operator in SSD framework. (Best viewed in Adobe Reader).

Contents

6. Clarifications

We add some clarifications for our main paper here.

1. All variable names used in Algo. 1, 2, and 3 have their usual meanings. For example, `a_t`, `a_t_minus1` denote the signal coefficients a_t, a_{t-1} respectively, while `sigma_t`, `sigma_tminus1` denote the noise schedule σ_t, σ_{t-1} respectively. `size_out` denotes the (height, width) of the output of `M` operator.

7. Future Works

The focus of this work has been to analyze the connection between diffusion models and scale space theory, while proposing to merge them using Scale Space Diffusion with Flexi-UNet. We do not use any advanced techniques to tune our framework or architectures for the most optimal performance. Instead, we use the standard hyperparameters from the base codebase to keep the choices simple and the number of experiments under check given the expense of each training. The use of advanced techniques is out of scope for this work given the conference length manuscript.

However, there are multiple future exploration directions which have high potential for improvement in performance. For example, adapting newer diffusion samplers instead of using DDPM-style samplers can improve both performance and inference speeds. Similarly, progressive curriculum learning for different layers or resolutions, as done by works with multi-resolution trainings [13, 22], should also yield improvement in training optimization.

Why not use a Transformer-based architecture? The two most popularly used architectures in diffusion are – convolutional UNet [34] based ADM [11], and vision transformer (ViT) based DiT [32]. Another popular architecture is U-ViT [4], that combines the skip connections from UNet with a ViT architecture. One thing to note is that, DiT was designed for latent spaces and hence did not take into consideration the blowing up of the quadratic complexity of the attention mechanism when applied in the pixel space [6]. U-ViT acknowledges this issue, and explicitly works in a latent space for higher resolutions. Newer works like HDiT [7] try to mitigate this issue using neighborhood attention instead of global attention is all layers. But such non-trivial design decisions in the architecture can develop into confounding factors. Since our goal is to understand how scale-spaces can be integrated into diffusion models, for simplicity we stick to the standard ADM base architecture, a widely used pixel and latent diffusion architecture [33]. Nonetheless, for future work, a similar integration of scale-space theory should also be explored with transformer based architectures.

Hyperparameter	CelebA-64	CelebA-128	CelebA-256	ImageNet-64
Noise Schedule	Linear	Linear	Linear	Linear
Denoising Steps	1000	1000	1000	1000
Optimizer	AdamW	AdamW	AdamW	AdamW
Batch Size	128	128	64	128
Learning Rate	0.0001	0.0001	0.00005	0.0001
Number of Iterations	1 million	300k	300k	1 million

Table 5. Hyperparameters for all datasets.

Implementation Choice	DDPM-ϵ	DDPM-x_0	Blurring Diffusion	SSD
Reverse Process Variance	fixed-large	fixed-large	fixed-small	fixed-small
Loss	L_{simple}	L_{simple} + Min-SNR-5	L_{simple}	L_{simple} + Min-SNR-5

Table 6. Additional implementation details.

Table 7. Inference time. By default we use DDPM sampling, but we also show $25^{\dagger}$ steps DDIM [36] speeds.

Method	#Steps	Speedup (Inference Time)	FID
DDPM-x_0	1000	1.00×	2.98
	250	4.18×	14.00
	$25^{\dagger}$	38.87×	4.70
DDPM-ϵ	1000	1.05×	2.22
	250	4.18×	11.02
	$25^{\dagger}$	38.06×	3.76
SSD(Flexi-UNet, 2L)	1000	1.18×	2.14
	250	4.80×	2.87
SSD(Flexi-UNet, 4L)	1000	1.58×	4.28
	250	5.91×	4.90

Table 8. Inference time (in secs) per gen at 64 res (1000 steps, bs=1, 1 A4000): Lanczos sampling vs. torch.randn call.

Method	SSD (2L)	SSD (4L)
w/ Lanczos	15.38	13.43
w/o Lanczos	15.35	13.40

8. Additional Material

8.1. Hyperparameters

The set of hyperparameters that we use for each dataset is summarized in Table 5. We also note additional experimental details in Table 6.

8.2. Parts of our Approach

Our approach consists of two parts. The first part is the Scale Space Diffusion mathematical formulation and the second part is the Flexi-UNet architecture. In the main paper, we have presented the combination of both parts as our complete approach. But here we also want to show that each part is effective on its own. So, in Section 8.2.1, we explore whether the mathematics behind SSD can be applied without a modified architecture, while in Section 8.2.2, we check if Flexi-UNet can be used without our mathematical framework, summarized in Table 9.

8.2.1. Validity of SSD

One way to verify whether SSD framework works without using a modified architecture is to assume that the actual states of the diffusion model are at a certain resolution, but when passing through the model, we resize them to the model input size. Similarly, the model outputs are resized to the required output resolution before applying losses. We test this with CelebA-32 dataset just to check the correctness of SSD. For this, we use a DDPM reimplementation (not ours) optimized for resolution 32 images [40] , since ADM's codebase does not support that resolution, and a smaller resolution is faster to verify on. We train these models for 300 epochs and use 5 steps of resolutions (2, 4, 8, 16, 32). We note their FIDs in Table 10.

Table 9. Parts of our approach, and validity of each part.

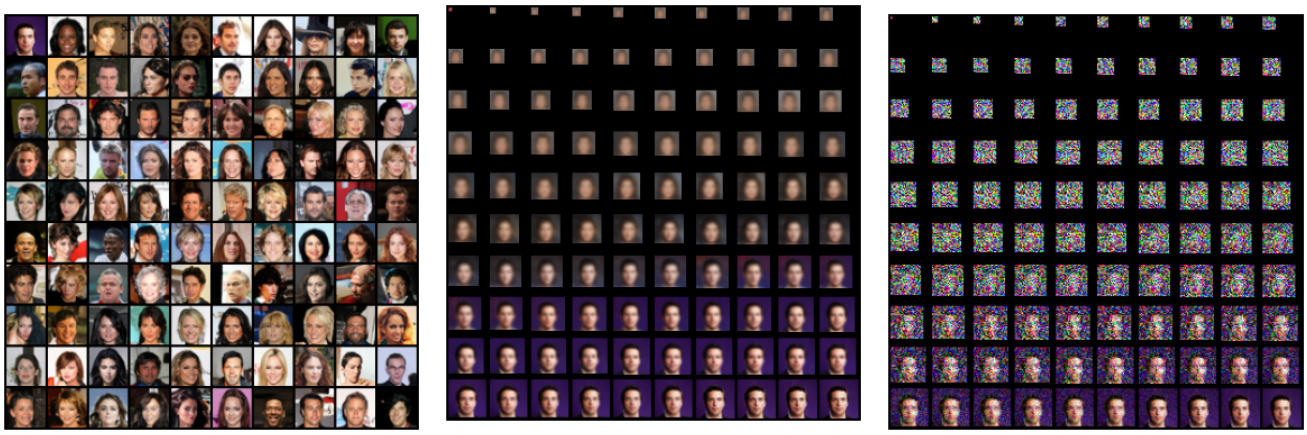
	SSD	Flexi-UNet
Section 8.2.1	✓	✗
Section 8.2.2	✗	✓
Main paper	✓	✓

Table 10. Results of only SSD (w/o Flexi-UNet) on CelebA-32. (Here we resize the inputs to the model input resolution.)

Method	FID
DDPM-ϵ	2.85
SSD (w/o Flexi-UNet, 5L)	5.55
SSD (w/o Flexi-UNet, gradual downsizing)	4.10

Alternative Degradation. All the degradations used in this work till now have been $2\times$ downsampling. However, given the general nature of the theory, it is not limited to just this choice. Here we test using a gradual downsizing instead of $2\times$ downsizing steps. In this degradation, whenever the resolution changes, it does so by only 1 pixel at a time. We try going from $2 \rightarrow 32$. We report its FID in Table 10. We show some static visual results in Fig. 9. We show some interesting animated visualizations (view in Adobe Reader) in Fig. 7, and Fig. 8.

Effect of Isotropic Approximation. Another thing we wanted to test was whether we could approximate the non-isotropic Gaussian noise sampling (Algo. 3) with isotropic Gaussian noise. For testing purposes, during the generation procedure (of the gradual downsizing degradation case), in the resolution changing steps, we first use Algo. 3 to sample non-isotropic noise, and then find the mean and variance over the height and width dimensions of this noise tensor. Instead of using the sampled non-isotropic noise for the stochasticity in Eq. 6, we instead use an isotropic noise sampled using `torch.randn()` with the calculated mean and variance. As seen in Fig. 10, this leads to the colors becoming flat and saturated, despite having facial structures. This shows that the assumption of isotropic covariance for the reverse process may not actually be valid, as assumed in [1]. And we need to sample from non-isotropic Gaussians depending upon the linear operator.



(a) Generated Samples (b) Predicted clean images $x_{0,\theta}^{r(t-1)}$ (c) Noisy states x_t

Figure 9. Visual results of SSD with gradual downsizing degradation (1 pixel downsizing instead of 2× downsizing)

Figure 10. Effect of using isotropic noise instead of non-isotropic noise in the reverse diffusion process of SSD.

8.2.2. Effectiveness of Flexi-UNet

In this section, we demonstrate that Flexi-UNet can naturally accommodate different formulations of the diffusion process to support multi-resolution inputs and outputs. To do so, we build upon previous works that introduce corrective noise when an upsampling operation is performed in the diffusion process [6, 20, 21]. We implement these ideas within Flexi-UNet to both validate the flexibility of our architecture and quantify the computational benefits obtained from operating across resolutions. A key challenge addressed in these works is the distribution mismatch that arises when a noisy latent is upsampled. Prior works [20] show that applying a 2× nearest-neighbor upsampling step produces a block-structured covariance that deviates from the forward diffusion trajectory. Their solution injects structured noise and identifies an adjusted timestep that realigns the upsampled latent with the original process. While this motivates our analysis, our setting is different from this in two ways: a) we operate entirely in pixel-space rather than latent space, and b) we consider multiple (more than one) upsampling stages throughout the denoising process. With these conditions in mind, our setup is as follows:

Let x_t^r be a valid DDPM forward state at a timestep t for resolution r:

$$\begin{aligned} x_t^r &\sim \mathcal{N}\big(\sqrt{\bar{\alpha}_t}\, x_0^r,\ (1-\bar{\alpha}_t)I\big)\,, \\ \text{Let } x_t^R &= \text{Upsample}(x_t^r), \\ x_t^R &\sim \mathcal{N}\big(\sqrt{\bar{\alpha}_t}\, U x_0^r,\ (1-\bar{\alpha}_t)\, UU^\top\big) \\ &\quad \text{where } U \text{ is the Upsampling matrix.} \end{aligned}$$

Let x_s^R be a valid DDPM forward state at some other timestep s for resolution R:

$$x_s^R \sim \mathcal{N}\big(\sqrt{\bar{\alpha}_s}\, x_0^R,\ (1-\bar{\alpha}_s)\, I\big)\,,$$

The upsampled state x_t^R has covariance proportional to $UU^\top$, which differs from the isotropic Gaussian noise assumed by the DDPM forward process at resolution R. To correct this mismatch, we add corrective noise and roll back to a previous timestep. Let the corrected sample be

$$\tilde{x}_t^R = a\, x_t^R + b\, z, \qquad z \sim \mathcal{N}\big(0,\ I - cUU^\top\big)\,.$$

Then the distribution of $\tilde{x}_t^R$ is

$$\tilde{x}_t^R \sim \mathcal{N}\big(a\sqrt{\bar{\alpha}_t}\, U x_0^r,\ a^2(1-\bar{\alpha}_t)\, UU^\top + b^2\left(I - cUU^\top\right)\big)\,.$$

We make an approximation to match the mean and covariance of $\tilde{x}_t^R$ to x_s^R

$$\begin{aligned} a^2\bar{\alpha}_t &= \bar{\alpha}_s \\ b^2 &= 1-\bar{\alpha}_s \\ a^2(1-\bar{\alpha}_t) &= b^2 c \end{aligned} \tag{8}$$

Table 11. Results of Flexi-UNet (w/o SSD) on CelebA-64. Computed at 500k iterations. Inference time is computed as the average time (in minutes) to generate a batch of samples (256 samples).

Method	FID	Inference Time
Flexi-UNet (w/o SSD, Equal, 2L)	2.44	15.52
Flexi-UNet (w/o SSD, Equal, 4L)	5.79	13.32
Flexi-UNet (w/ SSD, ConvexDecay0.5, 2L)	2.26	14.98
Flexi-UNet (w/ SSD, ConvexDecay0.5, 4L)	4.87	11.20

Solving the three equations mentioned in Equation 8, gives us

$$c = \frac{a^2(1-\bar{\alpha}_t)}{b^2} = \frac{\bar{\alpha}_s(1-\bar{\alpha}_t)}{\bar{\alpha}_t(1-\bar{\alpha}_s)} \tag{9}$$

We first obtain the value of $\bar{\alpha}_s$ that satisfies Equation 9 for a given choice of c, and then obtain the corresponding timestep s. We sweep through values of c in range $0 \leq c \leq 0.25$ (as mentioned in [20]) to produce different values of s. We compute all such candidate values of s and pick the best s empirically. For each value of c, we generate the corrected samples $\tilde{x}_t^R$ and the corresponding DDPM forward samples x_s^R using 2048 training images. We then compute the Jensen–Shannon divergence between these distributions to obtain the final backtracking index s as the one that produces the minimum JS divergence.

This experiment serves as our validation of our proposed method Flexi-UNet. During training, we follow a specific resolution schedule, so that for each timestep t, the model receives a state $x_t^{r(t)}$. To support distribution correction, we additionally include timestep s, $\tilde{x}_t^R$ to the training samples. During inference, the denoising process follows the standard reverse diffusion trajectory, with the following change: whenever the process reaches a timestep that has an upsampling step, the model rolls back to a slightly earlier timestep and continues denoising from that point at the higher resolution. This experiment illustrates the computational advantages of operating at multiple resolutions, using an architecture like Flexi-UNet, as a lot of the early denoising occurs at lower spatial resolutions. However, this setup requires rollback around each upsampling point, creating overlapping steps in the reverse process. While this model provides computational savings, there is an additional overhead of denoising for additional timesteps.

In Table 11, we show the FID values obtained for this experiment after training the model for 500k iterations. We compare the performance of Flexi-UNet trained with SSD to Flexi-UNet trained without SSD. We observe that Flexi-UNet with SSD has better FID values, while also being faster at inference.

8.3. Resolution Schedules

Here, we will define the functions that we used for the resolution schedules. We define what the resolution of the image should be given the diffusion timestep t, using a function $r(t)$. As shown in Fig. 5, we use a discrete version of the resolution schedule, but it is based on a continuous function. Suppose for the discrete version we use a list of resolutions $[r_{\text{min}}, 2r_{\text{min}}, \ldots, 2^{n-2}r_{\text{min}}, 2^{n-1}r_{\text{min}}]$ where r_{min} is the smallest resolution and n is the number of resolutions. For the continuous version, let's first define normalized time $\tau = t/(T-1)$, where T denotes the number of diffusion states. Then the normalized time to resolution schedule is defined as:

$$r_{\text{cont}}(\tau) = r_{\text{min}} \cdot 2^{(n-1)f(\tau)}$$

where $f(\tau)$ is the exponential schedule function that works as the multiplier to the exponent of 2. For example, when $f(\tau) = 0$, then $r_{\text{cont}}(\tau) = r_{\text{min}}$, while when $f(\tau) = 1$, then $r_{\text{cont}}(\tau) = r_{\text{max}} = 2^{n-1}r_{\text{min}}$.

For the discrete version, we want to similarly sample from $R = [r_{\text{min}}, 2r_{\text{min}}, \ldots, 2^{n-2}r_{\text{min}}, 2^{n-1}r_{\text{min}} = r_{\text{max}}]$, using the same schedule but over these discrete values. So, here we instead index the schedule function $i(\tau)$ that gives the index to select from R given τ.

$$r(\tau) = R[i(\tau)]$$

Similar to f, when $i(\tau) = 0$, we have $r(\tau) = r_{\text{min}}$, and when $i(\tau) = 1$, $r(\tau) = r_{\text{max}}$. Now we can introduce our schedules.

8.3.1. Equal

This is the easiest linear schedule.

- Continuous: $f(\tau) = 1 - \tau$
- Discrete: $i(\tau) = n - 1 - \lfloor n\tau \rfloor$

8.3.2. ConvexDecay_γ

With a $\gamma > 0$ parameter, this function can simulate a convex or concave function depending on this parameter.

- Continuous: $f(\tau) = 1 - (1-\tau)^\gamma$
- Discrete: $i(\tau) = n - 1 - \lfloor nf(\tau) \rfloor$

For $\gamma > 1$, it shows slow decay first, then faster, while for $\gamma < 1$, fast decay first, then slower.

8.3.3. TanhLikeDecay_γ

Here we wanted a function that looks like $\tanh(\cdot)$ function, which is steep at the highest and the lowest timesteps but is flat in the middle. This essentially spends more time in the middle resolutions. We approximate this using a polynomial.

First, we define a polynomial over a variable $u \in [-0.5, 0.5]$ as follows:

$$x(u, \gamma) = \text{sign}(u)|u|^\gamma + 0.5$$

$$p(x) = -2x^3 + 3x^2 - 0.5$$

The polynomial $p(x)$ is monotonically increasing in the range of $[-0.5, 0.5]$ for $x \in [0, 1]$, while $x(u)$ is a function that looks like the $\tanh(\cdot)$ function but is centered around 0.5. Essentially, $p(x(u))$ looks like the tanh shape and is

centered around the origin, but has varying range dependent on γ. We want this function to be equal to 1 at $u = 0.5$ and -1 at $u = -0.5$. To achieve that, we normalize this function:

$$\hat{p}(u, \gamma) = \frac{p(x(u, \gamma))}{p(x(0.5, \gamma))}$$

Finally, to shift this function from $[-0.5, 0.5] \to [-1, 1]$ to $[0, 1] \to [0, 1]$, we apply the following transformation:

$$\text{tanh\_like}(u, \gamma) = 0.5 \cdot \hat{p}(x(u - 0.5, \gamma)) + 0.5$$

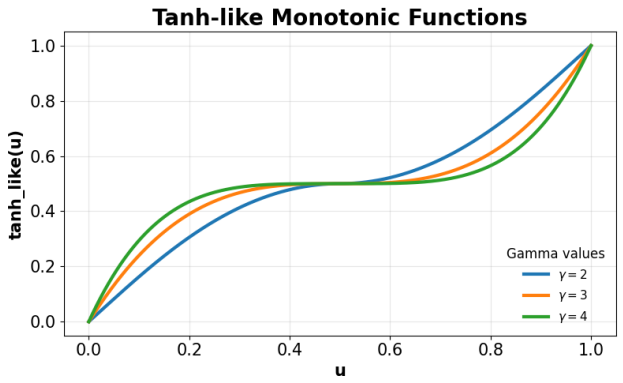


Figure 11. Visualization of tanh_like$(\cdot)$ for different γ's.

Now, based on this definition, we can define the schedule.

- Continuous: $f(\tau) = 1 - \text{tanh\_like}(1 - \tau, \gamma)$
- Discrete: $i(\tau) = \lfloor n f(\tau) \rfloor$

8.3.4. SigmoidLikeDecay_γ

Here we want a simoid-like curve, *i.e.*, steep in the middle while flatter at the beginning and the end. We can such a curve by inverting $\hat{p}(\cdot)$. Following similar stretching and normalization, we can define another function that goes from $[-0.5, 0.5] \to [-1, 1]$ as:

$$\hat{h}(u) = \frac{(0.5 \cdot \hat{p})^{-1}(u, \gamma)}{(0.5 \cdot \hat{p})^{-1}(-0.5, \gamma)}$$

. Using the same shifting to transform $[-0.5, 0.5] \to [-1, 1]$ to $[0, 1] \to [0, 1]$, we have:

$$\text{sigmoid\_like}(u, \gamma) = 0.5 \cdot \hat{h}(x(u - 0.5, \gamma)) + 0.5$$

Finally, we define the schedule.

- Continuous: $f(\tau) = 1 - \text{sigmoid\_like}(1 - \tau, \gamma)$
- Discrete: $i(\tau) = \lfloor n f(\tau) \rfloor$

8.4. More Comparisons

Upsampling Diffusion Probabilistic Models (UDPM) [1]. The mathematical formulation and implementation of SSD can be viewed as a generalization of UDPM. However, UDPM should be considered as a GAN instead of diffusion, as their performance degrades without perceptual and adversarial losses (Table 14) and generations are washed out (Fig. 12, right). Nonetheless, even without extra losses, SSD outperforms UDPM in FID and training time. Furthermore, UDPM has not been tested at resolutions higher than 64.

Table 12. FID comparison of SSD and super-resolved DDPM (64 res.) using an OpenImages pretrained 4× LDM.

Method	FID
SSD (3L, res. 256)	7.79
low-res diffusion (res. 64) + super-res (4×)	7.91

Table 13. Inference time per batch: SSD and LDMs (1000 steps, bs=32, A4000).

Method	Inference time (secs)
SSD (6L, res. 256)	495
LDM (res. 256)	515

Table 14. Comparison with UDPM at 64 resolution: FID, training (1 H100), and inference speed (1 A4000, bs=256)

Method	Inference steps	Inference Time / batch (in secs)	Training Time (250K iters) (in hours)	FID (250K iters)
DDPM-ϵ	1000	1018.07	17.575	2.36
SSD (2L)	1000	898.71	15.658	2.68
SSD (4L)	1000	672.09	13.095	4.1
UDPM	3	1.88	30.58	7.51
UDMP (w/o Adv. & Perceptual loss)	3	1.84	31.63	98.61

Figure 12. UDPM generations w/ (left) and w/o (right) adversarial and perceptual losses.

Latent Diffusion Models (LDM) [33]. LDMs operate in latent space with different architectures and rely on a compute-intensive pipeline, including two-stage VAE training on large-scale datasets such as OpenImages, making fair comparison difficult. Nonetheless, Table 13 shows SSD (6L) is faster than LDM. Moreover, SSD can be applied in latent space as a multi-resolution interpolation degradation, enabling more efficient Scale Space LDMs.

Low-res diffusion + super-res. Another baseline could be using a low-resolution generation and applying a super-resolution model over it. Table 12 shows that even with a pretrained LDM super-res model trained for 3× more iterations, and on a large dataset, SSD has better performance. Adding multiple stages normally leads to distribution shifts as well as more inference steps coming from different stages.

PixelFlow [6], DFM [14]. These are flow-based DiT models with differential equation solver-based sampling, and hence, are hard to compare fairly against. Nonetheless, in a fair setting in section 8.2.2, we recreate a multi-res pixel diffusion similar to PixelFlow, and show that using Flexi-UNet formulation outperforms it (Table 11).

8.5. Quantitative Results

Number of Inference Steps. In Table 7, we compare inference speed across different samplers and denoising steps. We report DDPM sampling with the default 1000 steps, a reduced 250 step process, and DDIM with 25 steps. For our

method, we report results with 1000 and 250 DDPM steps, since SSD is formulated in the DDPM setting. We observe that reducing the number of diffusion steps leads to a much larger performance degradation for DDPM-ϵ and DDPM-x_0 than for our approach. This aligns with prior observations that DDPM-ϵ models trained with the L_{simple} loss (with fixed sigmas) deteriorate substantially when the number of sampling steps is reduced [30], which is reflected in our results as well.

We note that SSD degrades far less when reducing the sampling steps to 250, while also providing substantial inference speedups. However, to ensure a fair comparison against baselines, we report all final quantitative results in the paper using the standard 1000-step setting. The speedup column in Table 7 reports the speedup obtained in generating a batch of 256 samples relative to the time taken by DDPM-x_0.

Lanczos sampling overhead. Table 8 shows that the overhead of using Lanczos instead of torch.randn call is negligible, since it is applied only in the resolution-changing steps ($1\times$ in SSD (2L), $2\times$ in SSD (3L)). Refer to Table 7 for comparison against DDPM.

8.6. Qualitative Results

We show qualitative results of SSD on every setting that we have trained and noted in Tables 2, and 3. For every setting, we show the progression of noisy states x_t and predicted clean images $x_{0,\theta}^{r(t-1)}$ during generation, and a grid of generated images. The results start on the next page.

Figure 13. Progression of noisy states x_t during generation using SSD (Flexi-UNet, 3L) on CelebA-256.

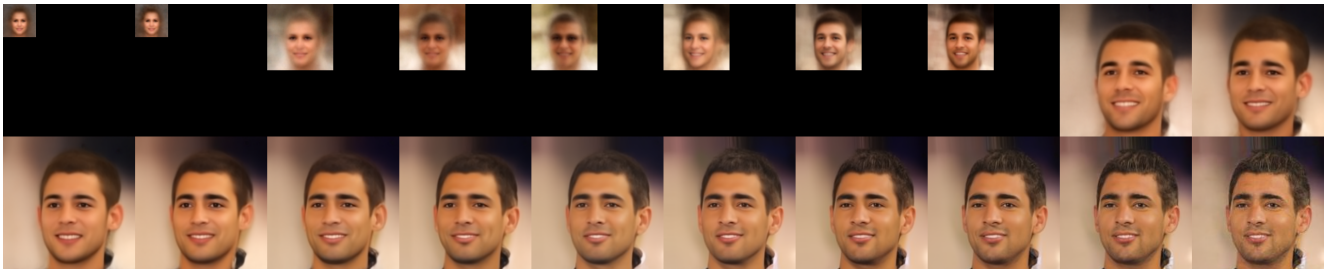


Figure 14. Progression of predicted clean images $x_{0,\theta}^{r(t-1)}$ during generation using SSD (Flexi-UNet, 3L) on CelebA-256.

Figure 15. Generated Samples using SSD (Flexi-UNet, 3L) on CelebA-256.

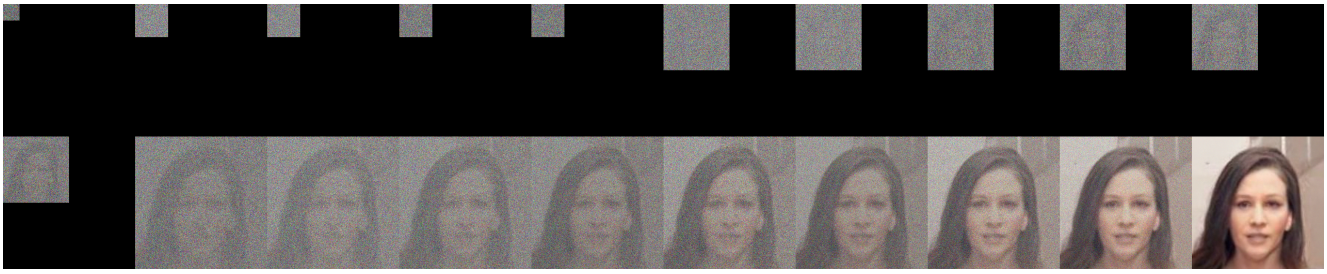


Figure 16. Progression of noisy states x_t during generation using SSD (Flexi-UNet, 4L) on CelebA-256.

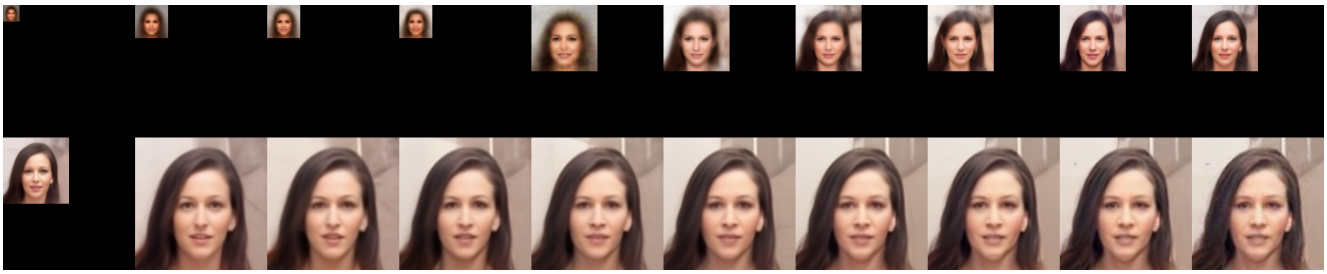


Figure 17. Progression of predicted clean images $x_{0,\theta}^{r(t-1)}$ during generation using SSD (Flexi-UNet, 4L) on CelebA-256.

Figure 18. Generated Samples using SSD (Flexi-UNet, 4L) on CelebA-256.

Figure 19. Progression of noisy states x_t during generation using SSD (Flexi-UNet, 6L) on CelebA-256.

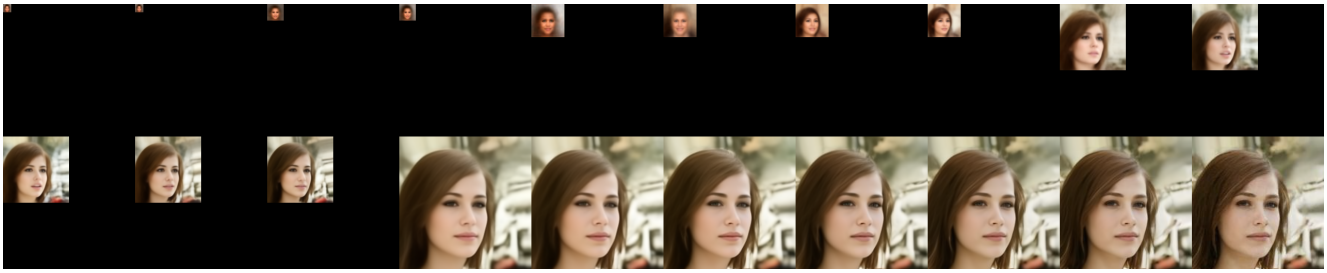


Figure 20. Progression of predicted clean images $x_{0,\theta}^{r(t-1)}$ during generation using SSD (Flexi-UNet, 6L) on CelebA-256.

Figure 21. Generated Samples using SSD (Flexi-UNet, 6L) on CelebA-256.

Figure 22. Progression of noisy states x_t during generation using SSD (Flexi-UNet, 3L) on CelebA-128.

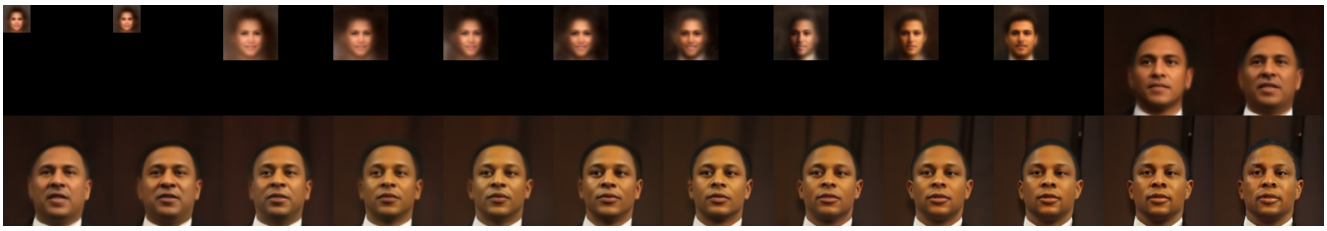


Figure 23. Progression of predicted clean images $x_{0,\theta}^{r(t-1)}$ during generation using SSD (Flexi-UNet, 3L) on CelebA-128.

Figure 24. Generated Samples using SSD (Flexi-UNet, 3L) on CelebA-128.

Figure 25. Progression of noisy states x_t during generation using SSD (Flexi-UNet, 5L) on CelebA-128.

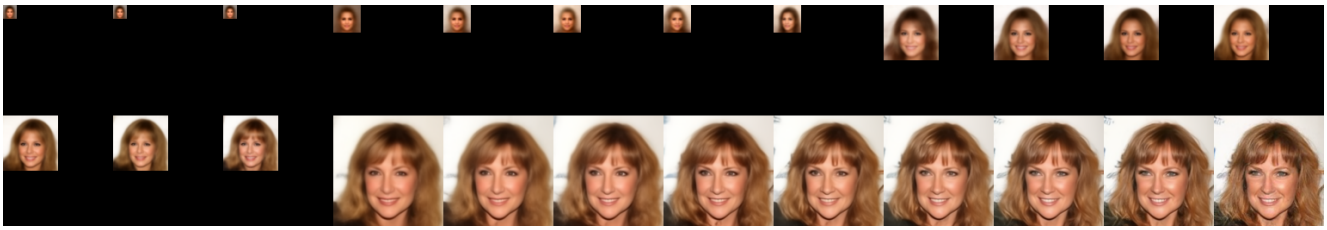


Figure 26. Progression of predicted clean images $x_{0,\theta}^{r(t-1)}$ during generation using SSD (Flexi-UNet, 5L) on CelebA-128.

Figure 27. Generated Samples using SSD (Flexi-UNet, 5L) on CelebA-128.

Figure 28. Progression of noisy states x_t during generation using SSD (Flexi-UNet, 2L) on CelebA-64.

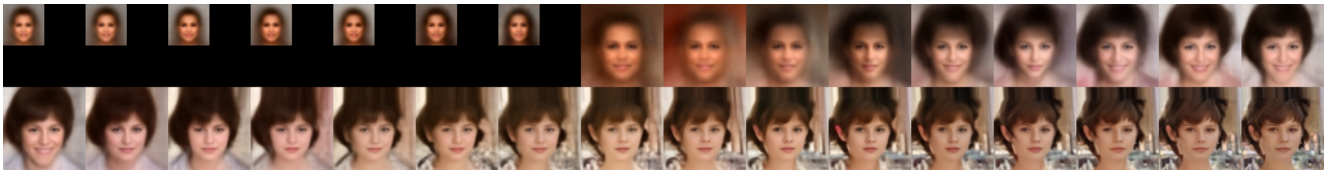


Figure 29. Progression of predicted clean images $x_{0,\theta}^{r(t-1)}$ during generation using SSD (Flexi-UNet, 2L) on CelebA-64.

Figure 30. Generated Samples using SSD (Flexi-UNet, 2L) on CelebA-64.

Figure 31. Progression of noisy states x_t during generation using SSD (Flexi-UNet, 3L) on CelebA-64.

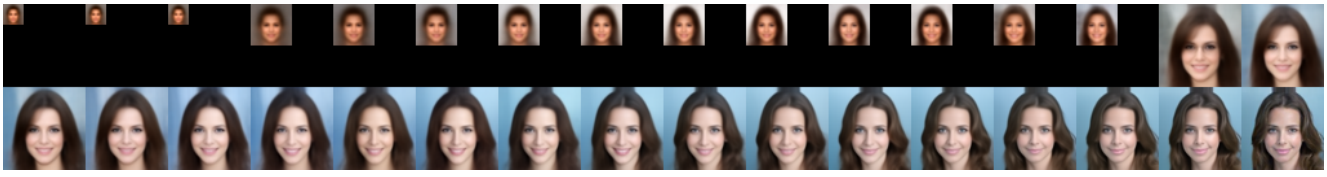


Figure 32. Progression of predicted clean images $x_{0,\theta}^{r(t-1)}$ during generation using SSD (Flexi-UNet, 3L) on CelebA-64.

Figure 33. Generated Samples using SSD (Flexi-UNet, 3L) on CelebA-64.

Figure 34. Progression of noisy states x_t during generation using SSD (Flexi-UNet, 4L) on CelebA-64.

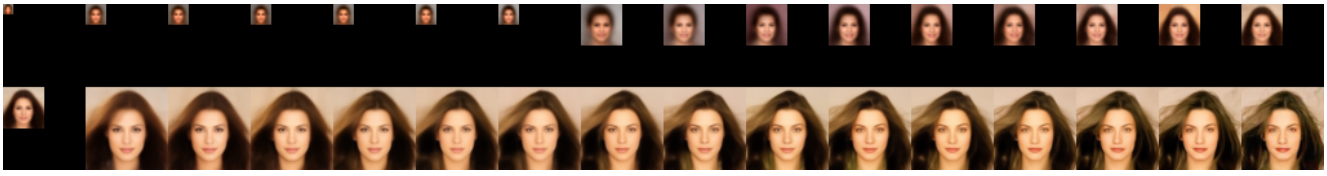


Figure 35. Progression of predicted clean images $x_{0,\theta}^{r(t-1)}$ during generation using SSD (Flexi-UNet, 4L) on CelebA-64.

Figure 36. Generated Samples using SSD (Flexi-UNet, 4L) on CelebA-64.

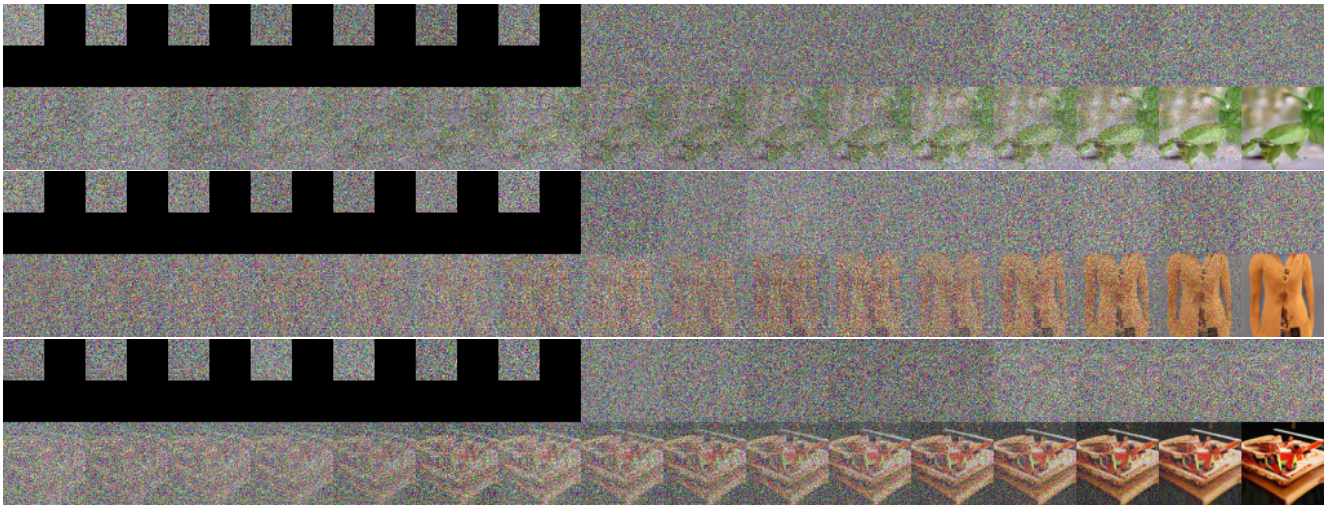


Figure 37. Progression of noisy states x_t during generation using SSD (Flexi-UNet, 2L) on ImageNet-64. Here we show the progression of 3 samples; each pair of rows corresponds to a single sample.

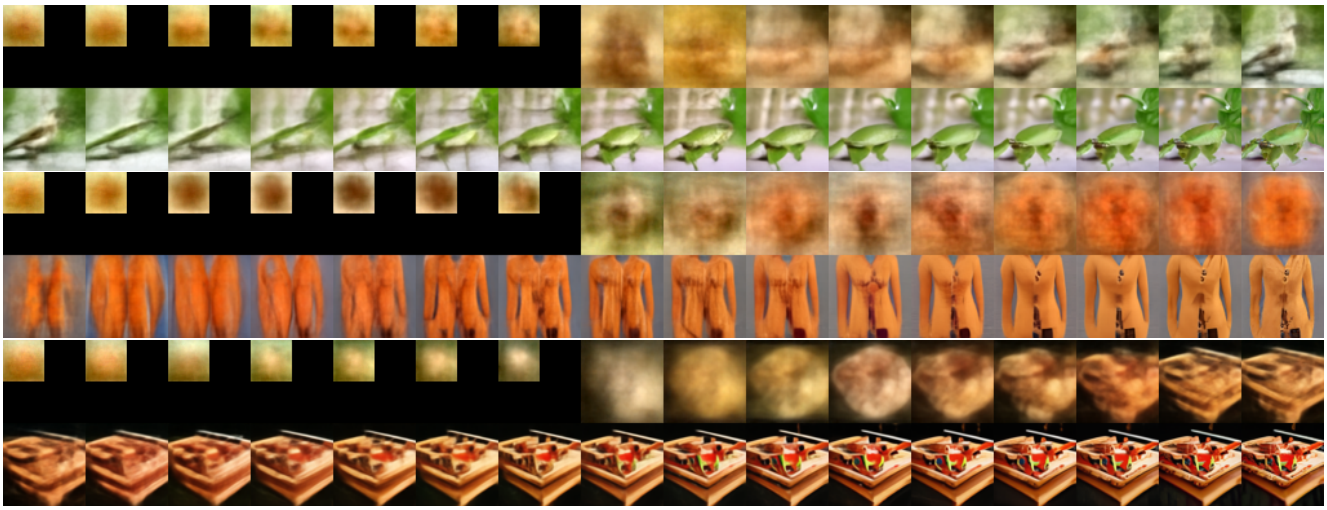


Figure 38. Progression of predicted clean images $x_{0,\theta}^{r(t-1)}$ during generation using SSD (Flexi-UNet, 2L) on ImageNet-64. Here we show the progression of 3 samples; each pair of rows corresponds to a single sample.

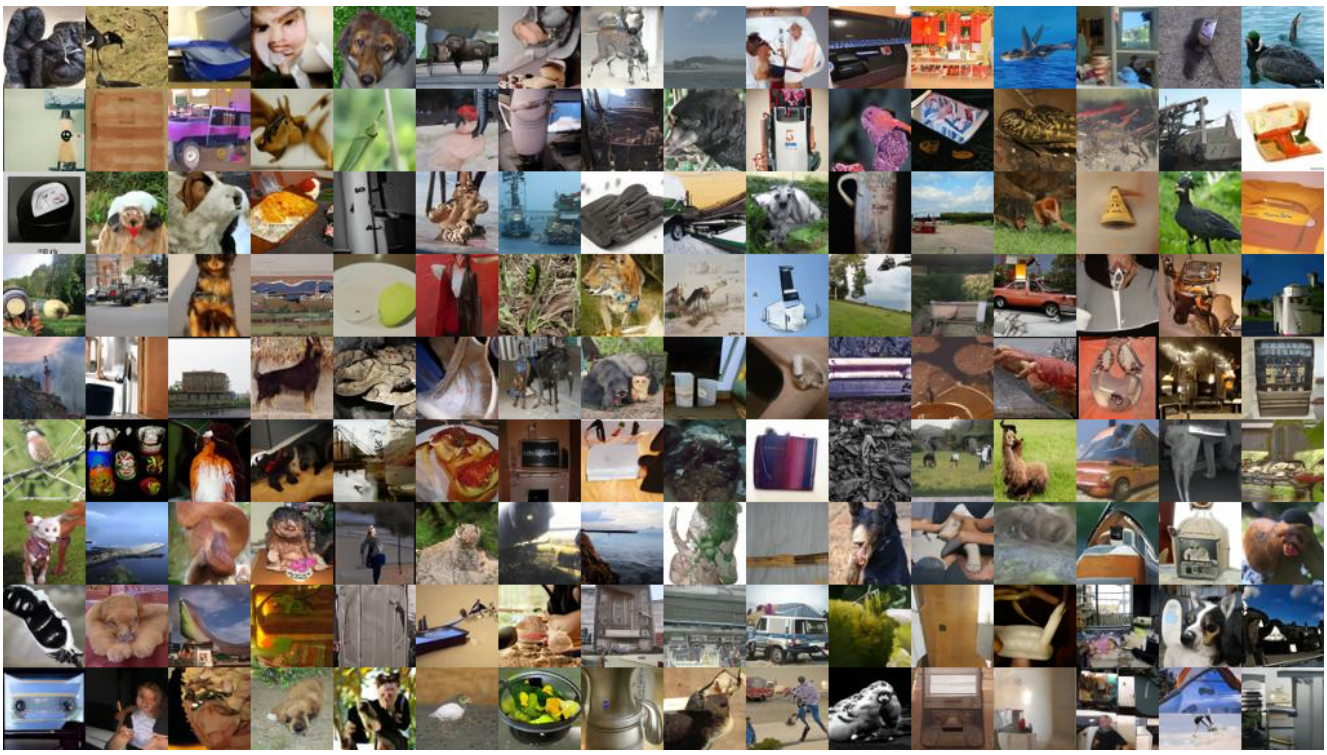


Figure 39. Generated Samples using SSD (Flexi-UNet, 2L) on ImageNet-64.

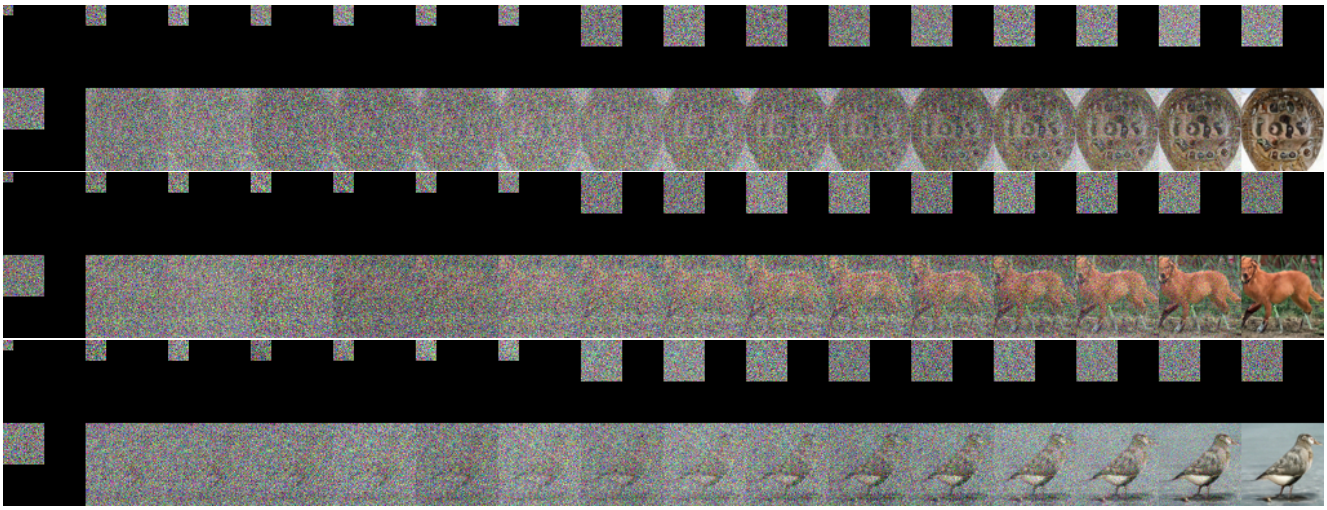


Figure 40. Progression of noisy states x_t during generation using SSD (Flexi-UNet, 4L) on ImageNet-64. Here we show the progression of 3 samples; each pair of rows corresponds to a single sample.

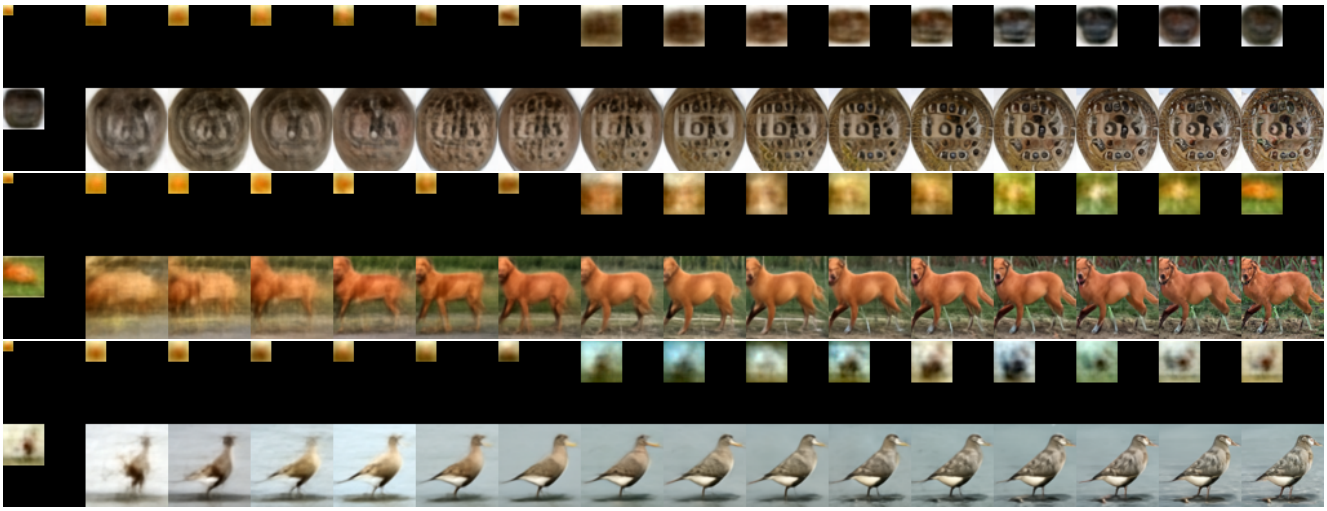


Figure 41. Progression of predicted clean images $x_{0,\theta}^{r(t-1)}$ during generation using SSD (Flexi-UNet, 4L) on ImageNet-64. Here we show the progression of 3 samples; each pair of rows corresponds to a single sample.

Figure 42. Generated Samples using SSD (Flexi-UNet, 4L) on ImageNet-64.

9. Mathematical Derivations

In this section, we provide derivations for various mathematical results provided in the main paper.

9.1. Forward Transition

Theorem 1 (Forward Transition)**.** *Let a generalized linear diffusion process be defined by*

$$x_t = M_t x_{t-1} + \eta_t, \qquad \eta_t \sim \mathcal{N}(0, \Sigma_{t|t-1}), \tag{10}$$

and suppose the marginal distribution satisfies

$$q(x_t \mid x_0) = \mathcal{N}(\mu_t, \Sigma_t). \tag{11}$$

Then the transition mean and covariance are given by

$$\mu_t = M_{1:t} x_0 \tag{12}$$

$$\Sigma_{t|t-1} = \Sigma_t - M_t \Sigma_{t-1} M_t^T. \tag{13}$$

Proof. The mean part is true by design of the cumulative linear operator. Here we derive $\Sigma_{t|t-1}$.

$$\begin{aligned} x_t &= M_t x_{t-1} + \eta_t, \quad \eta_t \sim \mathcal{N}(0, \Sigma_{t|t-1}) \\ &= M_t(M_{1:t-1} x_0 + \epsilon_{t-1}) + \eta_t, \quad \epsilon_{t-1} \sim \mathcal{N}(0, \Sigma_{t-1}) \\ &= M_{1:t} x_0 + M_t \epsilon_{t-1} + \eta_t. \end{aligned}$$

Hence,

$$\begin{aligned} \mathrm{Cov}(x_t \mid x_0) &= \mathrm{Cov}(M_t \epsilon_{t-1} + \eta_t \mid x_0) \\ &= \mathrm{Cov}(M_t \epsilon_{t-1} \mid x_0) + \mathrm{Cov}(\eta_t) \quad \text{(independence)} \\ &= M_t \Sigma_{t-1} M_t^T + \Sigma_{t|t-1}. \\ \Sigma_t &= M_t \Sigma_{t-1} M_t^T + \Sigma_{t|t-1}, \\ \implies \Sigma_{t|t-1} &= \Sigma_t - M_t \Sigma_{t-1} M_t^T. \end{aligned}$$

□

9.2. Posterior Distribution

Theorem 2 (Posterior Distribution)**.** *Consider the linear generalized linear diffusion process*

$$x_t = M_t x_{t-1} + \eta_t, \qquad \eta_t \sim \mathcal{N}(0, \Sigma_{t|t-1}), \tag{14}$$

with marginals

$$q(x_{t-1} \mid x_0) = \mathcal{N}(\mu_{t-1}, \Sigma_{t-1}), \tag{15}$$

$$q(x_t \mid x_0) = \mathcal{N}(\mu_t, \Sigma_t). \tag{16}$$

Then the posterior distribution

$$q(x_{t-1} \mid x_t, x_0) \tag{17}$$

is Gaussian:

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(\mu_{t \to t-1}, \Sigma_{t \to t-1}), \tag{18}$$

with

$$\Sigma_{t \to t-1} = (\Sigma_{t-1}^{-1} + M_t^T \Sigma_{t|t-1}^{-1} M_t)^{-1}, \tag{19}$$

$$\mu_{t \to t-1} = \Sigma_{t \to t-1} \left(\Sigma_{t-1}^{-1} \mu_{t-1} + M_t^T \Sigma_{t|t-1}^{-1} x_t \right). \tag{20}$$

Proof.

$$q(x_{t-1} \mid x_t, x_0) = \frac{q(x_t \mid x_{t-1}, x_0)\, q(x_{t-1} \mid x_0)}{q(x_t \mid x_0)} = \frac{q(x_t \mid x_{t-1})\, q(x_{t-1} \mid x_0)}{q(x_t \mid x_0)} \tag{a}$$

$$\propto \exp\Bigg(-\Big[(x_t - M_t x_{t-1})^\top \Sigma_{t|t-1}^{-1}(x_t - M_t x_{t-1})\Big] - \Big[(x_{t-1} - \mu_{t-1})^\top \Sigma_{t-1}^{-1}(x_{t-1} - \mu_{t-1})\Big] + \Big[(x_t - \mu_t)^\top \Sigma_t^{-1}(x_t - \mu_t)\Big]\Bigg) \tag{b}$$

$$= \exp\Bigg(-\Big[x_t^T \Sigma_{t|t-1}^{-1} x_t - (x_t^T \Sigma_{t|t-1}^{-1} M_t x_{t-1} + (x_t^T \Sigma_{t|t-1}^{-1} M_t x_{t-1})^T) + x_{t-1}^T M_t^T \Sigma_{t|t-1}^{-1} M_t x_{t-1}\Big] - \Big[x_{t-1}^T \Sigma_{t-1}^{-1} x_{t-1} - (\mu_{t-1}^T \Sigma_{t-1}^{-1} x_{t-1} + (\mu_{t-1}^T \Sigma_{t-1}^{-1} x_{t-1})^T) + \mu_{t-1}^T \Sigma_{t-1} \mu_{t-1}\Big] + C_1(x_0, x_t)\Bigg)$$

$$= \exp\Bigg(-x_{t-1}^T (\Sigma_{t-1}^{-1} + M_t^T \Sigma_{t|t-1}^{-1} M_t) x_{t-1} + \Big[(x_t^T \Sigma_{t|t-1}^{-1} M_t + \mu_{t-1}^T \Sigma_{t-1}^{-1}) x_{t-1} + ((x_t^T \Sigma_{t|t-1}^{-1} M_t + \mu_{t-1}^T \Sigma_{t-1}^{-1}) x_{t-1})^T\Big] + C_2(x_0, x_t)\Bigg)$$

In Eq. a, we first use Bayes' rule, and then use the Markov chain assumption. In Eq. b, we then substitute the marginal (Eq. 3) and forward transition (Eq. 4) distributions. Then we start collecting the terms quadratic (red) and linear (blue) in x_{t-1}. From the quadratic and linear terms, we can complete the square and hence extract the mean and variance of the posterior normal distribution:

$$\begin{aligned}
\Sigma_{t \to t-1} &= (\Sigma_{t-1}^{-1} + M_t^T \Sigma_{t|t-1}^{-1} M_t)^{-1} \\
\mu_{t \to t-1} &= \Sigma_{t \to t-1}(x_t^T \Sigma_{t|t-1}^{-1} M_t + \mu_{t-1}^T \Sigma_{t-1}^{-1})^T \\
&= \Sigma_{t \to t-1}(M_t^T \Sigma_{t|t-1}^{-1} x_t + \Sigma_{t-1}^{-1} \mu_{t-1}).
\end{aligned}$$

The last step comes from the fact that for a symmetric matrix A, $(A^{-1})^T = A^{-1}$, and covariance matrices are symmetric. □

9.3. Posterior Under Isotropic Marginals

Theorem 3 (Closed-Form Posterior Under Isotropic Marginals)**.** *Assume isotropic marginals*

$$\Sigma_t = \sigma_t^2 \mathbf{I}, \qquad \Sigma_{t-1} = \sigma_{t-1}^2 \mathbf{I}. \tag{21}$$

Then the posterior covariance simplifies to

$$\Sigma_{t \to t-1} = \sigma_{t-1}^2 \mathbf{I} - \frac{\sigma_{t-1}^4}{\sigma_t^2} M_t^T M_t, \tag{22}$$

and the posterior mean simplifies to

$$\mu_{t \to t-1} = \mu_{t-1} + \frac{\sigma_{t-1}^2}{\sigma_t^2} M_t^T \left(x_t - M_t \mu_{t-1}\right). \tag{23}$$

Proof.

$$\begin{aligned}
\Sigma_{t\to t-1} &= (\Sigma_{t-1}^{-1} + M_t^T \Sigma_{t|t-1}^{-1} M_t)^{-1} \\
&= \Sigma_{t-1} - \Sigma_{t-1} M_t^T (\Sigma_{t|t-1} + M_t \Sigma_{t-1} M_t^T)^{-1} M_t \Sigma_{t-1} && \text{(c)} \\
&= \Sigma_{t-1} - \Sigma_{t-1} M_t^T \Sigma_t^{-1} M_t \Sigma_{t-1} && \text{(d)} \\
&= \sigma_{t-1}^2 \mathbf{I} - \frac{\sigma_{t-1}^4}{\sigma_t^2} M_t^T M_t && \text{(e)}
\end{aligned}$$

We start from Eq. 5 derived in the previous section. In Eq. c, we used the Woodbury matrix identity $(A+UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1}+VA^{-1}U)^{-1}VA^{-1}$ with $A = \Sigma_{t-1}^{-1}, U = M_t^T, C = \Sigma_{t|t-1}^{-1}, V = M_t$. In Eq. d, we substitute the value of $\Sigma_{t|t-1}$ from Eq. 4. Finally, in Eq. e we substitute the isotropic values for Σ_t and Σ_{t-1}.

$$\begin{aligned}
\mu_{t\to t-1} &= \Sigma_{t\to t-1}(\Sigma_{t-1}^{-1}\mu_{t-1} + M_t^T \Sigma_{t|t-1}^{-1} x_t) = \Sigma_{t\to t-1}\Sigma_{t-1}^{-1}\mu_{t-1} + \Sigma_{t\to t-1} M_t^T \Sigma_{t|t-1}^{-1} x_t \\
&= (\Sigma_{t-1} - \Sigma_{t-1} M_t^T \Sigma_t^{-1} M_t \Sigma_{t-1})\Sigma_{t-1}^{-1}\mu_{t-1} + \Sigma_{t\to t-1} M_t^T \Sigma_{t|t-1}^{-1} x_t && \text{(f)} \\
&= (\mathbf{I} - \Sigma_{t-1} M_t^T \Sigma_t^{-1} M_t)\mu_{t-1} + \Sigma_{t\to t-1} M_t^T \Sigma_{t|t-1}^{-1} x_t \\
&= (\mathbf{I} - \Sigma_{t-1} M_t^T \Sigma_t^{-1} M_t)\mu_{t-1} + (\Sigma_{t-1}^{-1} + M_t^T \Sigma_{t|t-1} M_t)^{-1} M_t^T \Sigma_{t|t-1}^{-1} x_t && \text{(g)} \\
&= (\mathbf{I} - \Sigma_{t-1} M_t^T \Sigma_t^{-1} M_t)\mu_{t-1} + \Sigma_{t-1}^{-1} M_t^T (\Sigma_{t|t-1} + M_t \Sigma_{t-1} M_t^T)^{-1} x_t && \text{(h)} \\
&= (\mathbf{I} - \Sigma_{t-1} M_t^T \Sigma_t^{-1} M_t)\mu_{t-1} + \Sigma_{t-1}^{-1} M_t^T \Sigma_t^{-1} x_t && \text{(i)} \\
&= \mu_{t-1} + \Sigma_{t-1} M_t^T \Sigma_t^{-1}(x_t - M_t \mu_{t-1}) \\
&= \mu_{t-1} + \frac{\sigma_{t-1}^2}{\sigma_t^2} M_t^T (x_t - M_t \mu_{t-1}) && \text{(j)}
\end{aligned}$$

Starting from $\mu_{t\to t-1}$ in Eq. 5, in Eq. f we substitute $\Sigma_{t\to t-1}$ from Eq. d, and then in Eq g we substitute $\Sigma_{t\to t-1}$ from Eq. 5. In Eq. h, we use a corollary of Woodbury identity $(A+UCV)^{-1}UC = A^{-1}U(C^{-1}+VA^{-1}U)^{-1}$, with the same substitution as described above. In Eq. i, we substitute the value of $\Sigma_{t|t-1}$ from Eq. 4, and finally, we substitute the isotropic values for Σ_t and Σ_{t-1}.

□

References

[1] Shady Abu-Hussein and Raja Giryes. Udpm: Upsampling diffusion probabilistic models. *arXiv preprint arXiv:2305.16269*, 2023. 1, 3, 2, 5

[2] Yuval Atzmon, Maciej Bala, Yogesh Balaji, Tiffany Cai, Yin Cui, Jiaojiao Fan, Yunhao Ge, Siddharth Gururani, Jacob Huffman, Ronald Isaac, et al. Edify image: High-quality image generation with pixel space laplacian diffusion models. *arXiv preprint arXiv:2411.07126*, 2024. 1, 2

[3] Arpit Bansal, Eitan Borgnia, Hong-Min Chu, Jie S Li, Hamid Kazemi, Furong Huang, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Cold diffusion: Inverting arbitrary image transforms without noise. *NeurIPS*, 2023. 1

[4] Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A vit backbone for diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22669–22679, 2023. 1

[5] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 2009. 2

[6] Shoufa Chen, Chongjian Ge, Shilong Zhang, Peize Sun, and Ping Luo. Pixelflow: Pixel-space generative models with flow. *arXiv preprint arXiv:2504.07963*, 2025. 1, 2, 4, 8, 3, 5

[7] Katherine Crowson, Stefan Andreas Baumann, Alex Birch, Tanishq Mathew Abraham, Daniel Z Kaplan, and Enrico Shippole. Scalable high-resolution pixel-space image synthesis with hourglass diffusion transformers. In *Forty-first International Conference on Machine Learning*, 2024. 1

[8] Giannis Daras, Mauricio Delbracio, Hossein Talebi, Alex Dimakis, and Peyman Milanfar. Soft diffusion: Score matching with general corruptions. *Transactions on Machine Learning Research*, 2023. 1

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 2, 6

[10] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. *NeurIPS*, 2015. 2

[11] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. 5, 6, 1

[12] Gene H Golub and Gérard Meurant. *Matrices, moments and quadrature with applications*. Princeton University Press, 2009. 5

[13] Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Joshua M Susskind, and Navdeep Jaitly. Matryoshka diffusion models. In *ICLR*, 2024. 2, 1

[14] Moayed Haji-Ali, Willi Menapace, Ivan Skorokhodov, Arpit Sahni, Sergey Tulyakov, Vicente Ordonez, and Aliaksandr Siarohin. Improving progressive generation with decomposable flow matching. *arXiv preprint arXiv:2506.19839*, 2025. 2, 5

[15] 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 *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7441–7451, 2023. 5

[16] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 8

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 1, 2, 3, 6

[18] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *JMLR*, 2022. 2

[19] Emiel Hoogeboom and Tim Salimans. Blurring diffusion models. *ICLR*, 2023. 1, 4, 6, 7

[20] Wongi Jeong, Kyungryeol Lee, Hoigi Seo, and Se Young Chun. Upsample what matters: Region-adaptive latent sampling for accelerated diffusion transformers. *arXiv preprint arXiv:2507.08422*, 2025. 1, 2, 4, 8, 3

[21] Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. *arXiv preprint arXiv:2410.05954*, 2024. 1, 2, 4, 8, 3

[22] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *ICLR*, 2018. 2, 1

[23] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[24] Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of research of the National Bureau of Standards*, 45(4):255–282, 1950. 5

[25] Tony Lindeberg. Scale-space theory: A basic tool for analyzing structures at different scales. *Journal of applied statistics*, 21(1-2):225–270, 1994. 1, 2

[26] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015. 2, 6

[27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6

[28] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, pages 1150–1157. Ieee, 1999. 2

[29] Soumik Mukhopadhyay, Matthew Gwilliam, Yosuke Yamaguchi, Vatsal Agarwal, Namitha Padmanabhan, Archana Swaminathan, Tianyi Zhou, Jun Ohya, and Abhinav Shrivastava. Do text-free diffusion models learn discriminative visual representations? In *European Conference on Computer Vision*, pages 253–272. Springer, 2024. 1

[30] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021. 6

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming

Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 4

[32] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023. 2, 1

[33] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 2, 1, 5

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 2, 1

[35] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022. 5

[36] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 2

[37] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *ICLR*, 2021. 1

[38] Jiayan Teng, Wendi Zheng, Ming Ding, Wenyi Hong, Jianqiao Wangni, Zhuoyi Yang, and Jie Tang. Relay diffusion: Unifying diffusion process across resolutions for image synthesis. *arXiv preprint arXiv:2309.03350*, 2023. 2

[39] Ye Tian, Xin Xia, Yuxi Ren, Shanchuan Lin, Xing Wang, Xuefeng Xiao, Yunhai Tong, Ling Yang, and Bin Cui. Training-free diffusion acceleration with bottleneck sampling. *arXiv preprint arXiv:2503.18940*, 2025. 2

[40] XIANG Weilai. FutureXiang/Diffusion, 2025. original-date: 2022-10-18T11:42:46Z. 2