Diffusion Study Group #13

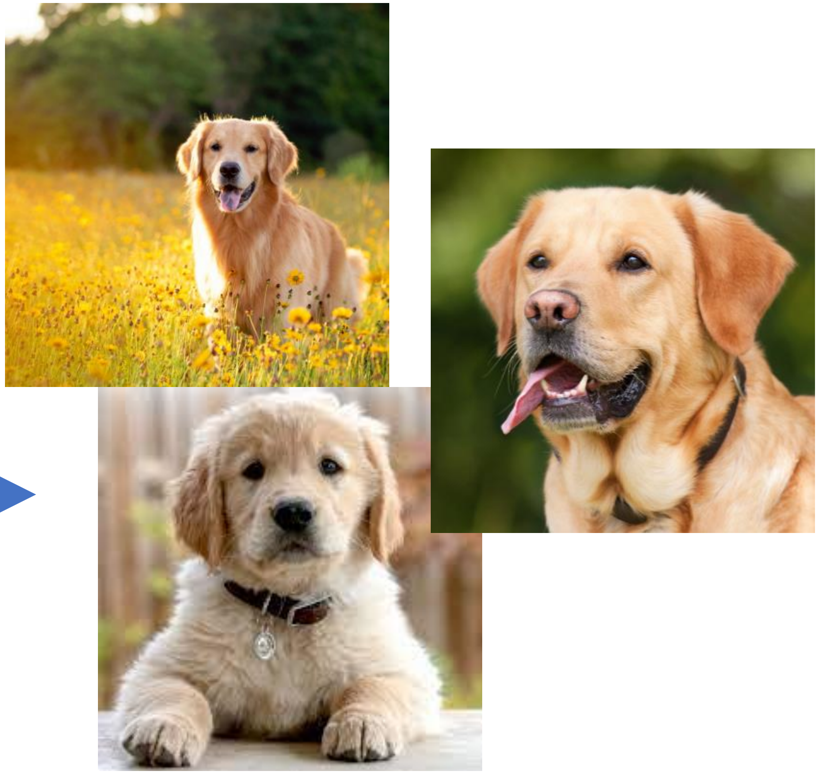
Tanishq Abraham

1/14/2023

Recap of diffusion models

What's the task?

Given these datapoints…



Can we generate more like it?

Forward process

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right)$$

$$q(\mathbf{x}_t|\mathbf{x}_0) := \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t) \mathbf{I}\right)$$

$\mathbf{x}_{25}$ | $\mathbf{x}_{0}$ / $\mathbf{x}_{50}$ | $\mathbf{x}_{75}$ | $\mathbf{x}_{100} = \mathbf{x}_T$

Observed image

Equivalent to Gaussian noise

Reverse process

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)\right)$$

$\mathbf{x}_0$ $\mathbf{x}_{25}$ $\mathbf{x}_{50}$ $\mathbf{x}_{75}$ $\mathbf{x}_{100} = \mathbf{x}_T$

Generated image!

Equivalent to Gaussian noise

What is score matching?

If the data distribution is $p(\mathbf{x})$, then the score function is defined as

$$\nabla_{\mathbf{x}} \log p(\mathbf{x})$$

Note that if $p(\mathbf{x}) = \frac{e^{-f(\mathbf{x})}}{Z}$ (Z is our normalizing constant that makes density estimation intractable), then:

$$\nabla_{\mathbf{x}} \log p\,(\mathbf{x}) = -\nabla_{\mathbf{x}} f\,(\mathbf{x}) - \underbrace{\nabla_{\mathbf{x}} \log Z}_{=0} = -\nabla_{\mathbf{x}} f\,(\mathbf{x})$$

Don't need Z!

Modeling the score function → score-based model

$$\mathbf{s}_\theta(\mathbf{x}) \approx \log p(\mathbf{x})$$

Trained with the following objective:

$$\mathbb{E}_{p(\mathbf{x})}\left[\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2\right]$$

Used for training energy-based models

Denoising score matching

Score matching of the perturbed distribution:

$$\mathcal{L}_{DSM} = \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}})}\left[\|\nabla_{\mathbf{x}} \log q_\sigma(\tilde{\mathbf{x}}) - \mathbf{s}_\theta(\tilde{\mathbf{x}})\|_2^2\right]$$

The following objective is equivalent!

$$\mathcal{L}_{DSM} = \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}},\mathbf{x})}\left[\|\nabla_{\mathbf{x}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) - \mathbf{s}_\theta(\tilde{\mathbf{x}})\|_2^2\right]$$

Since $\log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = -\frac{1}{2\sigma^2}(\tilde{\mathbf{x}} - \mathbf{x})^2$, then $\nabla_{\mathbf{x}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = -\frac{1}{\sigma^2}(\tilde{\mathbf{x}} - \mathbf{x})$

Final objective is:

$$\mathcal{L}_{DSM} = \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}},\mathbf{x})}\left[\left\|\frac{1}{\sigma^2}(\tilde{\mathbf{x}} - \mathbf{x}) + \mathbf{s}_\theta(\tilde{\mathbf{x}})\right\|_2^2\right]$$

Tweedie's formula - optimal denoising function $f^*(\tilde{\mathbf{x}}) = \mathbf{x} \approx \tilde{\mathbf{x}} + \sigma^2 \nabla_{\tilde{\mathbf{x}}} \log p(\tilde{\mathbf{x}})$

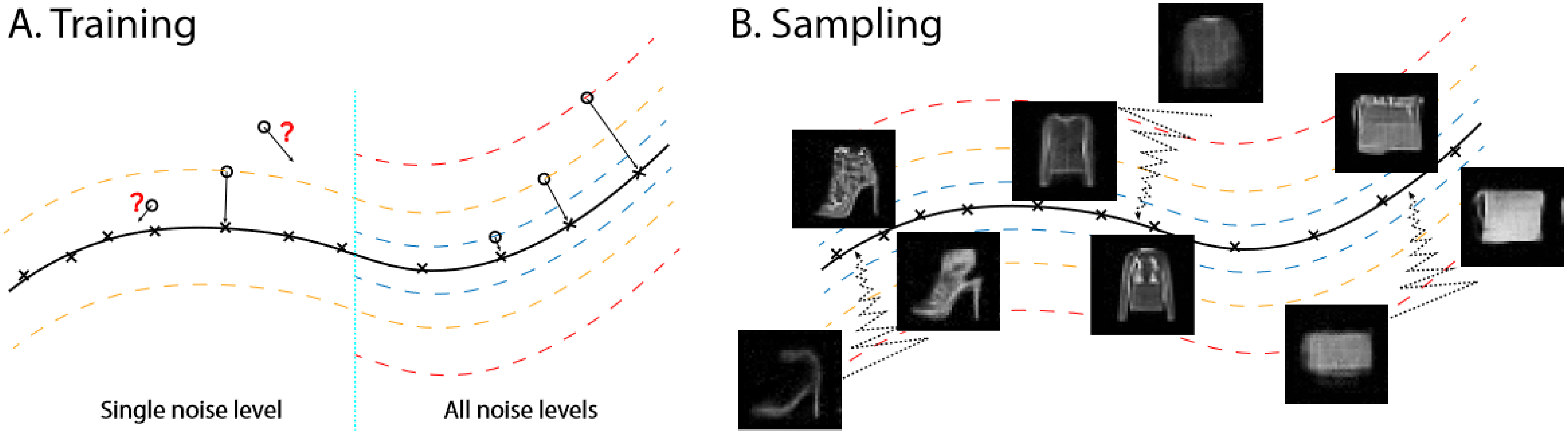
Denoising score matching written in DDPM notation

Denoising is equivalent to score matching:

$$\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|\mathbf{x}_0) = -\frac{1}{(1-\bar{\alpha}_t)}(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0) = -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}$$

$$\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) = -\sqrt{1-\bar{\alpha}_t}\,\mathbf{s}_\theta(\mathbf{x}_t, t)$$

Score matching perspective - Learning the data manifold!



OpenAI's ADM - Architectural Improvements

Timestep+label embeddings are incorporated through shift and scaling of the group normalization

$$AdaGN(h, y) = y_s \mathrm{GroupNorm}(h) + y_b$$

where h are the intermediate activations of a residual block, and $y = [y_s, y_b]$ are obtained from a linear projection of the embeddings

Ablated Diffusion Model (ADM):

- Variable width with 2 residual blocks per resolution
- multiple heads with 64 channels per head
- attention at 32, 16 and 8 resolutions
- BigGAN residual blocks for up and downsampling
- AdaGN for injecting timestep+label embeddings into residual blocks.

Classifier Guidance

Mathematical derivation in paper demonstrates that the mean for the reverse process can be updated to be:

$$\mu_y = \mu + \Sigma g$$

where $g = \nabla_{x_t} \log p_\phi(y|x_t)$ (the gradient of the classifier output w.r.t. the input image x_t)

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

```
Input: class label y, gradient scale s
x_T ← sample from N(0, I)
for all t from T to 1 do
    μ, Σ ← μ_θ(x_t), Σ_θ(x_t)
    x_{t-1} ← sample from N(μ + sΣ ∇_{x_t} log p_φ(y|x_t), Σ)
end for
return x_0
```

Classifier Guidance

The score-based formulation allows us to easily modify the DDIM sampling for classifier guidance. Specifically:

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

So we can derive an updated noise predictor function to use for classifier guidance:

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

Classifier-free Guidance

Using Bayes' Rule, we construct an implicit classifier from our conditional generative model, and use that for classifier guidance. We get the following updated model:

$$\tilde{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, \mathbf{c}) = (1 + w)\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{c}) - w\boldsymbol{\epsilon}_\theta(\mathbf{x}_t)$$

V-prediction, an alternative model objective

From Progressive Distillation paper

$$\text{Predicting } \mathbf{v} \equiv \alpha_t \epsilon - \sigma_t \mathbf{x}, \text{ which gives } \hat{\mathbf{x}} = \alpha_t \mathbf{z}_t - \sigma_t \hat{\mathbf{v}}_\theta(\mathbf{z}_t)$$

$$L_\theta = \|\mathbf{v}_t - \hat{\mathbf{v}}_t\|_2^2 = (1 + \frac{\alpha_t^2}{\sigma_t^2})\|\mathbf{x} - \hat{\mathbf{x}}_t\|_2^2; \text{ `SNR+1' weighting.}$$

Similar to the objective used in EDM (Karras et al.)

High-Resolution Image Synthesis with Latent Diffusion Models

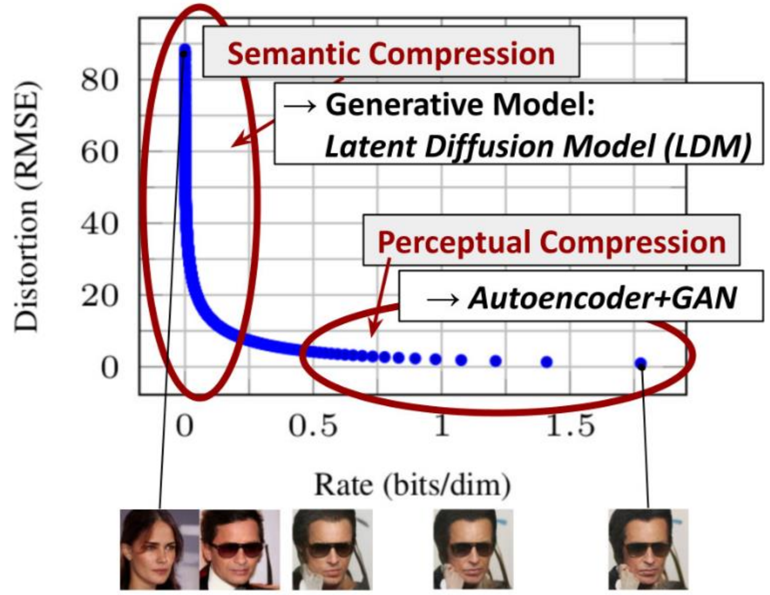
Rombach and Blattmann et al.

Departure to Latent Space

- Training and inference requires repeated function evaluations (and gradient computations) in the high-dimensional space of RGB images.

- Two stages of learning: perceptual compression and semantic compression

- Divide training accordingly: autoencoder + diffusion model



Perceptual Image Compression - autoencoder

Autoencoder trained with discriminator and perceptual loss to maintain good perceptual quality at increased compression rate

For an image $x \in \mathbb{R}^{H \times W \times 3}$ the encoder $\mathcal{E}$ encodes x into a latent $z = \mathcal{E}(x)$, and the decoder $\mathcal{D}$ reconstructs the image from the latent

$$\tilde{x} = \mathcal{D}(z) = \mathcal{D}\big(\mathcal{E}(x)\big) \approx x$$

where $z \in \mathbb{R}^{h \times w \times c}$

The encoder downsamples the image by a factor $f = \frac{H}{h} = \frac{W}{w}$, and different factors $f = 2^m$ are studied in the paper

Perceptual Image Compression - autoencoder

The reconstruction loss is the MAE + LPIPS perceptual loss.

A patch-based discriminator D_ψ is optimized to differentiate original images from reconstructions $\mathcal{D}(\mathcal{E}(x))$.

$$\mathcal{L}_{\mathrm{GAN}}(\{E, G, \mathcal{Z}\}, D) = [\log D(x) + \log(1 - D(\hat{x}))]$$

The GAN loss is adaptively weighted:

$$\lambda = \frac{\nabla_{G_L}[\mathcal{L}_{\mathrm{rec}}]}{\nabla_{G_L}[\mathcal{L}_{\mathrm{GAN}}] + \delta}$$

Perceptual Image Compression - autoencoder

To avoid arbitrarily scaled latent spaces, the latent is regularized to be zero centered with low variance. This is done by a regularizing loss term L_{reg}

Two options are tested:

1. Kullback-Leibler divergence between $q_{\mathcal{E}}(z|x) = \mathcal{N}\left(z; \mathcal{E}_{\mu}, \mathcal{E}_{\sigma^2}\right)$ (low weight used)

   This makes the autoencoder a VAE!

1. Regularize the latent space with a vector quantization layer by learning a codebook of a total of $|\mathcal{Z}|$ codes (high dimensionality used)

   This makes the autoencoder a VQVAE (same set up as VQGAN)!

Perceptual Image Compression - autoencoder

Full objective:

$$L_{autoencoder} = \min_{\mathcal{E},\mathcal{D}} \max_{\psi} \left(L_{rec}\left(x, \mathcal{D}\big(\mathcal{E}(x)\big)\right) - \lambda L_{GAN}\left(\mathcal{D}\big(\mathcal{E}(x)\big)\right) + L_{reg}(x; \mathcal{E}, \mathcal{D}) \right)$$

f	$\lvert\mathcal{Z}\rvert$	c	**R-FID** ↓	**R-IS** ↑	**PSNR** ↑	**PSIM** ↓	**SSIM** ↑
16 *VQGAN* [23]	16384	256	4.98	–	19.9 ± 3.4	1.83 ± 0.42	0.51 ± 0.18
16 *VQGAN* [23]	1024	256	7.94	–	19.4 ± 3.3	1.98 ± 0.43	0.50 ± 0.18
8 *DALL-E* [66]	8192	-	32.01	–	22.8 ± 2.1	1.95 ± 0.51	0.73 ± 0.13
32	16384	16	31.83	40.40 ± 1.07	17.45 ± 2.90	2.58 ± 0.48	0.41 ± 0.18
16	16384	8	5.15	144.55 ± 3.74	20.83 ± 3.61	1.73 ± 0.43	0.54 ± 0.18
8	16384	4	1.14	201.92 ± 3.97	23.07 ± 3.99	1.17 ± 0.36	0.65 ± 0.16
8	256	4	1.49	194.20 ± 3.87	22.35 ± 3.81	1.26 ± 0.37	0.62 ± 0.16
4	8192	3	0.58	224.78 ± 5.35	27.43 ± 4.26	0.53 ± 0.21	0.82 ± 0.10
4†	8192	3	1.06	221.94 ± 4.58	25.21 ± 4.17	0.72 ± 0.26	0.76 ± 0.12
4	256	3	0.47	223.81 ± 4.58	26.43 ± 4.22	0.62 ± 0.24	0.80 ± 0.11
2	2048	2	0.16	232.75 ± 5.09	30.85 ± 4.12	0.27 ± 0.12	0.91 ± 0.05
2	64	2	0.40	226.62 ± 4.83	29.13 ± 3.46	0.38 ± 0.13	0.90 ± 0.05
32	KL	64	2.04	189.53 ± 3.68	22.27 ± 3.93	1.41 ± 0.40	0.61 ± 0.17
32	KL	16	7.3	132.75 ± 2.71	20.38 ± 3.56	1.88 ± 0.45	0.53 ± 0.18
16	KL	16	0.87	210.31 ± 3.97	24.08 ± 4.22	1.07 ± 0.36	0.68 ± 0.15
16	KL	8	2.63	178.68 ± 4.08	21.94 ± 3.92	1.49 ± 0.42	0.59 ± 0.17
8	KL	4	0.90	209.90 ± 4.92	24.19 ± 4.19	1.02 ± 0.35	0.69 ± 0.15
4	KL	3	0.27	227.57 ± 4.89	27.53 ± 4.54	0.55 ± 0.24	0.82 ± 0.11
2	KL	2	0.086	232.66 ± 5.16	32.47 ± 4.19	0.20 ± 0.09	0.93 ± 0.04

Table 8. Complete autoencoder zoo trained on OpenImages, evaluated on ImageNet-Val. † denotes an attention-free autoencoder.

Latent Diffusion Model

With the autencoder trained, the diffusion model can be trained separately to denoise the latents:

$$L_{LDM} := \mathbb{E}_{\mathcal{E}(x), \epsilon \sim \mathcal{N}(0,1), t}\Big[\|\epsilon - \epsilon_\theta(z_t, t)\|_2^2\Big]$$

Conditioning Mechanisms

Turn diffusion models into more flexible conditional image generators by augmenting their underlying U-Net backbone with the cross-attention mechanism

Introduce a domain-specific encoder τ_θ that encodes the conditioning y (such as text) to an intermediate representation $\tau_\theta(y) \in \mathbb{R}^{M \times d_\tau}$

This is introduced into the U-net via a cross-attention layer

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) \cdot V$$

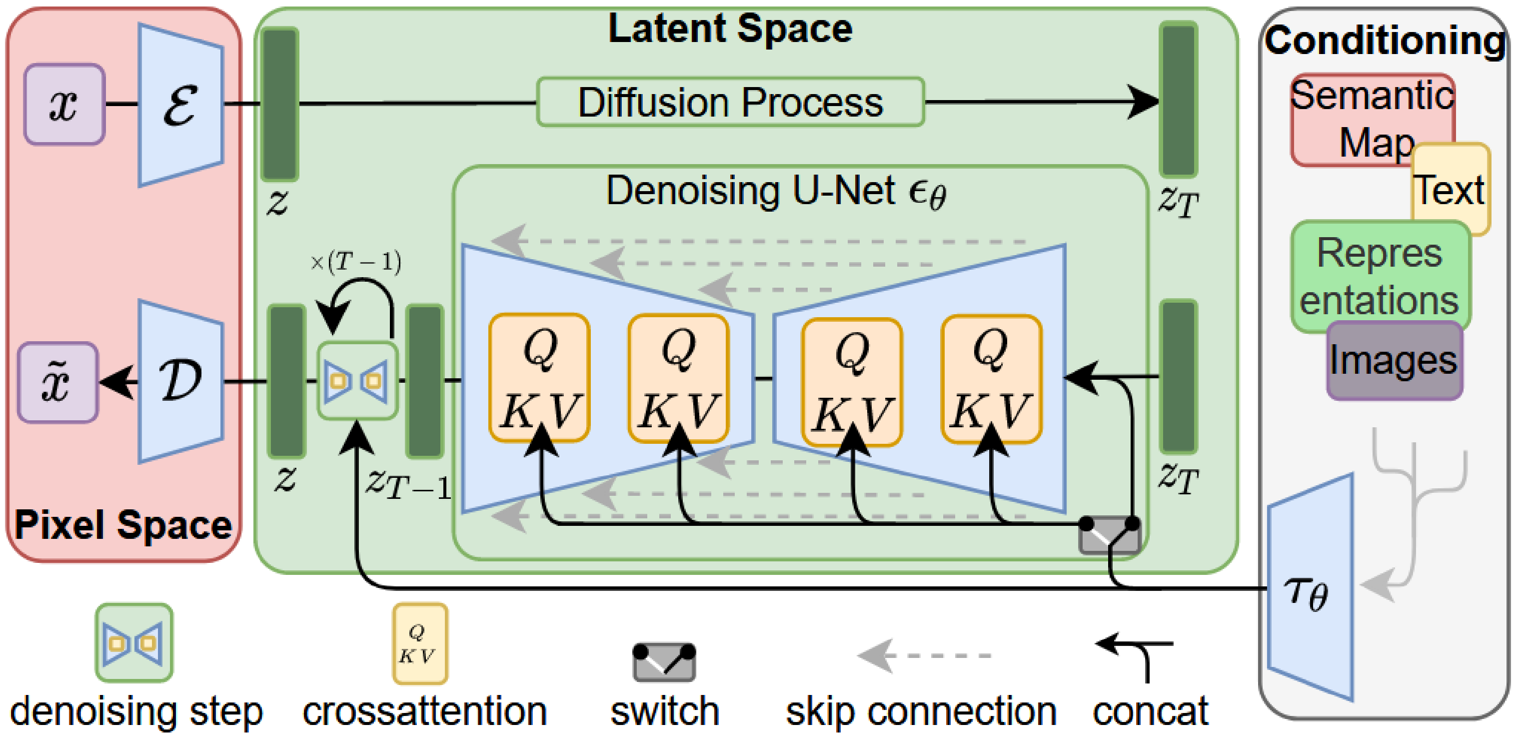
$$Q = W_Q^{(i)} \cdot \varphi_i(z_t),\ K = W_K^{(i)} \cdot \tau_\theta(y),\ V = W_V^{(i)} \cdot \tau_\theta(y)$$

where $\varphi_i(z_t) \in \mathbb{R}^{N \times d_\epsilon^i}$ is the flattened intermediate U-net representations and W are learnable weights

Latent Diffusion Model

Final objective (with conditioning):

$$L_{LDM} := \mathbb{E}_{\mathcal{E}(x),y,\epsilon\sim\mathcal{N}(0,1),t}\Big[\|\epsilon - \epsilon_\theta(z_t, t, \tau_\theta(y))\|_2^2\Big]$$

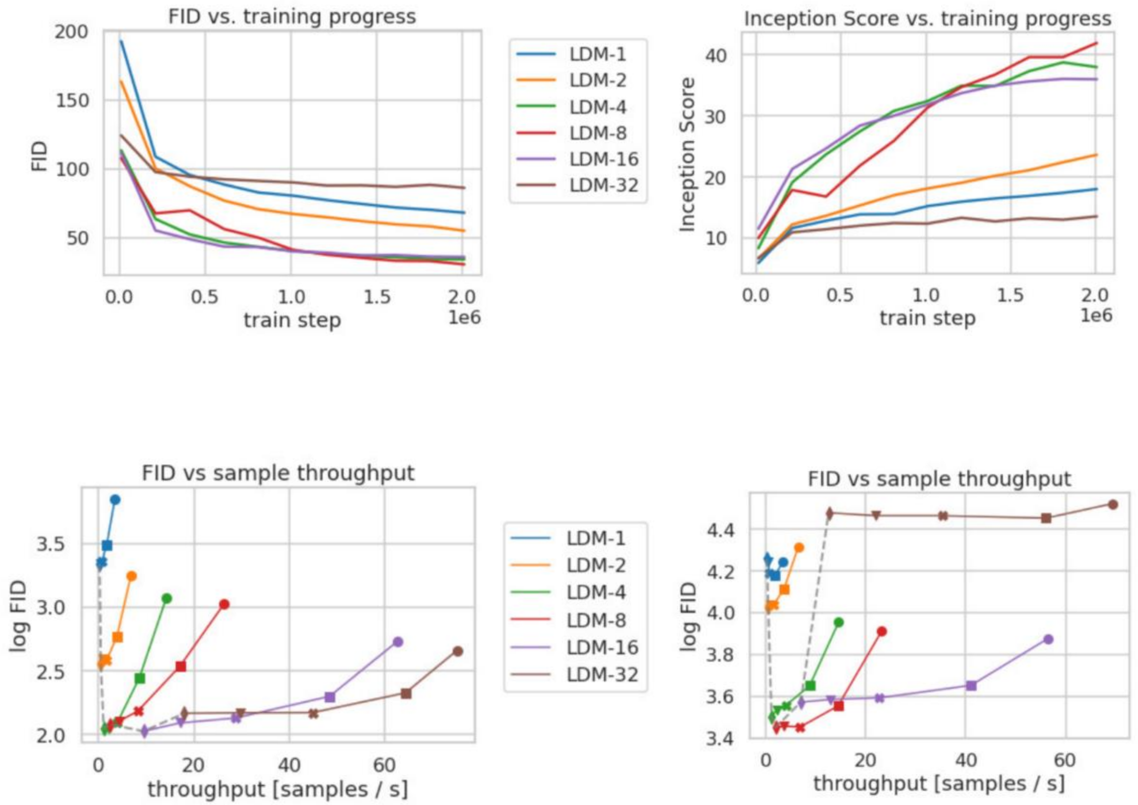


Results – Perceptual Compression Tradeoffs

All models trained on a single A100

VQGAN used since achieves better sample quality, though reconstruction is worse!

LDM-4,8 provides the best tradeoff between speed and generation quality



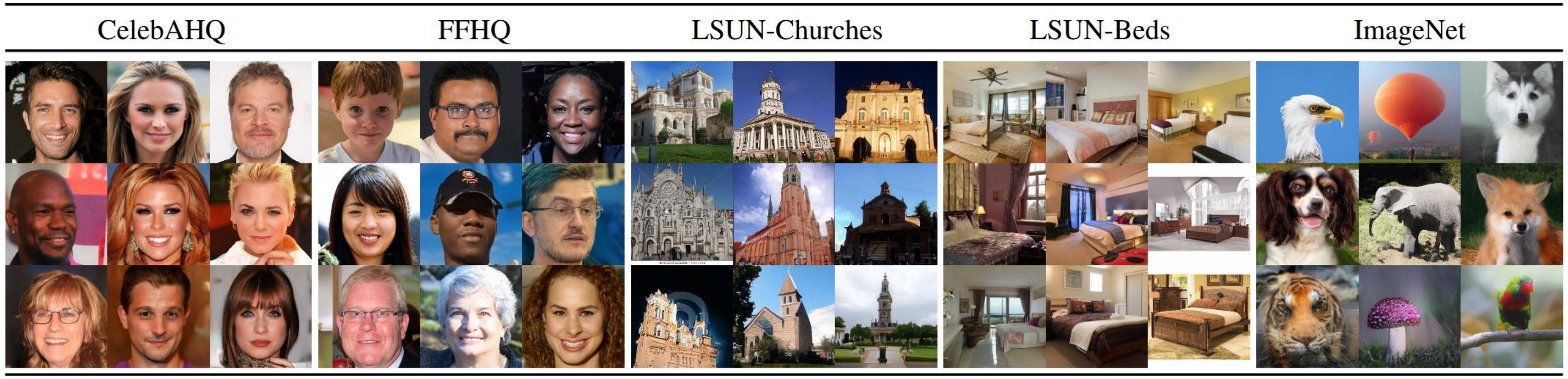
Results – Image Generation with Latent Diffusion

New SOTA on CelebA-HQ, close to ADM on LSUN-Bedrooms while using half its parameters and 4x less training resources.

Outperforms Latent Space Generative Model, suggesting decoupling the autencoder and LDM training is an easier task

CelebA-HQ 256 × 256				FFHQ 256 × 256			
Method	FID ↓	Prec. ↑	Recall ↑	**Method**	FID ↓	Prec. ↑	Recall ↑
DC-VAE [63]	15.8	-	-	ImageBART [21]	9.57	-	-
VQGAN+T. [23] (k=400)	10.2	-	-	U-Net GAN (+aug) [77]	10.9 (7.6)	-	-
PGGAN [39]	8.0	-	-	UDM [43]	5.54	-	-
LSGM [93]	7.22	-	-	StyleGAN [41]	4.16	0.71	0.46
UDM [43]	7.16	-	-	ProjectedGAN [76]	**3.08**	0.65	0.46
LDM-4 (ours, 500-s[†])	**5.11**	0.72	0.49	*LDM-4* (ours, 200-s)	4.98	**0.73**	**0.50**

LSUN-Churches 256 × 256				LSUN-Bedrooms 256 × 256			
Method	FID ↓	Prec. ↑	Recall ↑	**Method**	FID ↓	Prec. ↑	Recall ↑
DDPM [30]	7.89	-	-	ImageBART [21]	5.51	-	-
ImageBART [21]	7.32	-	-	DDPM [30]	4.9	-	-
PGGAN [39]	6.42	-	-	UDM [43]	4.57	-	-
StyleGAN [41]	4.21	-	-	StyleGAN [41]	2.35	0.59	0.48
StyleGAN2 [42]	3.86	-	-	ADM [15]	1.90	**0.66**	**0.51**
ProjectedGAN [76]	**1.59**	0.61	0.44	ProjectedGAN [76]	**1.52**	0.61	0.34
*LDM-8** (ours, 200-s)	4.02	**0.64**	**0.52**	*LDM-4* (ours, 200-s)	2.95	**0.66**	0.48



Results – Conditional Latent Diffusion

Trained a 1.45B parameter KL-regularized LDM conditioned on language prompts on LAION-400M. A BERT-tokenizer for the text was used and τ_θ was implemented as a transformer.

CFG greatly improves performance, comparable to SOTA methods with limited parameters.

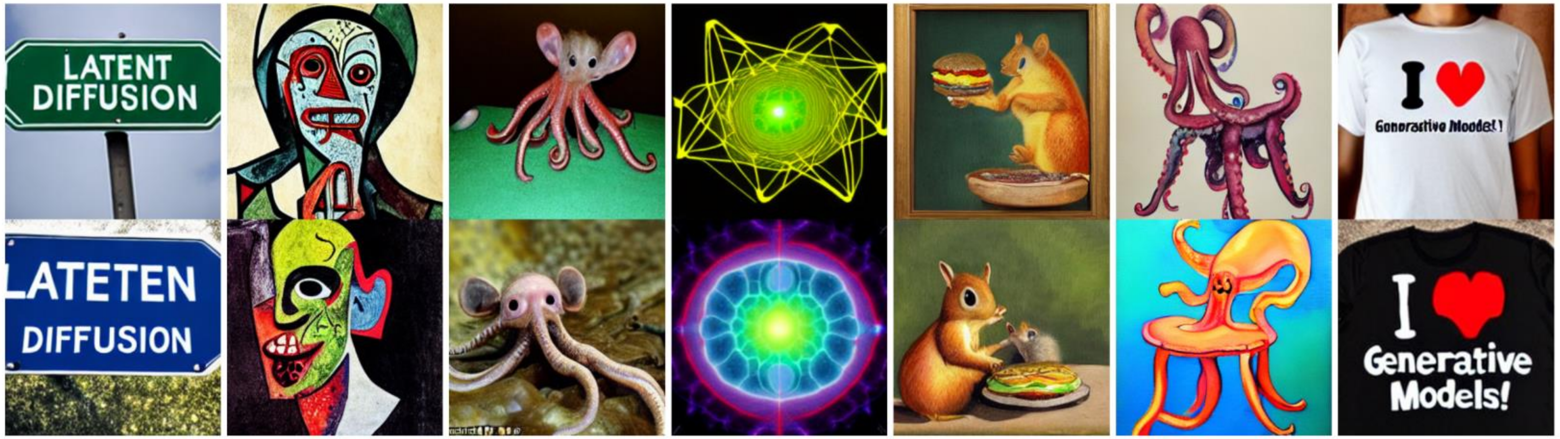
SOTA on semantic layout-to-image synthesis and class-conditioned ImageNet synthesis (beating ADM)

Text-Conditional Image Synthesis

Method	FID ↓	IS↑	N_{params}	
CogView† [17]	27.10	18.20	4B	self-ranking, rejection rate 0.017
LAFITE† [109]	26.94	<u>26.02</u>	75M	
GLIDE* [59]	<u>12.24</u>	-	6B	277 DDIM steps, c.f.g. [32] $s = 3$
Make-A-Scene* [26]	**11.84**	-	4B	c.f.g for AR models [98] $s = 5$
LDM-KL-8	23.31	20.03$\pm$0.33	1.45B	250 DDIM steps
*LDM-KL-8-G**	12.63	**30.29**$\pm$0.42	1.45B	250 DDIM steps, c.f.g. [32] $s = 1.5$

Text-to-Image Synthesis on LAION. 1.45B Model.

'A street sign that reads "Latent Diffusion" ' | *'A zombie in the style of Picasso'* | *'An image of an animal half mouse half octopus'* | *'An illustration of a slightly conscious neural network'* | *'A painting of a squirrel eating a burger'* | *'A watercolor painting of a chair that looks like an octopus'* | *'A shirt with the inscription: "I love generative models!" '*



Results – Conditional Latent Diffusion

Latent Diffusion generalizes to larger resolution!

Figure 9. A *LDM* trained on 256^2 resolution can generalize to larger resolution (here: 512×1024) for spatially conditioned tasks such as semantic synthesis of landscape images. See Sec. 4.3.2.

Results – Super-Resolution with Latent Diffusion

Model trained on ImageNet with same image degradation pipeline as SR3

LDM-SR outperforms SR3 in FID while SR3 has a better IS

Method	FID ↓	IS ↑	PSNR ↑	SSIM ↑	N_{params}	$[\frac{\text{samples}}{s}](^*)$
Image Regression [72]	15.2	121.1	**27.9**	**0.801**	625M	N/A
SR3 [72]	5.2	**180.1**	26.4	0.762	625M	N/A
LDM-4 (ours, 100 steps)	$2.8^{\dagger}/4.8^{\ddagger}$	166.3	24.4 ± 3.8	0.69 ± 0.14	**169M**	4.62
emphLDM-4 (ours, big, 100 steps)	$\mathbf{2.4}^{\dagger}/\mathbf{4.3}^{\ddagger}$	174.9	24.7 ± 4.1	0.71 ± 0.15	552M	4.5
LDM-4 (ours, 50 steps, guiding)	$4.4^{\dagger}/6.4^{\ddagger}$	153.7	25.8 ± 3.7	0.74 ± 0.12	184M	0.38

Table 5. $\times 4$ upscaling results on ImageNet-Val. (256^2); †: FID features computed on validation split, ‡: FID features computed on train split; *: Assessed on a NVIDIA A100

Figure 10. ImageNet 64→256 super-resolution on ImageNet-Val. *LDM-SR* has advantages at rendering realistic textures but SR3 can synthesize more coherent fine structures. See appendix for additional samples and cropouts. SR3 results from [72].

Results – Inpainting with Latent Diffusion

A new SOTA FID for inpainting

Method	40-50% masked		All samples	
	FID ↓	LPIPS ↓	FID ↓	LPIPS ↓
LDM-4 (ours, big, w/ ft)	**9.39**	0.246± 0.042	**1.50**	0.137± 0.080
LDM-4 (ours, big, w/o ft)	12.89	0.257± 0.047	2.40	0.142± 0.085
LDM-4 (ours, w/ attn)	11.87	0.257± 0.042	2.15	0.144± 0.084
LDM-4 (ours, w/o attn)	12.60	0.259± 0.041	2.37	0.145± 0.084
LaMa [88]†	12.31	**0.243**± 0.038	2.23	**0.134**± 0.080
LaMa [88]	12.0	**0.24**	2.21	0.14
CoModGAN [107]	10.4	0.26	1.82	0.15
RegionWise [52]	21.3	0.27	4.75	0.15
DeepFill v2 [104]	22.1	0.28	5.20	0.16
EdgeConnect [58]	30.5	0.28	8.37	0.16

Table 7. Comparison of inpainting performance on 30k crops of size 512 × 512 from test images of Places [108]. The column *40-50%* reports metrics computed over hard examples where 40-50% of the image region have to be inpainted. †recomputed on our test set, since the original test set used in [88] was not available.

Stable Diffusion

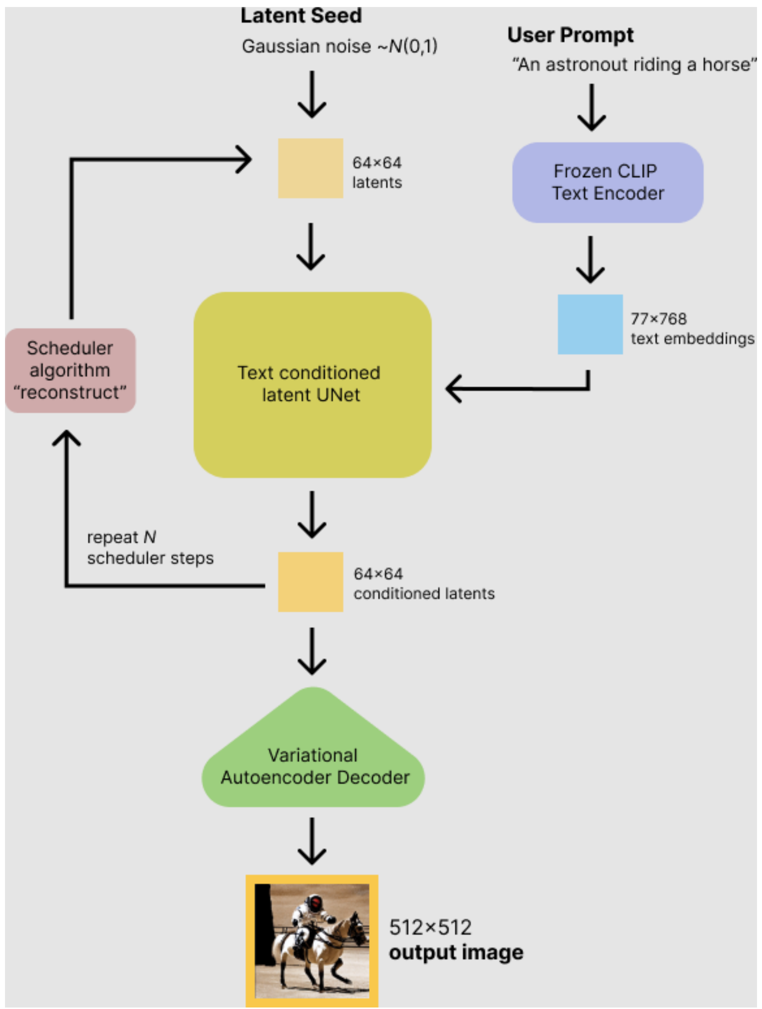
Robin Rombach and Patrick Esser

Components of Stable Diffusion

Main difference from latent diffusion is the text conditioning! CLIP text encoder, gives (77, 1024) embedding (not pooled)

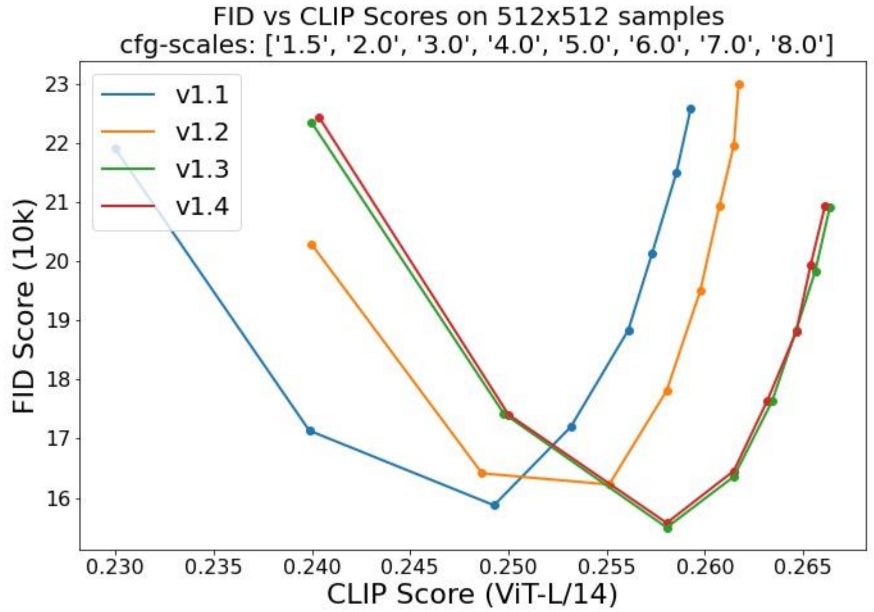
The autoencoder used is a KL, f=8 pretrained on OpenImages

Different versions: v1.1 → v2.1 (so far)



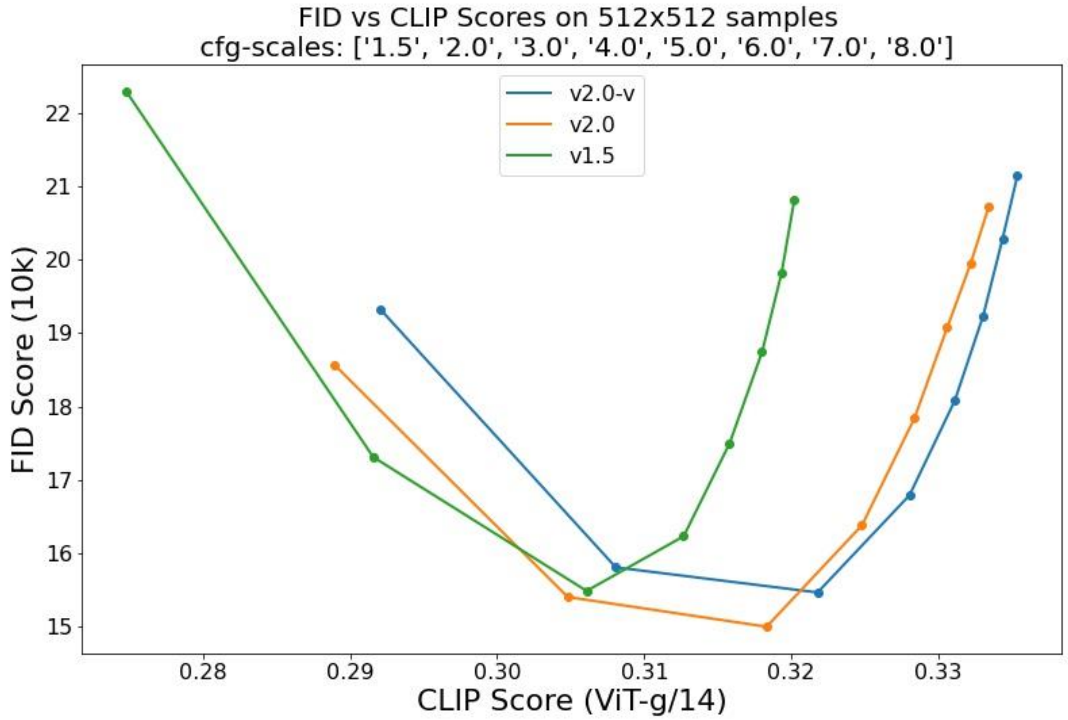
Differences between versions (1.x)

V1.x – uses CLIP ViT-L/14 text encoder, 860M U-net and noise prediction	
V1.1	trained on 237,000 steps at resolution 256x256 on laion2B-en, followed by 194,000 steps at resolution 512x512 on laion-high-resolution (170M examples from LAION-5B with resolution >= 1024x1024)
V1.2	Resumed from stable-diffusion-v1-1. 515,000 steps at resolution 512x512 on "laion-improved-aesthetics"
V1.3	Resumed from stable-diffusion-v1-2. 195,000 steps at resolution 512x512 on "laion-improved-aesthetics" and 10 % dropping of the text-conditioning to improve classifier-free guidance sampling.
V1.4	Resumed from stable-diffusion-v1-2.225,000 steps at resolution 512x512 on "laion-aesthetics v2 5+" and 10 % dropping of the text-conditioning to improve classifier-free guidance sampling.
V1.5	Resumed from stable-diffusion-v1-2 - 595,000 steps at resolution 512x512 on "laion-aesthetics v2 5+" and 10 % dropping of the text-conditioning to improve [classifier-free guidance sampling].



Differences between versions (2.x)

V2.x – uses OpenCLIP-ViT/H text encoder, 865M U-net and v-prediction	
V2-base	550k steps at resolution 256x256 on a subset of LAION-5B filtered for explicit pornographic material, using the LAION-NSFW classifier with punsafe=0.1 and an aesthetic score >= 4.5. 850k steps at resolution 512x512 on the same dataset with resolution >= 512x512.
V2	Resumed from 512-base-ema.ckpt and trained for 150k steps using a v-objective on the same dataset. Resumed for another 140k steps on a 768x768 subset of our dataset.
V2.1-base	Fine-tuned on 512-base-ema.ckpt 2.0 with 220k extra steps taken, with punsafe=0.98 on the same dataset.
V2.1	Resumed from 768-v-ema.ckpt 2.0 with an additional 55k steps on the same dataset (punsafe=0.1), and then fine-tuned for another 155k extra steps with punsafe=0.98.



Other models and approaches available

Models available:

- Inpainting models
- Depth-conditioned model
- 4x Upscaler
- Fine-tuned decoder
- Distilled model coming soon!

img2img – SDEdit (discussed earlier)

CLIP-guided Stable Diffusion – simple classifier guidance

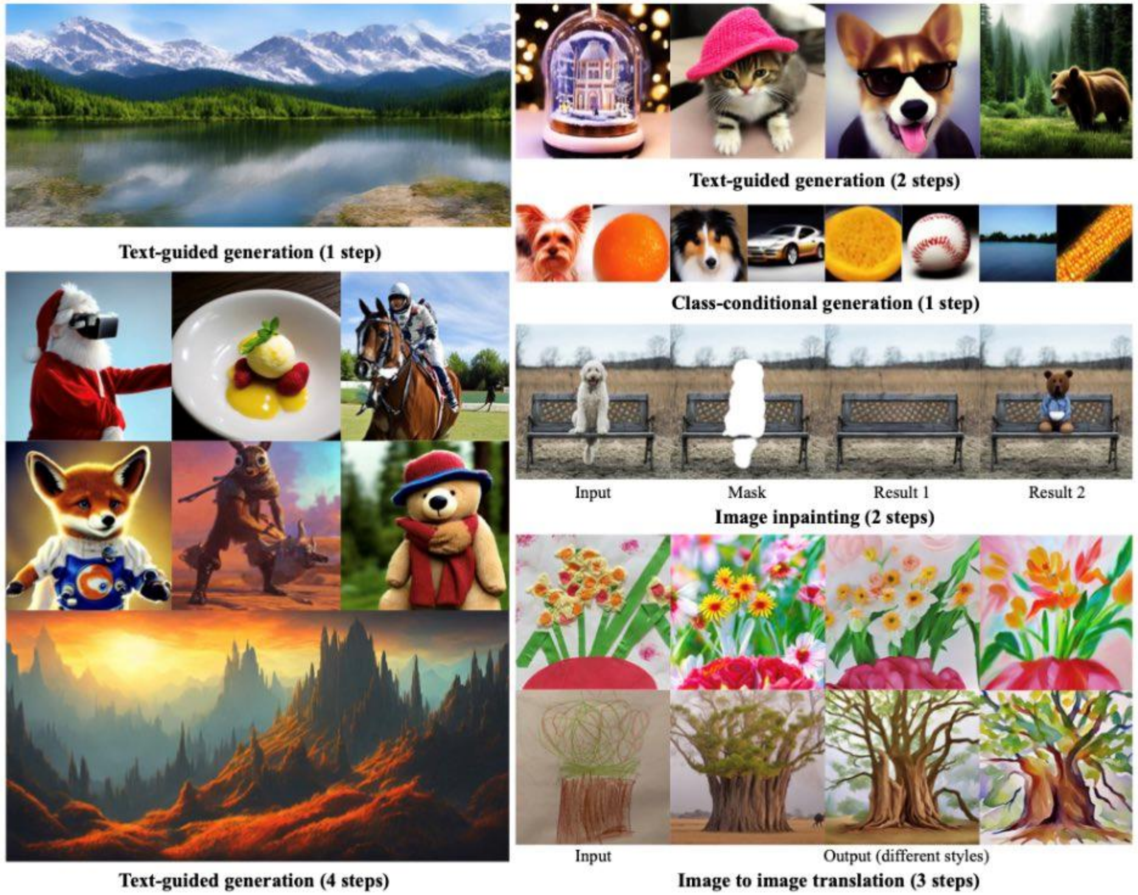


Figure 1. Distilled Stable Diffusion samples generated by our method. Our two-stage distillation approach is able to generate realistic images using only 1 to 4 denoising steps on various tasks. Compared to standard classifier-free guided diffusion models, we reduce the total number of sampling steps by at least 20×.