DIFFUSION MODELS

CSE 705 SEMINAR (FALL 2024) DAY 2

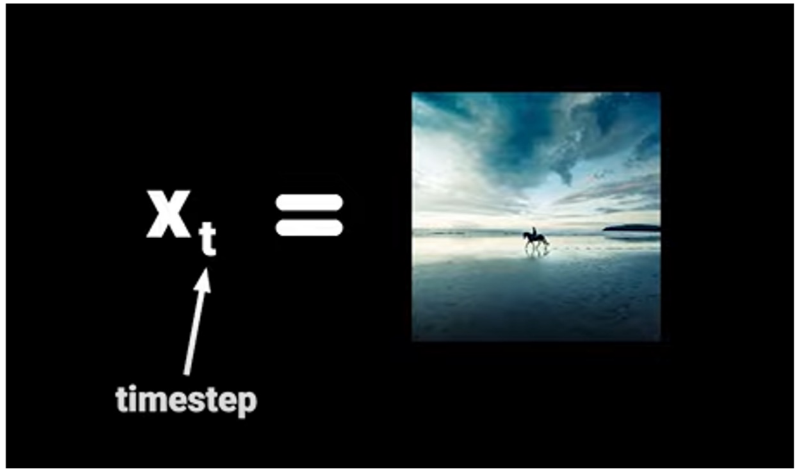
Presented By:
Team Sushi (Vamshi Krishna Kyatham, Xiaofeng Chen, Rahul Dasari)

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

LET’S CONTINUE WHERE WE LEFT FROM..

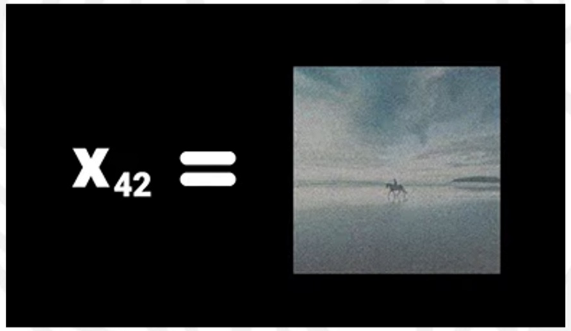
MATH BEHIND DDPM

Let’s fix some Notations

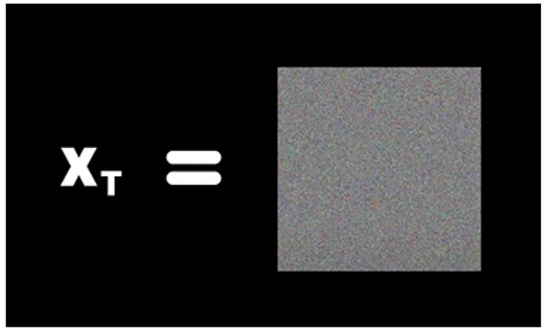


For Example: The Original Image at timestep 0 is:

The image after iteratively adding noise in the forward process at timestep 42 is:



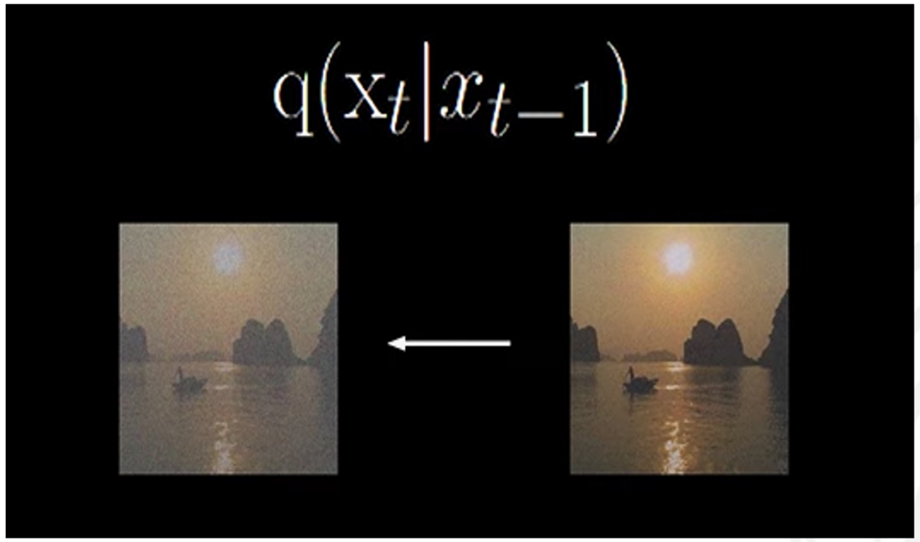
The final image after following isotropic Gaussian Distribution:



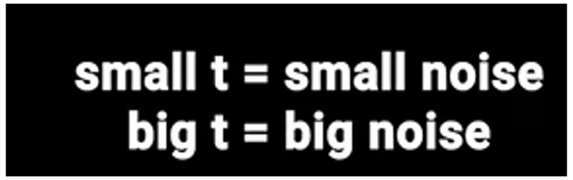
Initially in the base paper T was set to 1000

Now let’s define some functions

Forward Process: $q(x_t \mid x_{t-1})$ defines the forward process.

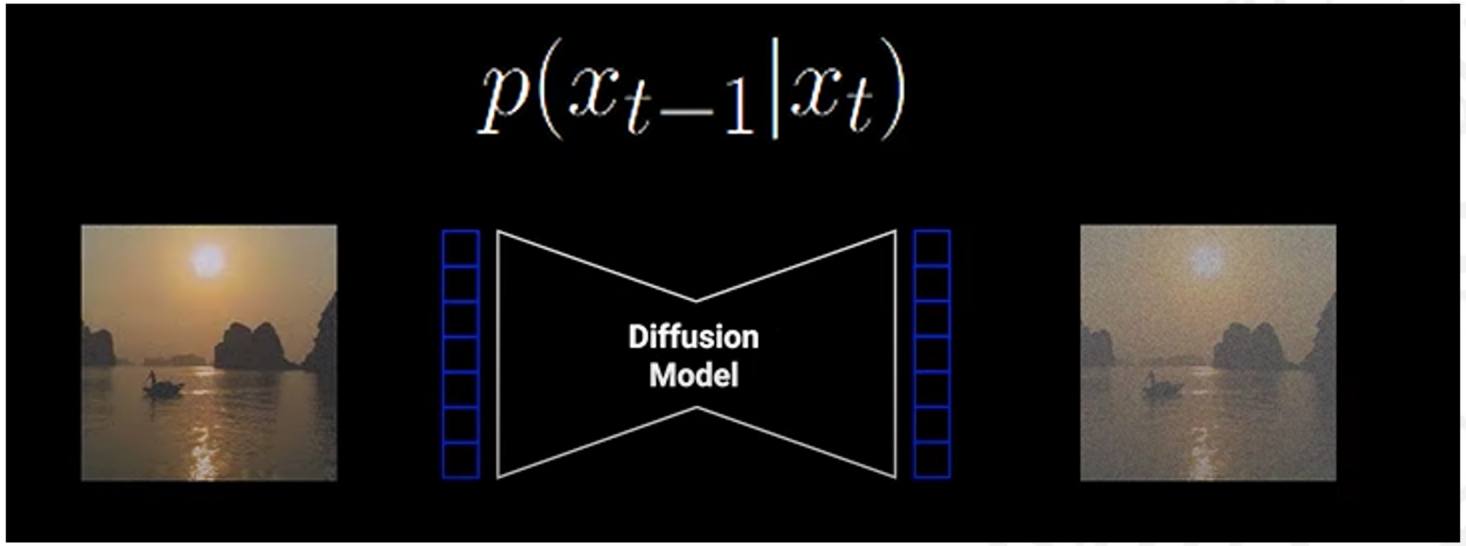


Timestep t - 1 corresponds to image with less noise and timestep t corresponds to image with more noise.


small t = small noise
big t = big noise

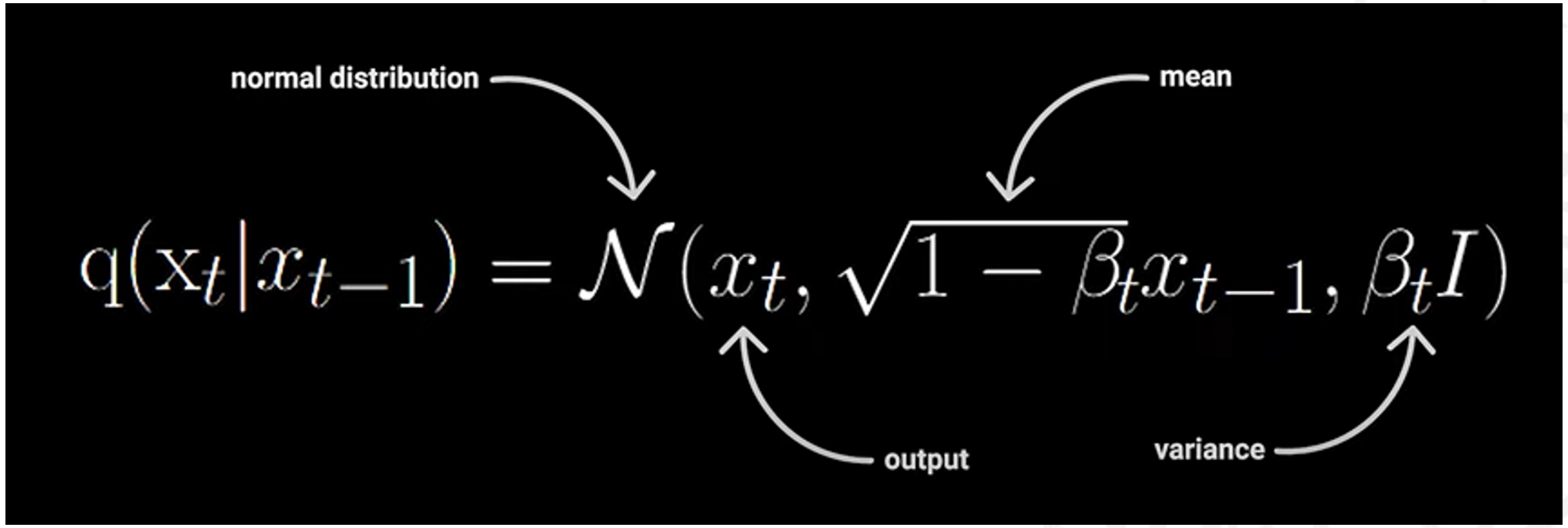
Now let's define some functions

Reverse Process: $p(x_{t-1} \mid x_t)$ defines the reverse process.



Timestep t - 1 corresponds to image with less noise and timestep t corresponds to image with more noise.

Let’s dive into the forward process

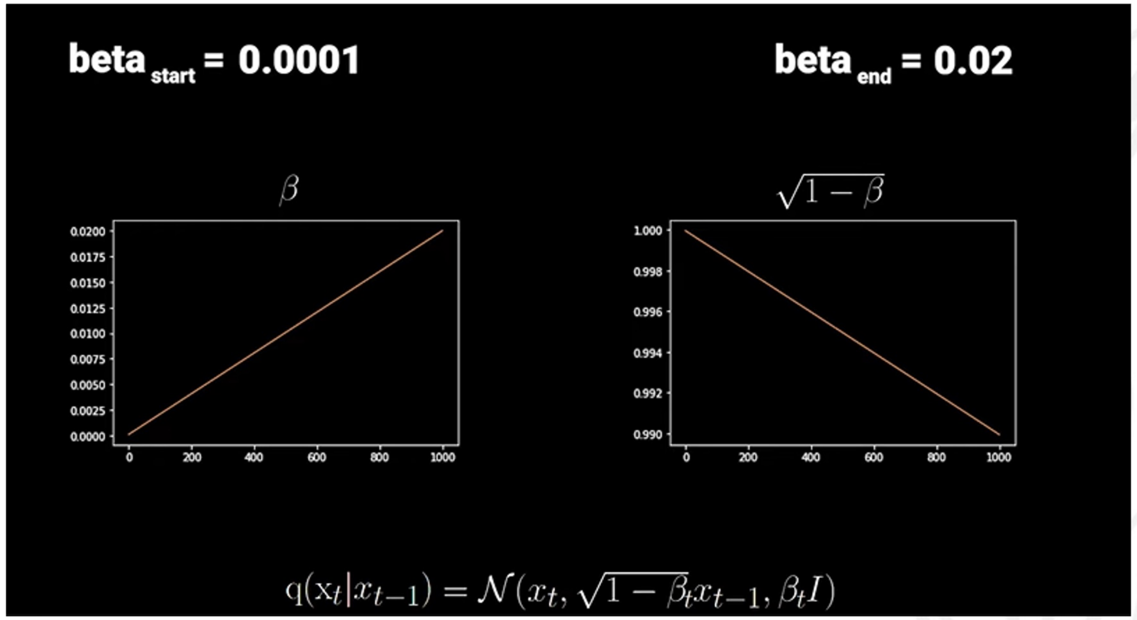


$$q(x_t|x_{t-1}) = \mathcal{N}(x_t, \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

Beta refers to the schedule we discussed on the previous week. It varies between the range 0 to 1 and ensures that the data is being scaled so that the variance doesn’t explode.

Let’s look into the Schedule

Linear Schedule: The defined start and end values of this schedule are 0.0001 and 0.02.



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

We can see from the graphs that we more and more scale down the image which acts as the counterpart for increasing the variance over time and keeping the variance in bound without exploding.

Cool, we now understand how to apply the forward step using the formula

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

This is for one time step. But we need to do this iteratively for 1000 timesteps which needs to repeat the formula 1000 times.

But there is an easier way to do it in one single step.

Let's define some Notations for that:

$$\alpha_t = 1 - \beta_t$$

$$\bar{\alpha}_t = \prod_{s=1}^{t} a_s$$

Cumulative Product of Alpha's till 't'

$$e.g.\ t = 8 \qquad \alpha_8 = \alpha_1 \cdot \alpha_2 \cdot \alpha_3 \cdot \alpha_4 \cdot \alpha_5 \cdot \alpha_6 \cdot \alpha_7 \cdot \alpha_8$$

Let's rewrite the forward process equation $q(x_t \mid x_{t-1})$

$$
\begin{aligned}
q(x_t|x_{t-1}) &= \mathcal{N}(x_t, \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \\
&= \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon \\
&= \sqrt{\alpha_t}\, x_{t-1} + \sqrt{1-\alpha_t}\,\epsilon \\
&= \sqrt{\alpha_t\alpha_{t-1}}\, x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\,\epsilon \\
&= \sqrt{\alpha_t\alpha_{t-1}\alpha_{t-2}}\, x_{t-3} + \sqrt{1-\alpha_t\alpha_{t-1}\alpha_{t-2}}\,\epsilon \\
&= \sqrt{\alpha_t\alpha_{t-1}\ldots\alpha_1\alpha_0}\, x_0 + \sqrt{1-\alpha_t\alpha_{t-1}\ldots\alpha_1\alpha_0}\,\epsilon
\end{aligned}
$$

Reparameterization Trick:

$$\mathcal{N}(\mu, \sigma^2) = \mu + \sigma \cdot \epsilon$$

Epsilon follows a Normal Distribution

The final form of forward process can be written as:

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

You can see here that you can go from x_0 to x_t. directly. Now you can understand why we used cumulative alphas. We can simplify the formula.

Now let’s look into the **reverse process**:

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

We would need two neural networks to parameterize the Normal Distribution which we can sample from to get x_{t-1}

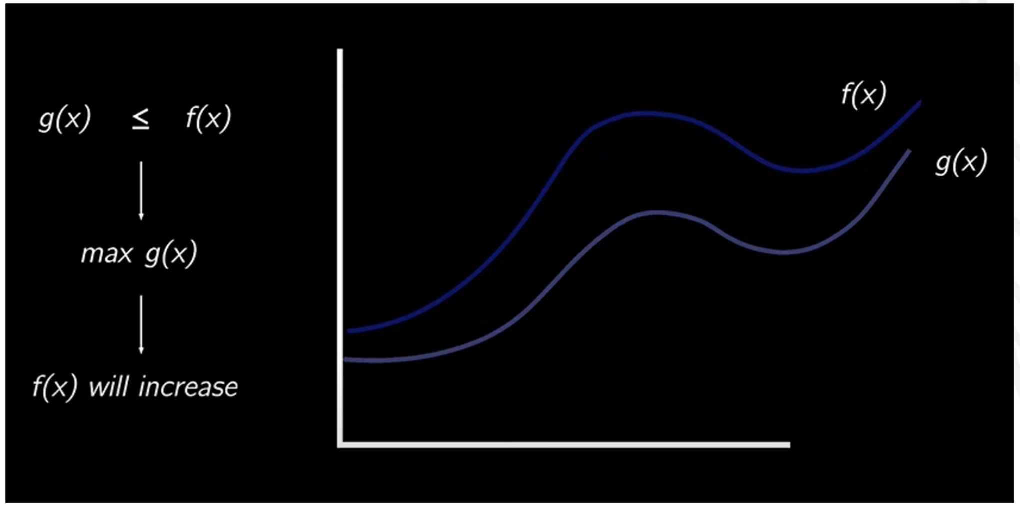
$$p(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \underbrace{\mu_\theta(x_t, t)}_{\text{we only need to predict this}}, \overset{\overset{\text{we don't need a neural network for that}}{\downarrow}}{\overset{\text{fixed}}{\overbrace{\Sigma_\theta(x_t, t)}}})$$

In order to have an easy time, we will start by looking at the loss function. It is a simple negative log likelihood.

$$-\,log(p_\theta(x_0))$$

We know that p(x_0) isn't nicely computable as it depends on all the time steps coming before x_0 upto x_T
Keeping track of all these is not possible

One trick we can use to solve this is Variational Lower Bound



So we can subtract KL Divergence which is a measure how similar two distributions are which is always +ve.

$$-\,log(p_\theta(x_0)) \leq -log(p_\theta(x_0)) + D_{KL}(q(x_{1:T}|x_0)||p_\theta(x_{1:T}|x_0))$$

$$D_{KL}(p||q) = \int_x p(x) log \frac{p(x)}{q(x)} dx$$

We are adding because we need to minimize the loss function. But this is still not computable as well.

$$\overbrace{-\,log(p_\theta(x_0)) \leq -log(p_\theta(x_0))}^{=} + D_{KL}(q(x_{1:T}|x_0)||p_\theta(x_{1:T}|x_0))$$

$$\begin{gathered} D_{KL}(q(x_{1:T}|x_0)||p_\theta(x_{1:T}|x_0)) \\ \downarrow \\ log(\frac{q(x_{1:T}|x_0)}{p_\theta(x_{1:T}|x_0)} \end{gathered}$$

Apply Bayesian rule to the lower term.

$$\begin{tikzcd}
 & & log(\frac{q(x_{1:T}|x_0)}{p_\theta(x_{1:T}|x_0)}) \arrow[d] \arrow[lldd, dash, rounded corners, to path={-| (\tikztotarget)}] \\
\frac{p_\theta(x_{0:T})}{p_\theta(x_0)} \arrow[d, dash] & \frac{p_\theta(x_0, x_{1:T})}{p_\theta(x_0)} \arrow[l, mapsto] & \frac{p_\theta(x_0|x_{1:T})p_\theta(x_{1:T})}{p_\theta(x_0)} \arrow[l, mapsto] \\
log(\frac{q(x_{1:T}|x_0)}{\frac{p_\theta(x_{0:T})}{p_\theta(x_0)}}) \arrow[r] & log(\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}) + log(p_\theta(x_0)) &
\end{tikzcd}$$

We use joint probability in some places

$$-\,log(p_\theta(x_0)) \leq \overbrace{-log(p_\theta(x_0)) + log(\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}) + log(p_\theta(x_0))}$$

We get rid of that annoying quantity.

Great, Now this is our variational lower bound which we can minimize.

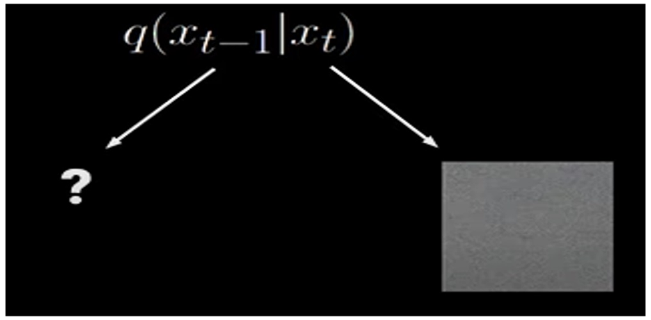
$$- \, log(p_\theta(x_0)) \leq log(\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})})$$

The term at the top of second term is the forward process

$$\begin{aligned}
log(\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}) &= \log(\frac{\prod_{t=1}^{T} q(x_t|x_{t-1})}{p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)}) \\
&= \text{-}\log(\mathrm{p}(\mathrm{x}_T)) + log(\frac{\prod_{t=1}^{T} q(x_t|x_{t-1})}{\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)}) \\
&= \text{-}\log(\mathrm{p}(\mathrm{x}_T)) + \sum_{t=1}^{T} log(\frac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)}) \qquad t=1 \\
&= \text{-}\log(\mathrm{p}(\mathrm{x}_T)) + \sum_{t=2}^{T} log(\frac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)}) + log(\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)})
\end{aligned}$$

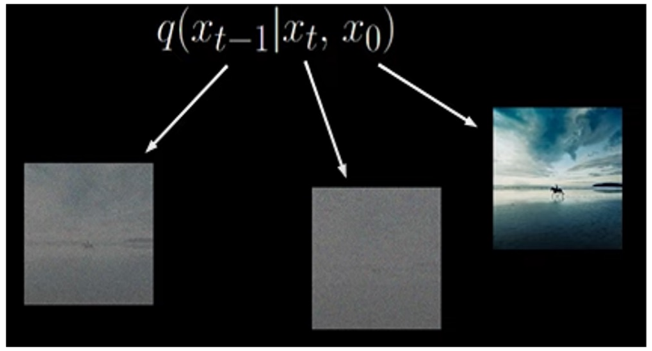
The cumulative product is taken out which convert to the summation.

Later we will divide the summation to two parts. One at time step 1 and rest in another term. We will see why we did that.



$$q(x_t|x_{t-1}) = \frac{q(x_{t-1}|x_t)\,q(x_t)}{q(x_{t-1})}$$

We will apply Bayes Rule to the top term of second equation. All these terms has very high variance since we don't know where we started from.



$$q(x_t|x_{t-1}) = \frac{q(x_{t-1}|x_t)\,q(x_t)}{q(x_{t-1})}$$

$$\Rightarrow \frac{q(x_{t-1}|x_t, x_0)\,q(x_t|x_0)}{q(x_{t-1}|x_0)}$$

This is possible when it is extra conditioned with x_0
This formula also has a closed form solution which we will talk later.

If we didn’t remove the last term, it would have been mess.

Let’s continue.

$$= -\log(p(x_T)) + \sum_{t=2}^{T} log\left(\frac{q(x_{t-1}|x_t,\ x_0)}{p_\theta(x_{t-1}|x_t)}\right) + \boxed{\sum_{t=2}^{T} log\left(\frac{q(x_t|x_0)}{q(x_{t-1}|x_0)}\right)} + log\left(\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right)$$

T=4

$$\sum_{t=2}^{4} log\left(\frac{q(x_t|x_0)}{q(x_{t-1}|x_0)}\right) = log\left(\prod_{t=2}^{4} \frac{q(x_t|x_0)}{q(x_{t-1}|x_0)}\right) = log\left(\frac{q(x_2|x_0)\, q(x_3|x_0)\, q(x_4|x_0)}{q(x_1|x_0)\, q(x_2|x_0)\, q(x_3|x_0)}\right)$$

$$= -\log(p(x_T)) + \sum_{t=2}^{T} log\left(\frac{q(x_{t-1}|x_t,x_0)\, q(x_t|x_0)}{p_\theta(x_{t-1}|x_t)\ \ q(x_{t-1}|x_0)}\right) + log\left(\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right)$$

$$\boxed{q(x_1|x_0)} = \frac{q(x_0|x_1)\, q(x_1)}{q(x_0)}$$

$$\Rightarrow \frac{q(x_0|x_1,x_0)\, \boxed{q(x_1|x_0)}}{q(x_0|x_0)}$$

$$= -\log(p(x_T)) + \sum_{t=2}^{T} log\left(\frac{q(x_{t-1}|x_t,\ x_0)}{p_\theta(x_{t-1}|x_t)}\right) + \quad log\left(\frac{q(x_T|x_0)}{q(x_1|x_0)}\right) \quad + log\left(\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right)$$

$$= \boxed{-\log(p(x_T))} + \sum_{t=2}^{T} log\left(\frac{q(x_{t-1}|x_t,\ x_0)}{p_\theta(x_{t-1}|x_t)}\right) + \boxed{log\left(\frac{q(x_T|x_0)}{q(x_1|x_0)}\right) + log\left(\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right)}$$

$$\log\left(\frac{q(x_T|x_0)}{p(x_T)}\right) \qquad \boxed{\log(q(x_T|x_0))} - \log(p_\theta(x_0|x_1))$$

$$= \log\left(\frac{q(x_T|x_0)}{p(x_T)}\right) + \sum_{t=2}^{T} log\left(\frac{q(x_{t-1}|x_t,\ x_0)}{p_\theta(x_{t-1}|x_t)}\right) - \log(p_\theta(x_0|x_1))$$

$$= D_{KL}(q(x_T|x_0)\,||\,p(x_T)) + \sum_{t=2}^{T} D_{KL}(q(x_{t-1}|x_t,\ x_0)\,||\,p_\theta(x_{t-1}|x_t)) - \log(p_\theta(x_0|x_1)$$

$$\boxed{D_{KL}(q(x_T|x_0)\,||\,p(x_T))} + \sum_{t=2}^{T} D_{KL}(q(x_{t-1}|x_t,\ x_0)\,||\,p_\theta(x_{t-1}|x_t)) - \log(p_\theta(x_0|x_1))$$

We can ignore the first term as it indicates forward process and q has no learnable parameters.The KL DIvergence will be small because forward process converges to Normal Distribution and difference is small with a random sampled noise.We can use the third term for scaling but we can ignore that for now.

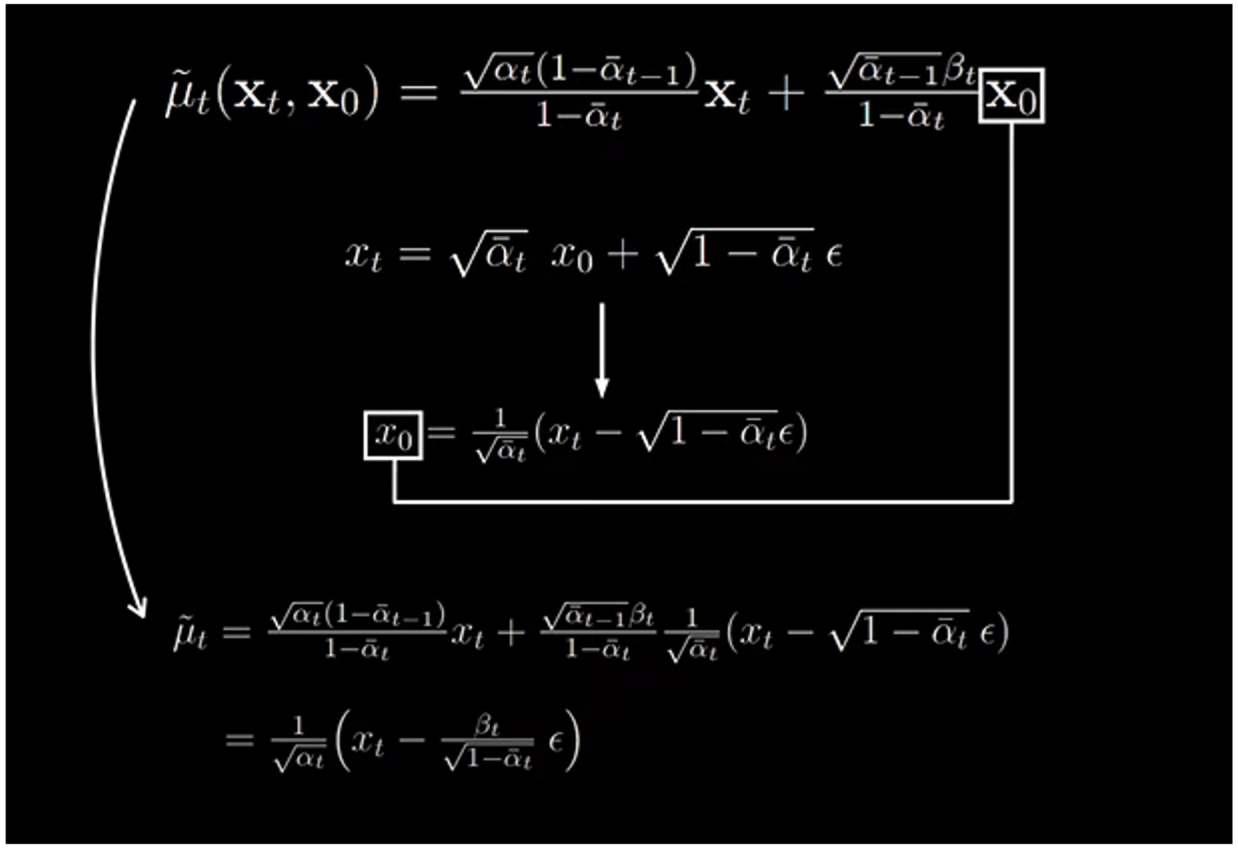
$$\sum_{t=2}^{T} D_{KL}(\underbrace{q(x_{t-1}|x_t,\ x_0)}_{= \mathcal{N}(x_{t-1};\tilde{\mu}_t(x_t,x_0),\tilde{\beta}_t I)}\,||\,\underbrace{p_\theta(x_{t-1}|x_t)}_{= \mathcal{N}(x_{t-1};\mu_\theta(x_t,t),\beta I)}) - \log(p_\theta(x_0|x_1))$$

This is the closed form solution I was referring to. The derivation is similar as explained in the forward process and the final form would look like

$$\tilde{\mu}_t(\mathbf{x}_t,\mathbf{x}_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 \qquad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\cdot\beta_t$$

Let’s focus on mean as we know variance is constant.

From forward process, we know the value of x_t



$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boxed{\mathbf{x}_0}$$

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

$$\downarrow$$

$$\boxed{x_0} = \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1-\bar{\alpha}_t}\epsilon)$$

$$\tilde{\mu}_t = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1-\bar{\alpha}_t}\;\epsilon)$$

$$= \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\;\epsilon\right)$$

We can see at the end that we are subtracting random scaled noise from x_t

$$\sum_{t=2}^{T} D_{KL}(q(x_{t-1}|x_t,\ x_0)\,||\,p_\theta(x_{t-1}|x_t)) - \log(p_\theta(x_0|x_1)$$

$$= \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I) \qquad\qquad = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \beta I)$$

$$\frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon\right)$$

Now this is what our Neural Network needs to predict and the authors decided to use simple MSE

$$L_t = \frac{1}{2\sigma_t^2} || \, \tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t) ||^2$$

Substituting the mean we get

$$L_t = \frac{1}{2\sigma_t^2} || \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \, \epsilon \right) - \mu_\theta(x_t, t) ||^2$$

x_t is the input to the model. Instead of predicting this entire first term, why don't we just predict the noise?

$$L_t = \frac{1}{2\sigma_t^2} || \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \, \epsilon \right) - \mu_\theta(x_t, t) ||^2$$

$$\Big\downarrow$$

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

Once we substitute mean and simplify we get,

$$\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\hat{\alpha}_t)}||\epsilon - \epsilon_\theta(x_t, t)||^2$$

The authors found that we can ignore the first scaling term for further simplification as it doesn't effect the value much.

So Finally, we get

$$||\epsilon - \epsilon_\theta(x_t, t)||^2$$

Finally, we can conclude that predicting the noise of the image is all what we need from the U-Net.

From the finding of the noise we can get back our image

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

$$\mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

$$\mathcal{N}\left(x_{t-1};\ \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(x_t, t)\right), \beta_t\right)$$

Applying

Reparameterization Trick:

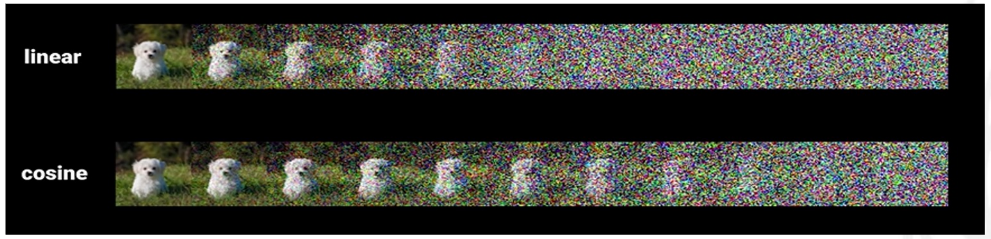
$$\mathcal{N}(\mu, \sigma^2) = \mu + \sigma \cdot \epsilon$$

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(x_t, t)\right) + \sqrt{\beta_t}\,\epsilon$$

Epsilon is noise sampled from random Normal

Some Improvement we can think of

- We can improve the schedule. We saw that Linear Schedule de-structures the image at early timesteps and some steps are redundant at last. We can use Cosine Schedule for improving the adding of the noise segment.



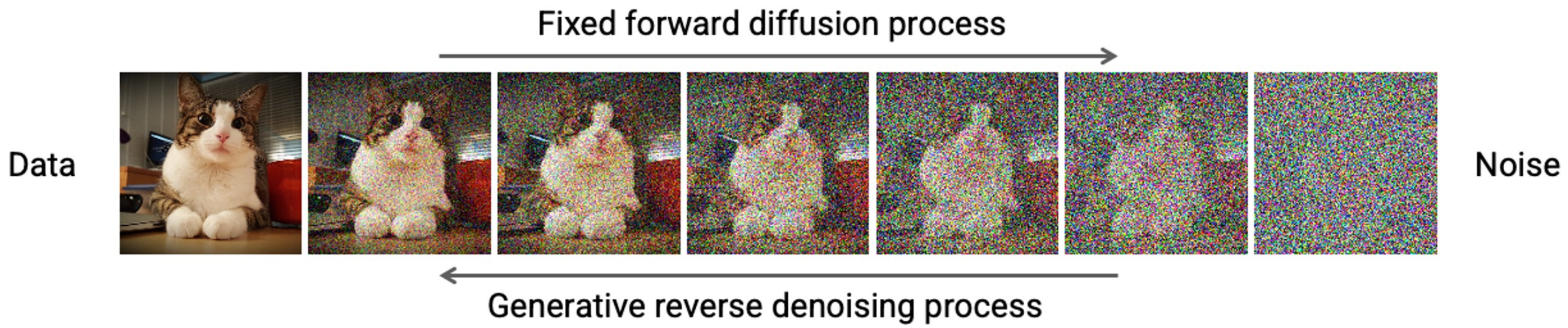
- We saw that we fixed the variance using the schedule. We can learn the variance as well using an another Neural Network.
- We can explore on the Architectures following the similar principles. Why can't we use the latent space technique of VAE's here?

Let's see the third step in detail.

LATENT DIFFUSION MODEL

Paper Link:
High-Resolution Image Synthesis with Latent Diffusion Models

Denosing Diffusion Model


Fixed forward diffusion process
Data
Noise
Generative reverse denoising process

Sequential sampling process

Iterative refinement

Operate directly in pixel space

LDM helps with Limited Computationally Resource

Latent Space

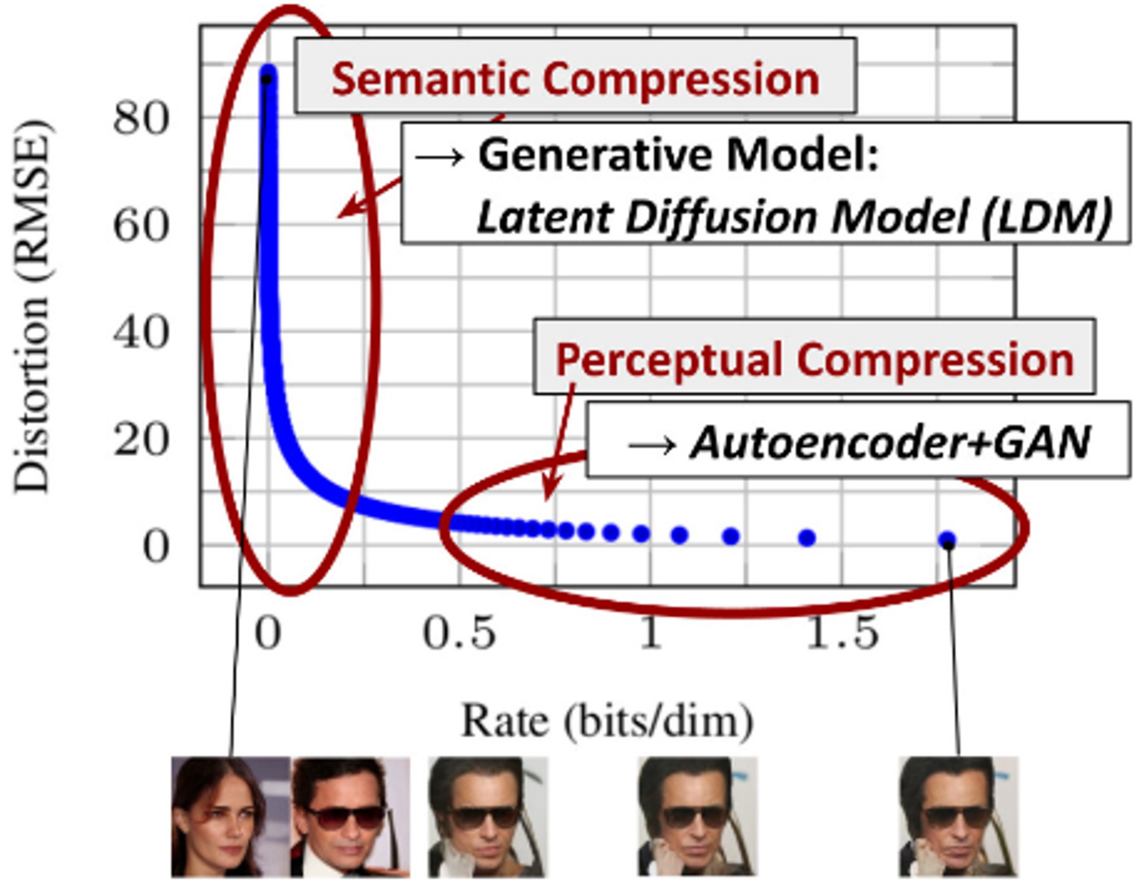
Complexity reduction (focus on the important ,semantic bits of data)

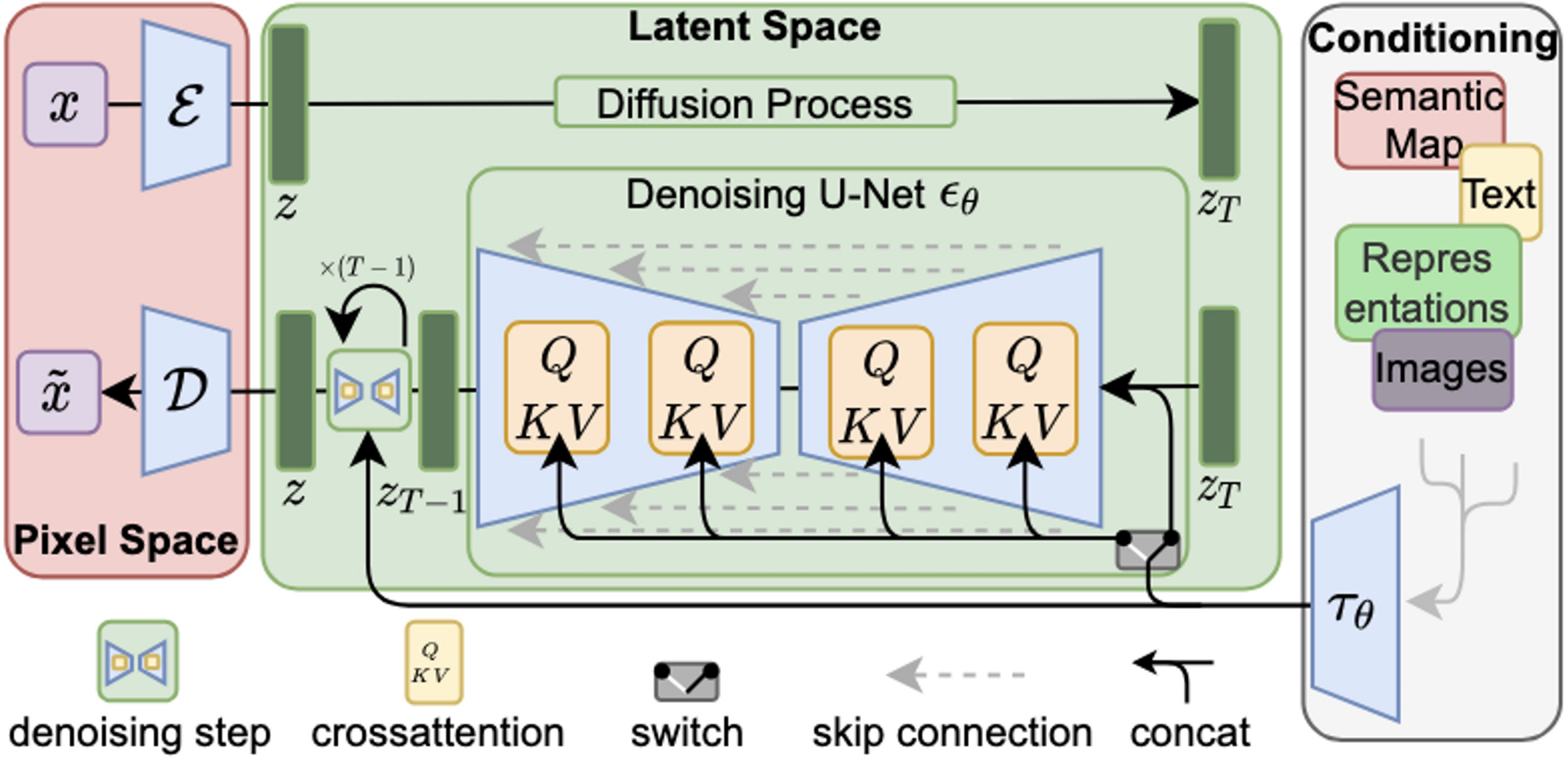
Detail preservation (train in low-dimensional, computationally much more efficient space)

Cross-attention layer

Generator for general conditioning inputs in a convolutional manner

Perceptual compression and Semantic compression



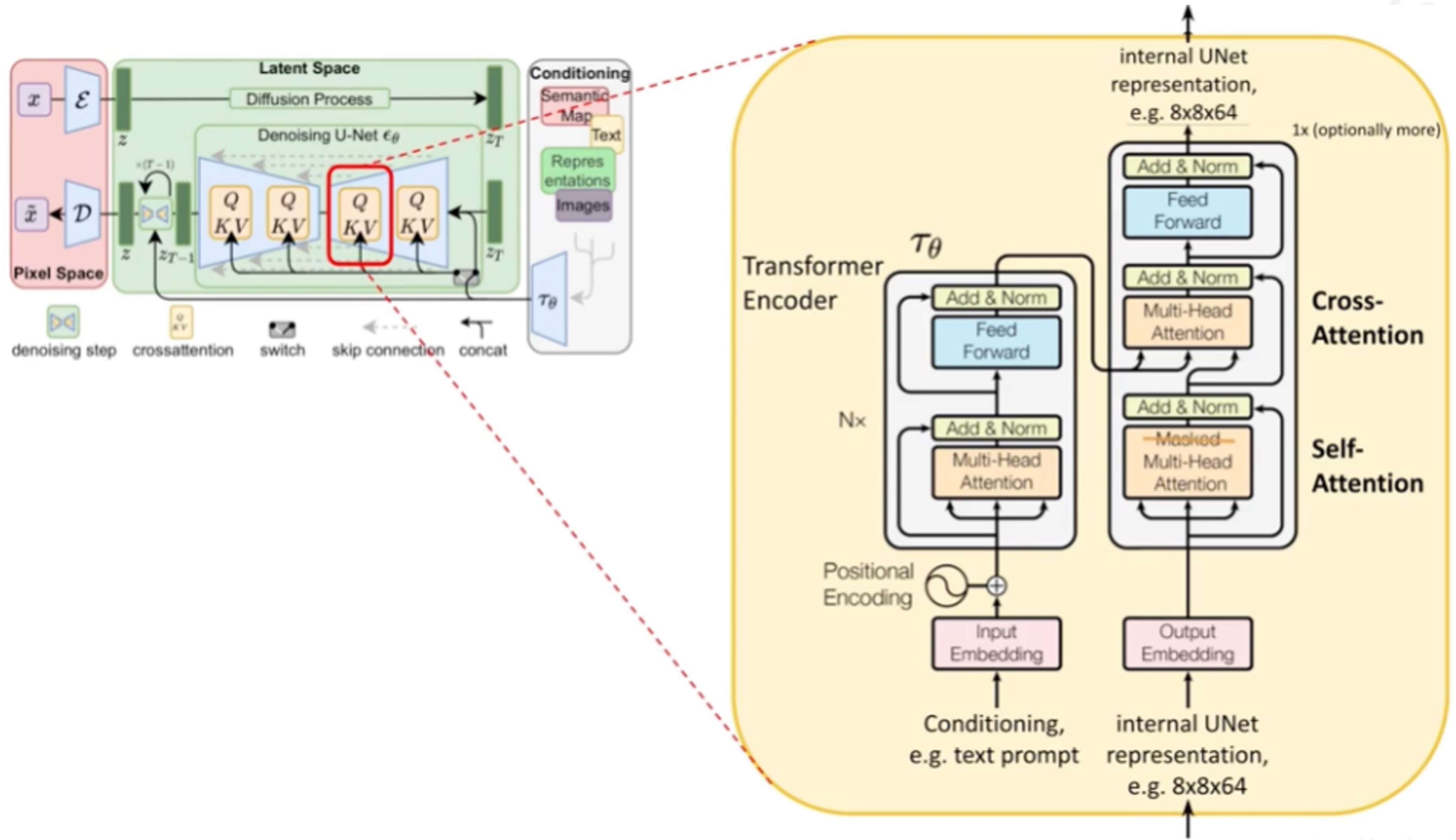

Latent Space
Pixel Space
Conditioning
Diffusion Process
Denoising U-Net ϵ_θ
Semantic Map
Text
Repres entations
Images
x
$\mathcal{E}$
$\tilde{x}$
$\mathcal{D}$
z
z_T
z_{T-1}
$\times(T-1)$
Q
KV
τ_θ
denoising step
crossattention
switch
skip connection
concat

Train Autoencoder in Adversarial Manner

$$L_{\text{Autoencoder}} = \min_{\mathcal{E},\mathcal{D}} \max_{\psi} \left(L_{rec}(x, \mathcal{D}(\mathcal{E}(x))) - L_{adv}(\mathcal{D}(\mathcal{E}(x))) + \log D_{\psi}(x) + L_{reg}(x; \mathcal{E}, \mathcal{D}) \right)$$

$L_{rec}(x, \mathcal{D}(\mathcal{E}(x)))$	Reconstruction Loss
$L_{adv}(\mathcal{D}(\mathcal{E}(x)))$	Adversarial Loss
$\log D_{\psi}(x)$	Log-likelihood of the discriminator Correctly classifying the real data
$L_{reg}(x; \mathcal{E}, \mathcal{D}))$	Regularization term

Cross Attention Layer



Conditioning mechanism in LDM

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

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

Here $\varphi_i(z_t)$ denotes a (flattened) intermediate representation of the UNet implementing

Diffusion models and Latent diffusion model

Diffusion Model

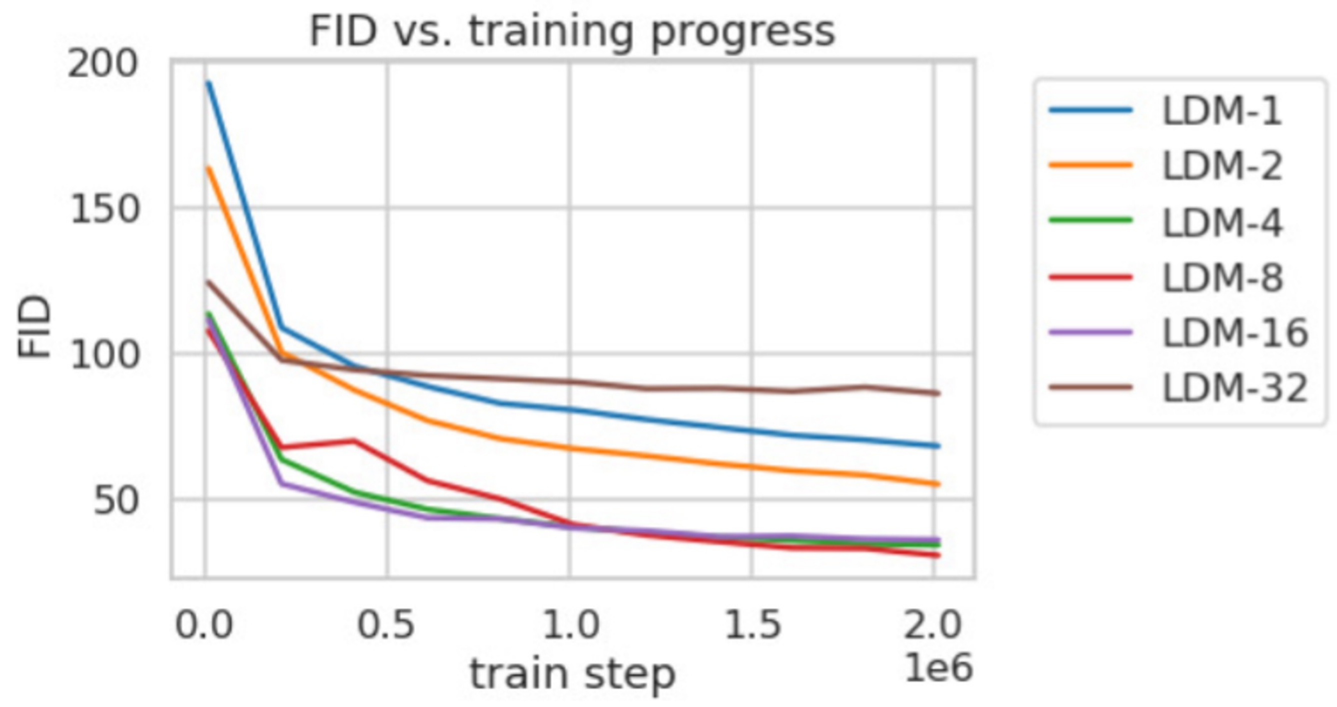
$$L_{DM} = \mathbb{E}_{x,\epsilon\sim\mathcal{N}(0,1),t}\Big[\|\epsilon - \epsilon_\theta(x_t,t)\|_2^2\Big] , \qquad (1)$$

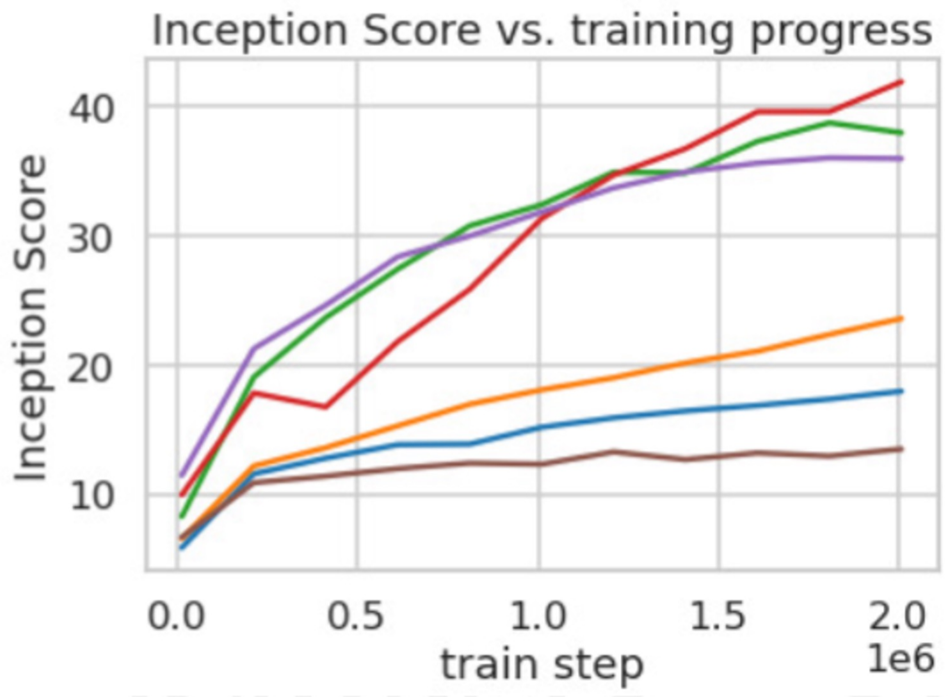
Latent Diffusion Model

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

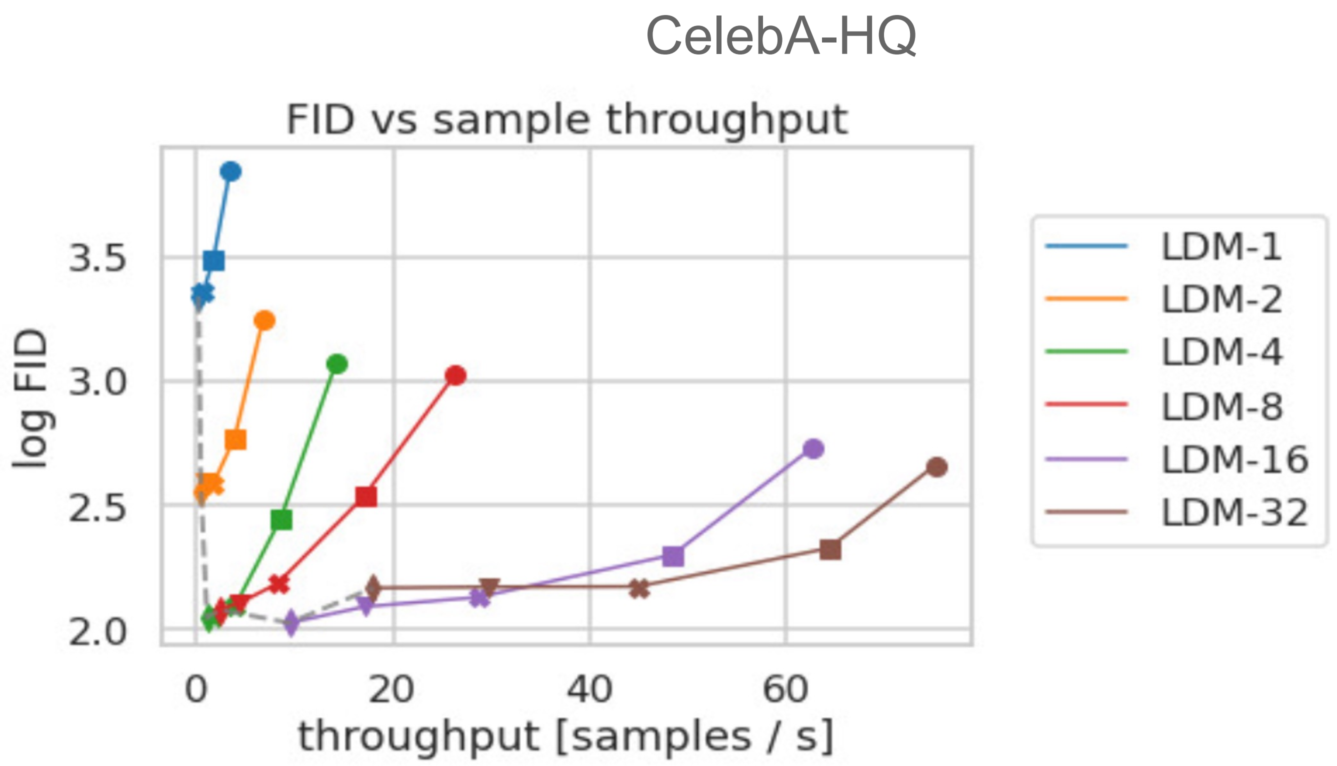
$$L_{LDM} := \mathbb{E}_{\mathcal{E}(x),y,\epsilon\sim\mathcal{N}(0,1),t}\Big[\|\epsilon - \epsilon_\theta(z_t,t,\tau_\theta(y))\|_2^2\Big] , \quad (3)$$

Latent diffusion model Downsampling Experiments





Latent diffusion model Downsampling Experiments



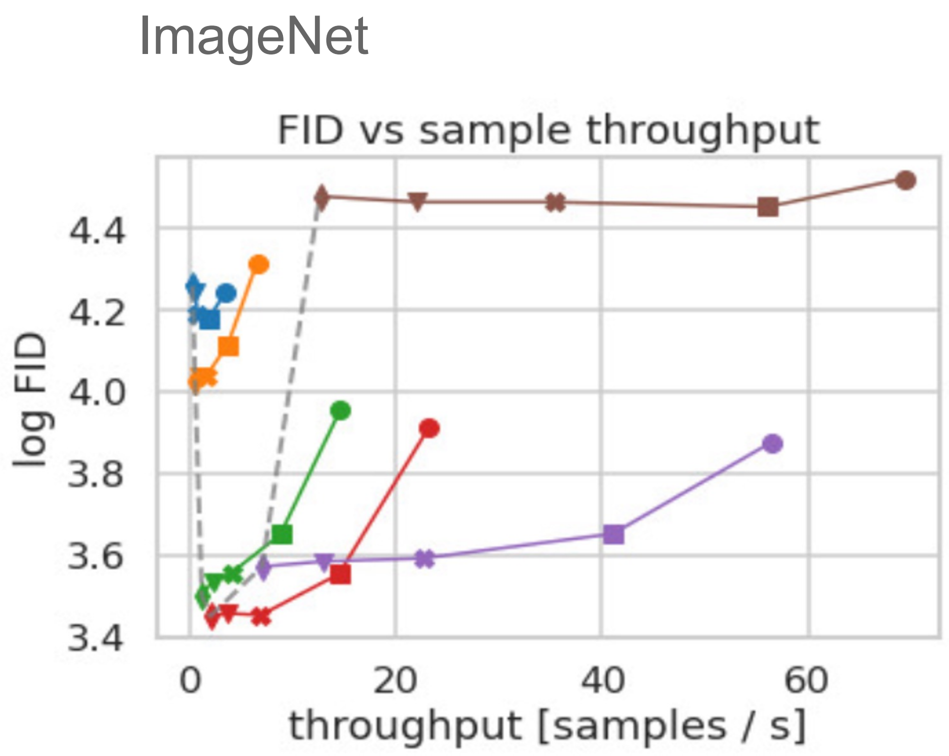


Figure 7. Comparing LDMs with varying compression on the CelebA-HQ (left) and ImageNet (right) datasets. Different markers indicate {10, 20, 50, 100, 200} sampling steps using DDIM, from right to left along each line. The dashed line shows the FID scores for 200 steps, indicating the strong performance of LDM- {4-8}. FID scores assessed on 5000 samples. All models were trained for 500k (CelebA) / 2M (ImageNet) steps on an A100

Qualitative Results On Each Dataset

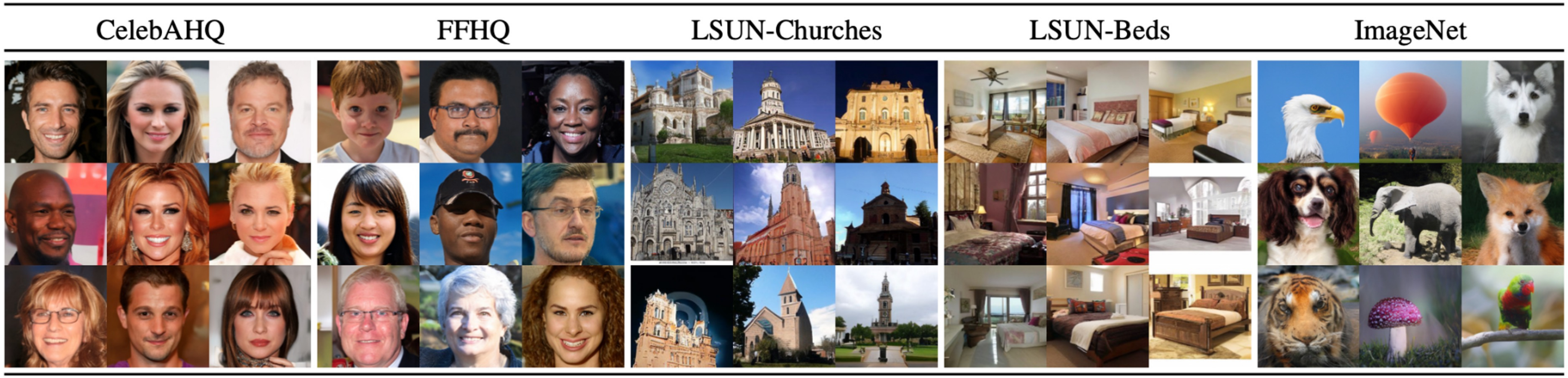
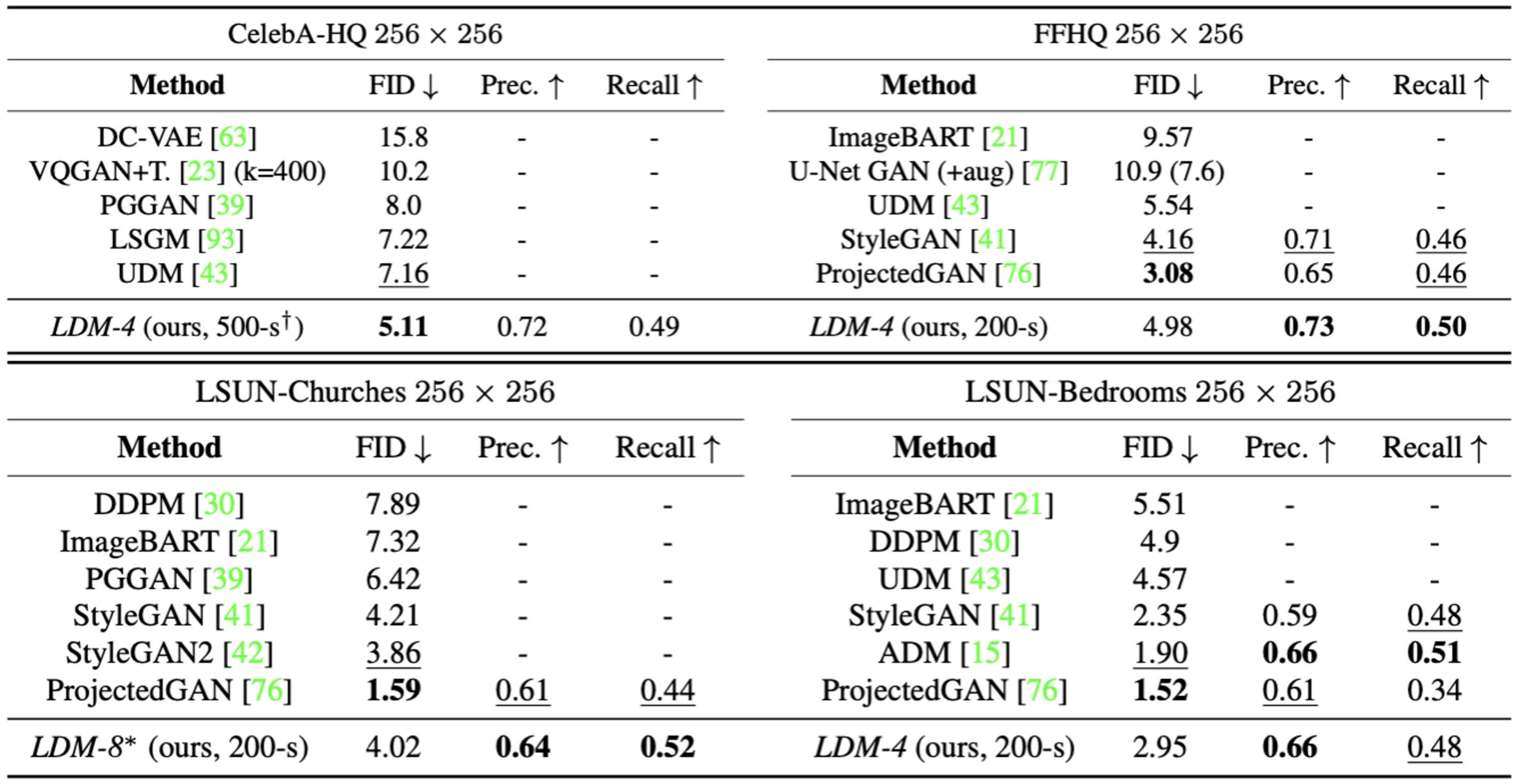


Figure 4. Samples from *LDMs* trained on CelebAHQ [39], FFHQ [41], LSUN-Churches [102], LSUN-Bedrooms [102] and class-conditional ImageNet [12], each with a resolution of 256 × 256. Best viewed when zoomed in. For more samples *cf*. the supplement.

Unconditional Image generation with LDM



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

Table 1. Evaluation metrics for unconditional image synthesis. CelebA-HQ results reproduced from [43, 63, 100], FFHQ from [42, 43]. [†]: N-s refers to N sampling steps with the DDIM [84] sampler. *: trained in *KL*-regularized latent space. Additional results can be found in the supplementary.

Text-to-Image Conditioning

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!” ’

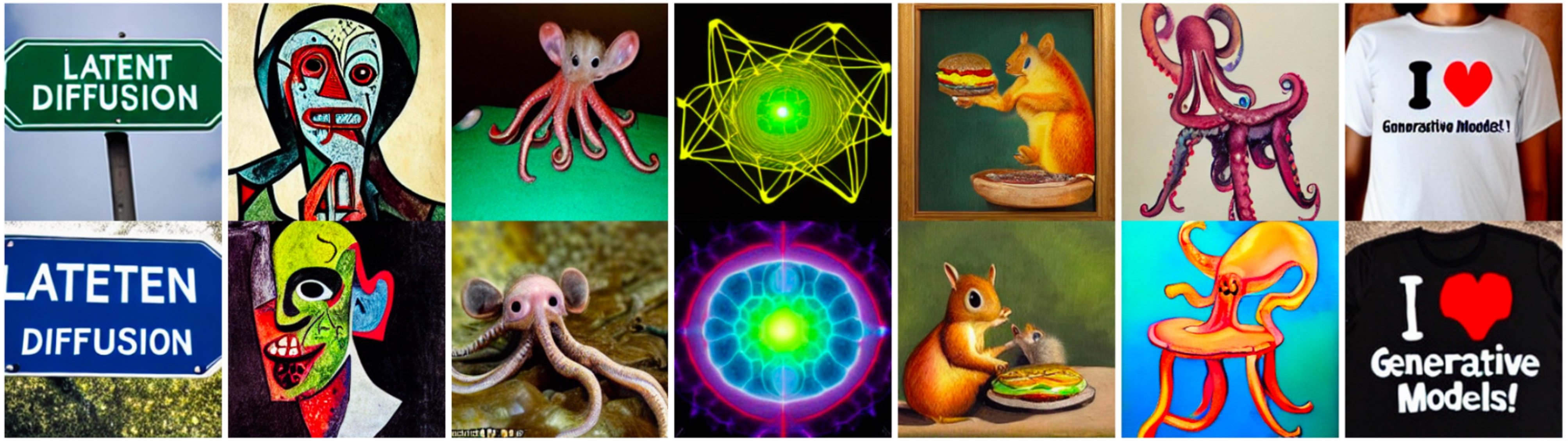
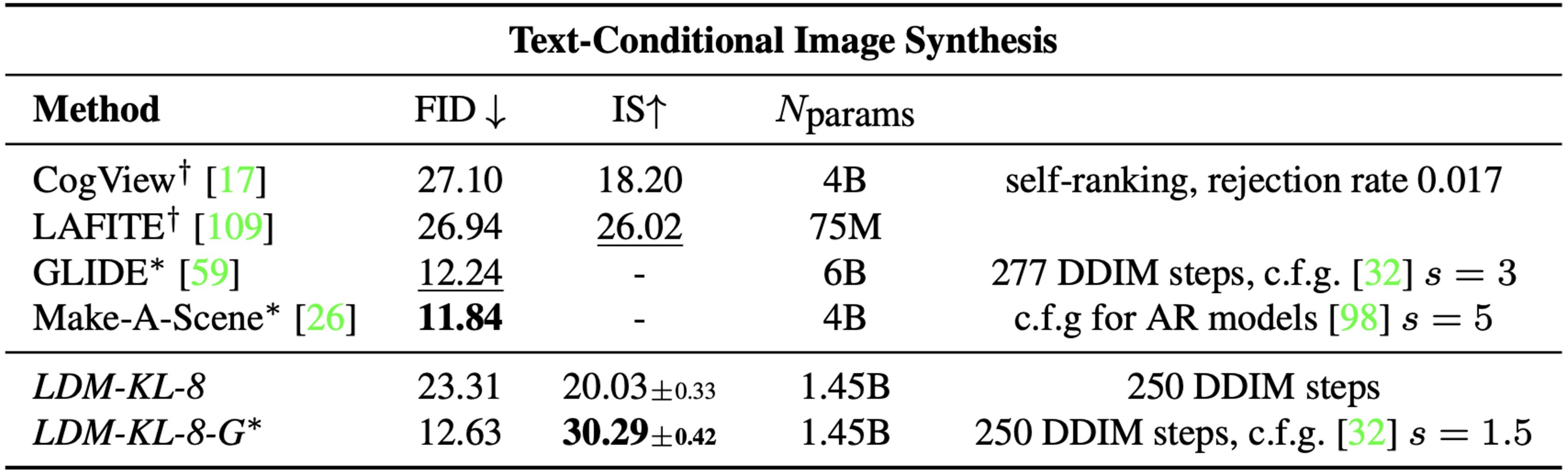


Figure 5. Samples for user-defined text prompts from our model for text-to-image synthesis, *LDM-8 (KL)*, which was trained on the LAION [78] database. Samples generated with 200 DDIM steps and $\eta = 1.0$. We use unconditional guidance [32] with $s = 10.0$.

Image generation with Latent diffusion model



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	26.02	75M	
GLIDE* [59]	12.24	-	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] $s = 1.5$

Table 2. Evaluation of text-conditional image synthesis on the 256 × 256-sized MS-COCO [51] dataset: with 250 DDIM [84] steps our model is on par with the most recent diffusion [59] and autoregressive [26] methods despite using significantly less parameters. †/*:Numbers from [109]/ [26]

Layout to Image Synthesis

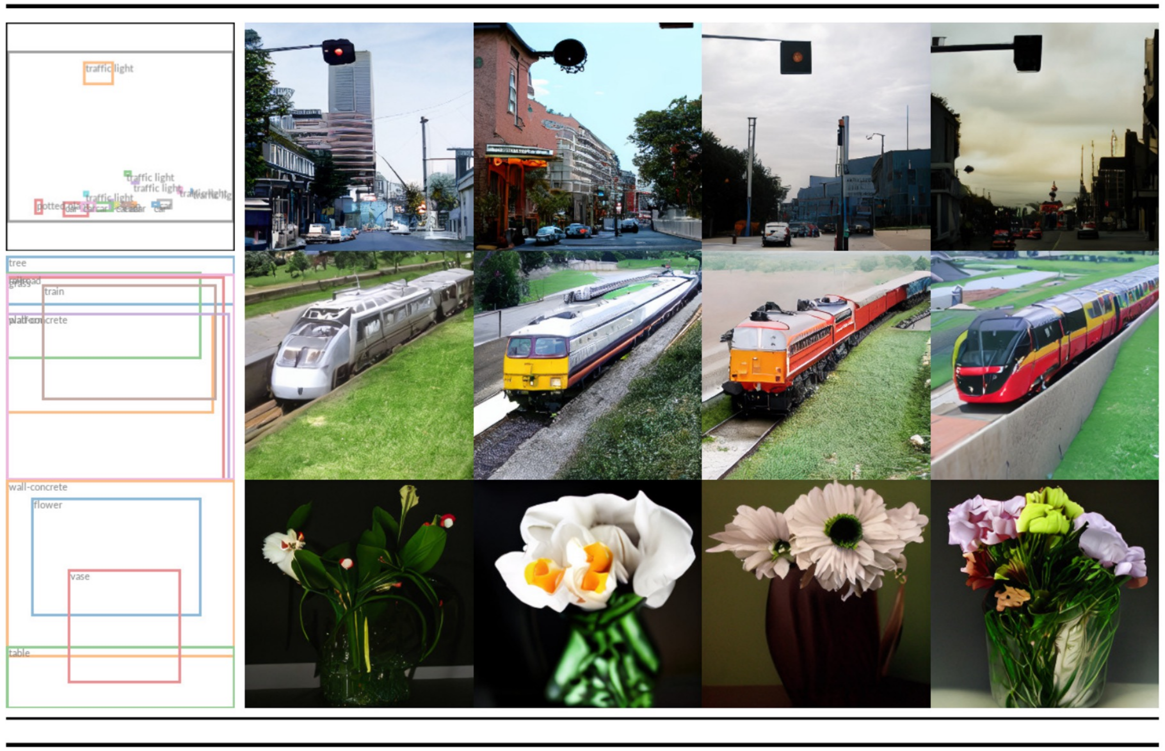


Figure 8. Layout-to-image synthesis with an LDM on COCO [4], see Sec. 4.3.1. Quantitative evaluation in the supplement D.3.

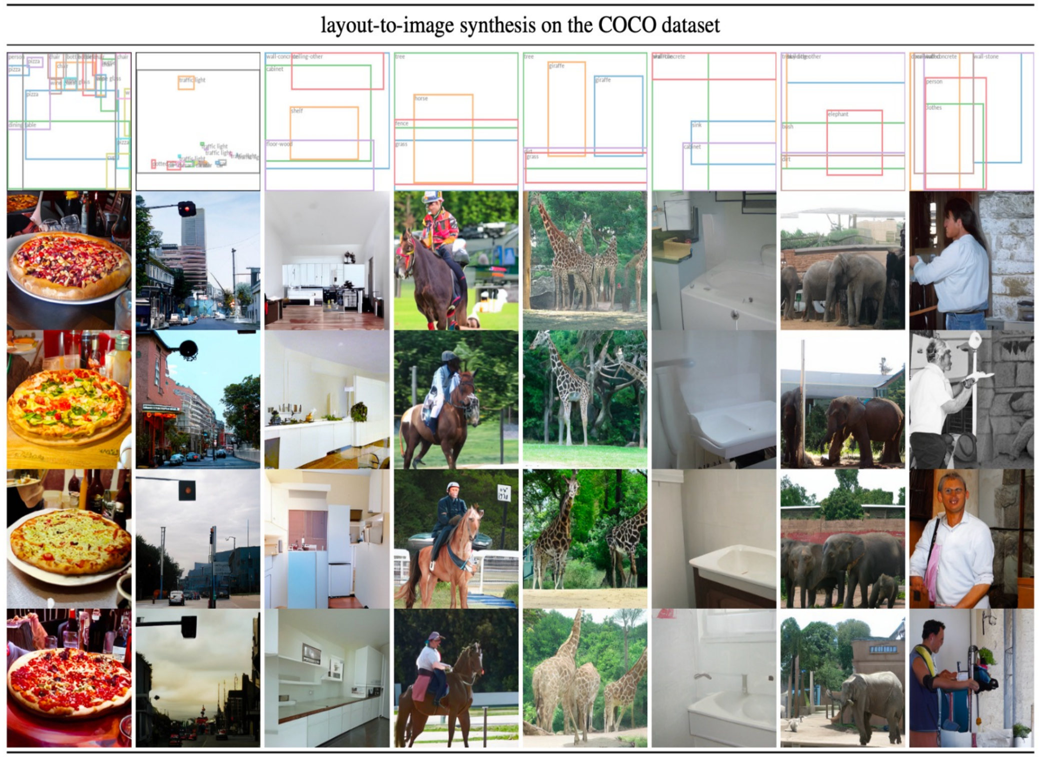


Figure 9. More samples from our best model for layout-to-image synthesis, LDM-4, which was trained on the Open Images dataset and fine tuned on the COCO dataset. Samples generated with 100 DDIM steps and η = 0. Layouts are from the COCO validation set.

Convolutional Sampling Beyond 256 ^{2}

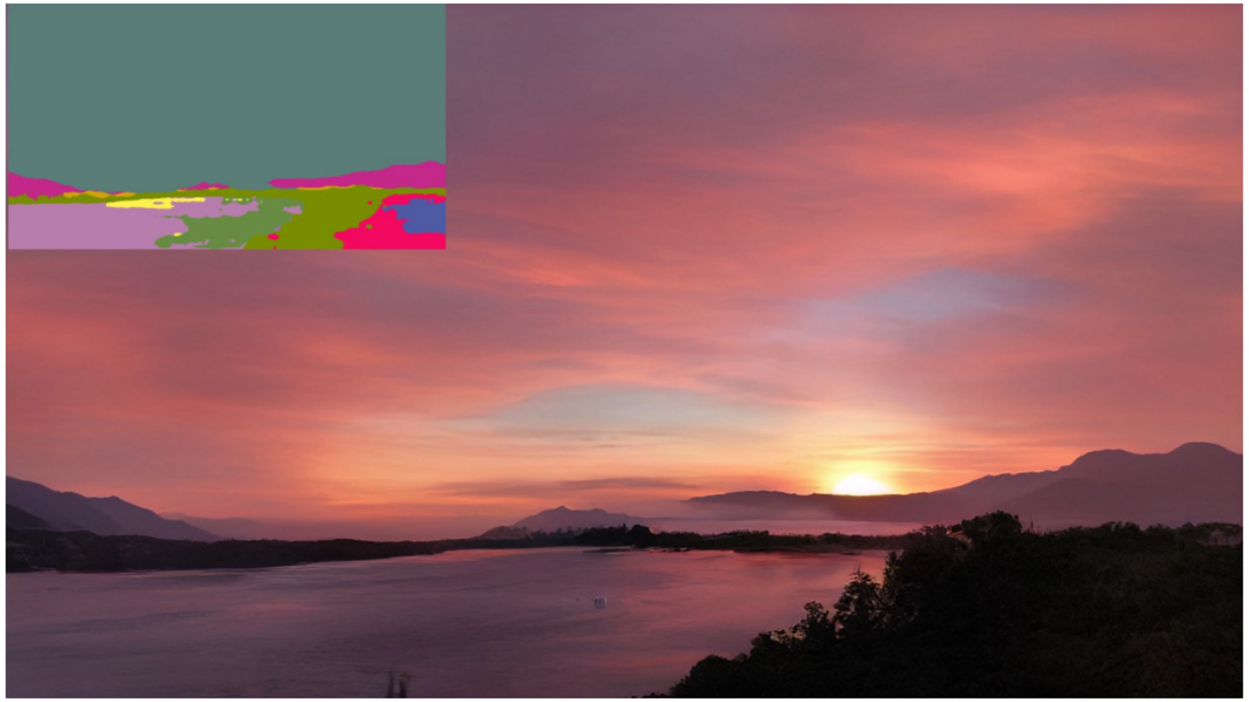


Figure 9. A *LDM* trained on 256^2 resolution can generalize to larger resolution (here: 512×1024) for spatially conditioned tasks such as semantic synthesis of landscape images. See Sec. 4.3.2.

Super-Resolution with Latent Diffusion

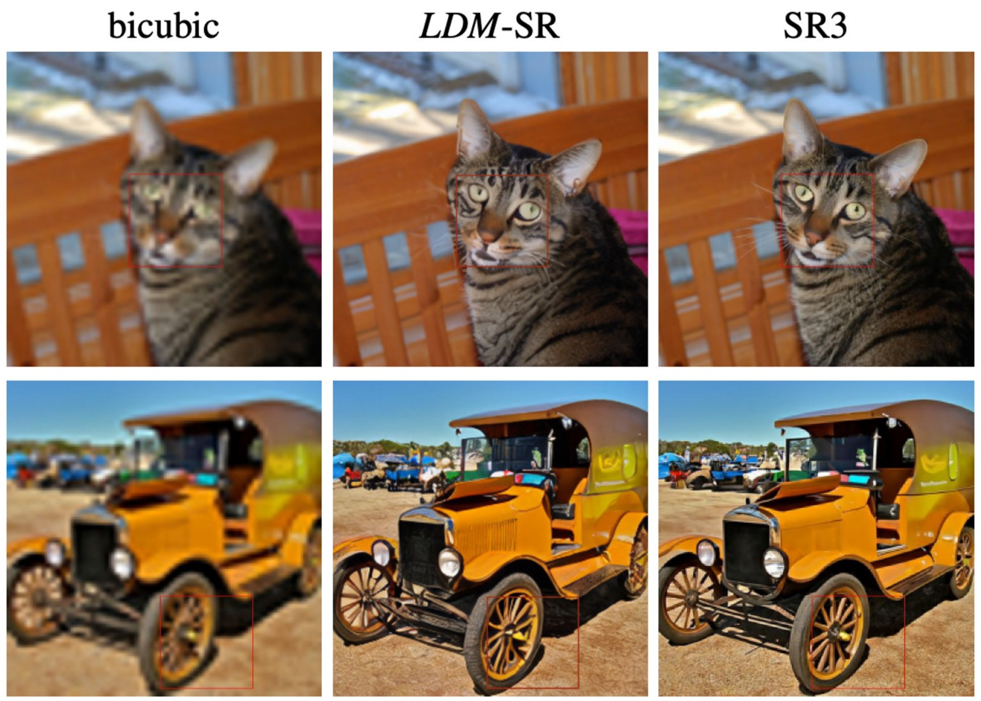
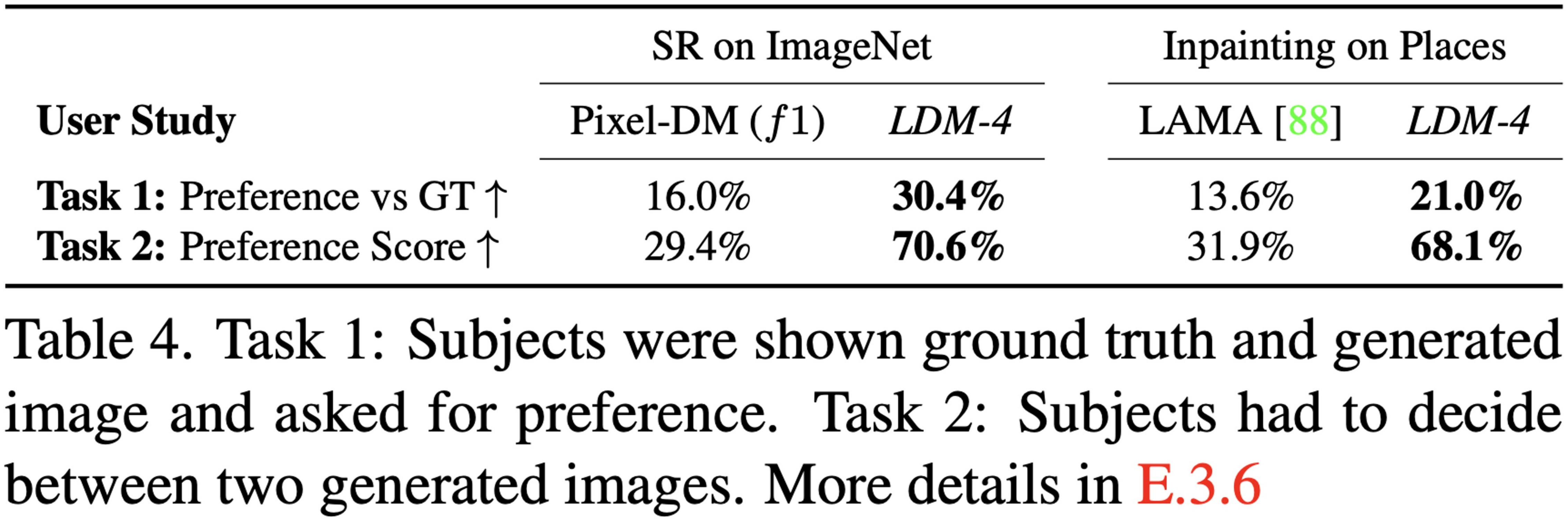


Figure 10. ImageNet 64→256 super-resolution on ImageNet-Val. *LDM-SR* has advantages at rendering realistic textures but SR3 can synthesize more coherent fine structures. See appendix for additional samples and cropouts. SR3 results from [72].

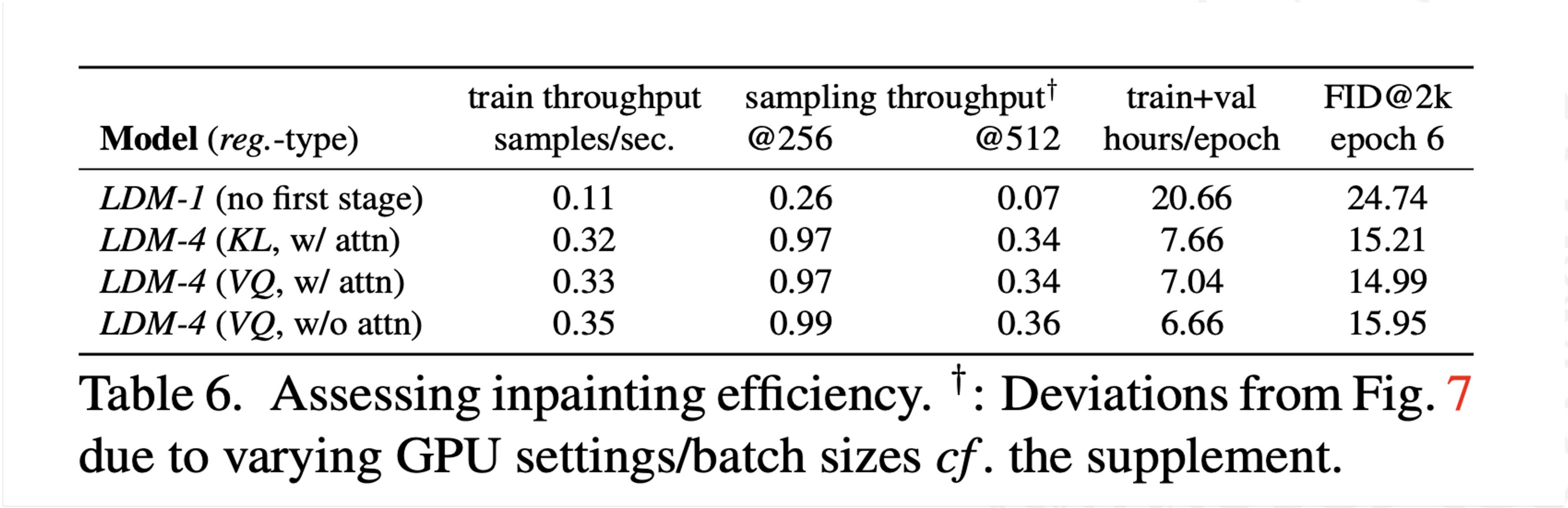
Super-Resolution with Latent Diffusion



	SR on ImageNet		Inpainting on Places	
User Study	Pixel-DM ($f1$)	*LDM-4*	LAMA [88]	*LDM-4*
Task 1: Preference vs GT ↑	16.0%	**30.4%**	13.6%	**21.0%**
Task 2: Preference Score ↑	29.4%	**70.6%**	31.9%	**68.1%**

Table 4. Task 1: Subjects were shown ground truth and generated image and asked for preference. Task 2: Subjects had to decide between two generated images. More details in E.3.6

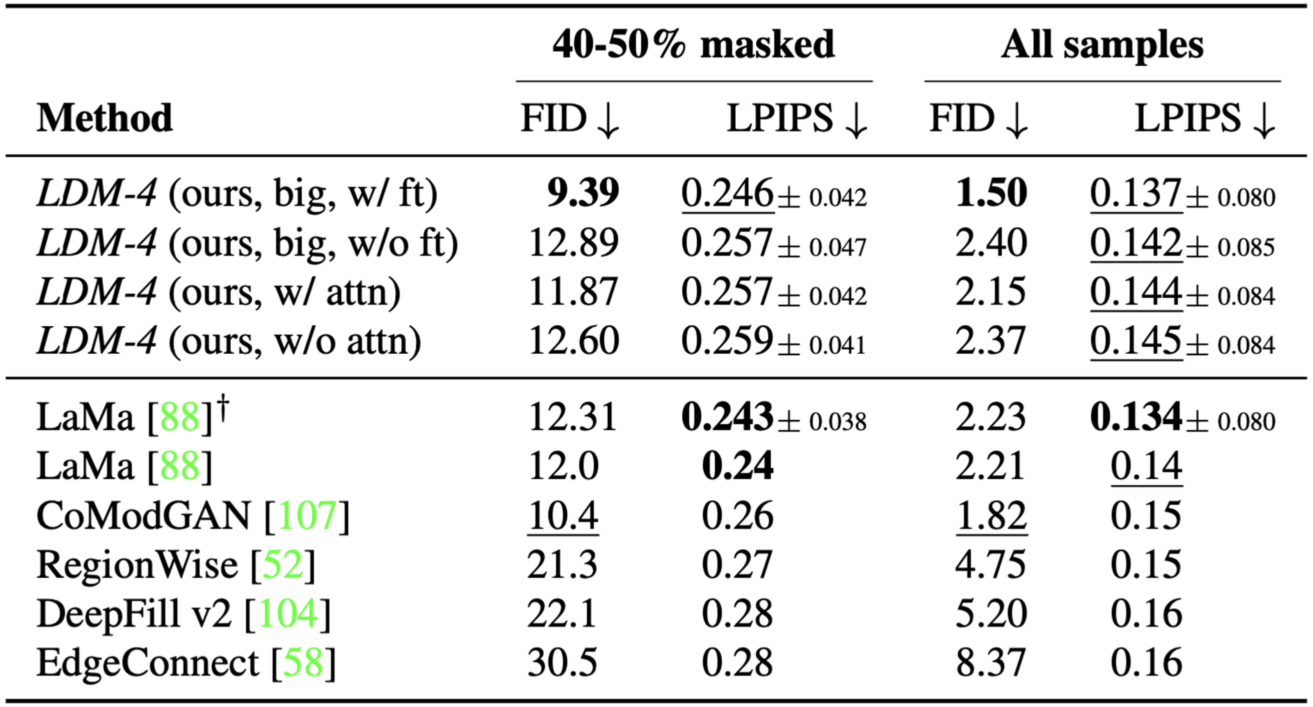
Super-Resolution with Latent Diffusion



Model (*reg.*-type)	train throughput samples/sec.	sampling throughput† @256	@512	train+val hours/epoch	FID@2k epoch 6
LDM-1 (no first stage)	0.11	0.26	0.07	20.66	24.74
LDM-4 (*KL*, w/ attn)	0.32	0.97	0.34	7.66	15.21
LDM-4 (*VQ*, w/ attn)	0.33	0.97	0.34	7.04	14.99
LDM-4 (*VQ*, w/o attn)	0.35	0.99	0.36	6.66	15.95

Table 6. Assessing inpainting efficiency. †: Deviations from Fig. 7 due to varying GPU settings/batch sizes *cf*. the supplement.

Inpainting with Latent Diffusion



	40-50% masked		All samples	
Method	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.

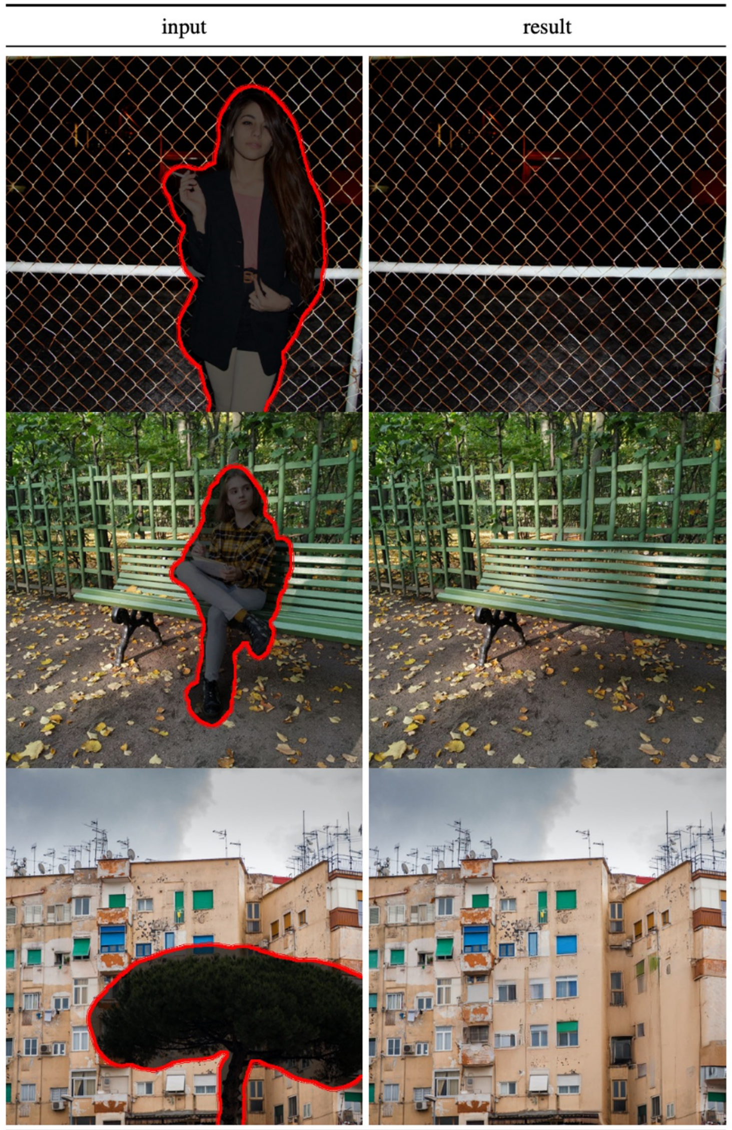
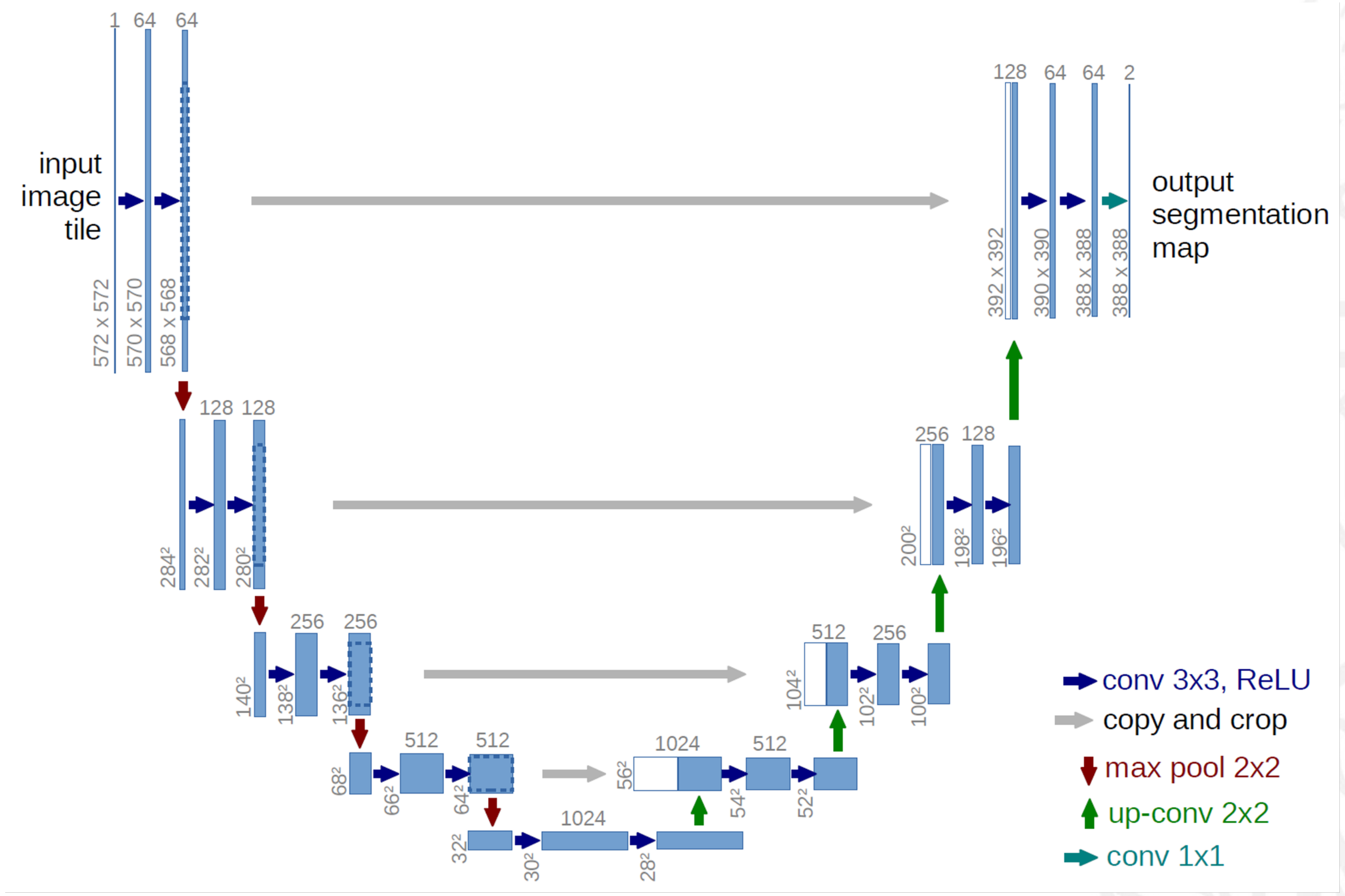


Figure 11. Qualitative results on object removal with our *big, w/*

U-Net Skip Connection



PAPERS REFERRED

- Score Based Models - https://arxiv.org/pdf/1907.05600v3
- Denoising Diffusion Probabilistic Model - https://arxiv.org/pdf/2006.11239
- For Math Derivation - https://lilianweng.github.io/posts/2021-07-11-diffusion-models/, Outlier
- Some improvements done by OpenAI - https://arxiv.org/pdf/2102.09672
- Latent Diffusion Models - https://arxiv.org/pdf/2112.10752

THANK YOU!