Deep Learning for Data Science DS 542

https://dl4ds.github.io/fa2025/

Latent Diffusion Models

Original slides by Tom Gardos,
other content from Understanding Deep Learning unless otherwise cited

Plan for Today

- Diffusion model math
- Training process (again)
- Motivation for latent diffusion
- Latent diffusion
- Conditional generation

Dall-E 3

Mathematics of Diffusion Models

Assume the forward and reverse process operate in T steps.

Both forward and reverse process are discrete so becomes a *Markov chain* with *Gaussian transition probability*.

$\mathcal{N}(\mu, \sigma^2)$

Mathematics of Diffusion Models

$x_0 \to x_1 \to \dots \to x_T$

Denote x_0 as a sample from a distribution $q(x_0)$. original distribution complicated!

Forward process: gaussian transition probability

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \underbrace{\sqrt{(1-\beta_t)}}_{\text{shrink factor}}\, x_{t-1}, \underbrace{\beta_t I}_{\text{noise scale}}) \quad \text{where} \quad t \in \mathbb{N}$$

β_t chosen noise scale at time t

and where β_t indicates trade-off between info to be kept from previous step and new noise added.

$T = 1000$ often for training

sometimes smaller T used for gen speed

$\beta_1 \sim 1e^{-4}$ $\qquad$ $\beta_T \sim 0.02$

mostly signal, small noise

paper recommendations, linearly interpolate in between

$\mathcal{N}(\mu, \sigma^2)$

Mathematics of Diffusion Models

Denote x_0 as a sample from a distribution $q(x_0)$.

Forward process: gaussian transition probability

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{(1-\beta_t)}\, x_{t-1}, \underbrace{\beta_t I}_{\text{variance}}) \quad \text{where} \quad t \in \mathbb{N}$$

and where β_t indicates trade-off between info to be kept from previous step and new noise added.

We can equivalently write

$$x_t = \sqrt{(1-\beta_t)}\, x_{t-1} + \underbrace{\sqrt{\beta_t}}_{\text{std}}\, \epsilon_t \qquad \epsilon_t \sim \mathcal{N}(0, I)$$

Discretized diffusion process

$\boxed{\mathcal{N}(\mu, \sigma^2)}$

Mathematics of Diffusion Models

Through recurrence, we can represent any step in the chain as directly represented from x_0:

cumulative variance

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

cumulatice
scaling from t steps

where

$$\alpha_t = (1 - \beta_t) \quad \text{and} \quad \bar{\alpha}_t = \textstyle\prod_{i=1}^{t} \alpha_i = \prod_{i=1}^{t} (1 - \beta_i)$$

and from the Markov property, the entire forward trajectory is

$$q(x_{0:T}) = q(x_0) \prod_{t=1}^{T} q(x_t|x_{t-1})$$

The reverse process

With the assumption on the drift and diffusion coefficients, the reverse of the diffusion process takes the same form.

Reverse gaussian transition probability

$$q(x_{t-1}|x_t)$$

backwards

said this was a diffusion process too last time but need to learn func

can then be approximated by

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \underbrace{\mu_\theta(x_t, t), \Sigma_\theta(x_t, t)}_{\text{learn mean + variance}})$$

where μ_θ and Σ_θ are two functions parameterized by θ and learned.

The reverse process conditional

give independence so we can just multiply

Using the Markov property, the probability of a given backward trajectory can be approximated by

$$\mathrm{p}_\theta(x_{0:T}) = \underline{p(x_T)} \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

"start" of reverse process

where $p(x_T)$ is an isotropic gaussian distribution that does not depend on θ

$$p(x_T) = \mathcal{N}(x_T; 0, I)$$

b/c we targeted this w/ forward process design

FIXED FORWARD PROCESS

Initial distribution

$q(x_0)$

Gaussian transition kernel

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

$p_\theta(x_0)$ should match $q(x_0)$

learned func

Approximation of

$q(x_{t-1}|x_t)$

Gaussian transition kernel with parameters to be learned

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

Initial distribution

$p(x_T) = \mathcal{N}(x_t; 0, I)$

LEARNED BACKWARD PROCESS

Questions

How do we learn the parameters θ for μ_θ and Σ_θ?

What is the loss to be optimized?

- We hope that $p_\theta(x_0)$, the distribution of the last step of the reverse process, will be close to $q(x_0)$

make these match...

probability distributions comparison

☆ KL divergence

Earth Mover's / Wasserstein

Optimization Objective

truth

distribution being optimized

$$\mu_\theta^*, \Sigma_\theta^* = \underbrace{\arg\min_{\mu_\theta, \Sigma_\theta}}_{\text{pick best params to make distributions close}} \left(D_{KL}(q(x_0)||p_\theta(x_0))\right)$$

$$= \arg\min_{\mu_\theta, \Sigma_\theta} \left(- \int q(x_0) \log \left(\frac{p_\theta(x_0)}{q(x_0)} \right) dx_0 \right)$$

flipped from normal formula

$$= \arg\min_{\mu_\theta, \Sigma_\theta} \left(- \int q(x_0) \log(p_\theta(x_0)) dx_0 \right)$$

b/c $q(x_0)\log(q(x_0))$ does not have θ

Skipping a lot more math

- Expand p-theta as marginalization integral
- Use Jensen's inequality to define a slightly simpler upper bound to the loss
- Some manipulations with Bayes' Theorem
- Properties of KL divergence of two gaussian distributions
- An additional simplification suggested by [Ho et al 2020]

Diffusion models in practice

Simple formula best if iterative.

★ Disclaimer.
Even though total noise was estimated, only remove about 1 step of noise at a time.
One step output is poor.

We have the forward process

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

Alternative:
Estimate x_0 or $\mu(\cdot\cdot)$ directly...

and our reverse process

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

and we want to train to minimize this simplified upper bound ★

$$\mathbb{E}_{x_0,t,\epsilon}\left(||\underbrace{\epsilon}_{\text{actual noise added to sample}} - \underbrace{\epsilon_\theta(x_t, t)}_{\text{estimate of noise.}}||^2\right) = \underbrace{\mathbb{E}_{x_0,t,\epsilon}\left(||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||^2\right)}_{\text{easier formula from paper "a simpler variational bound"}}$$

Dall-E 3

Any Questions?

???

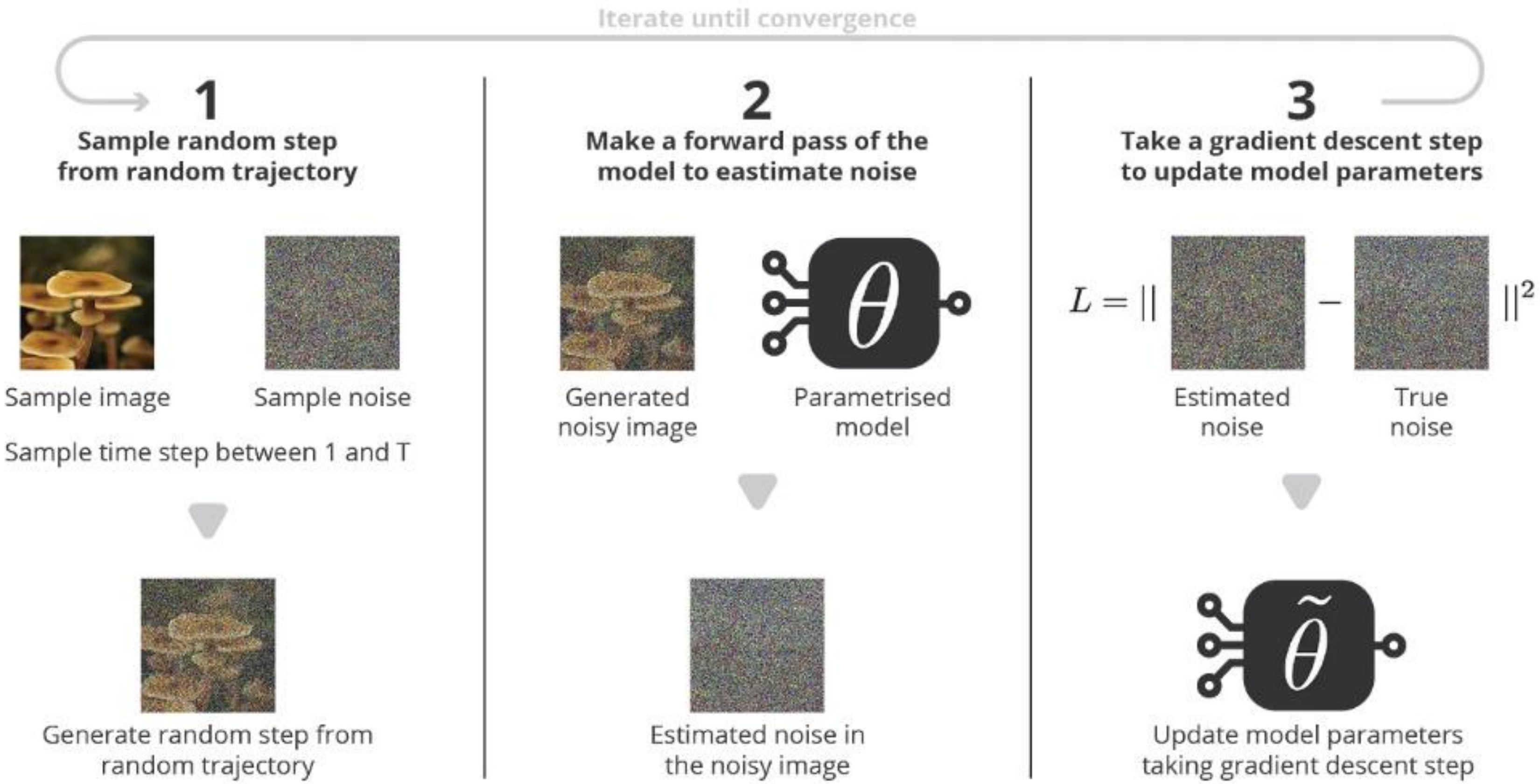
Moving on

- Diffusion model math
- Training process (again)
- Motivation for latent diffusion
- Latent diffusion
- Conditional generation

Training Process

Algorithm 1 Training

1: **repeat**
2: $x_0 \sim q(x_0)$ ▷*Sample random initial data*
3: $t \sim \text{Uniform}(\{1, ..., T\})$ ▷*Sample random step*
4: $\epsilon \sim \mathcal{N}(0, I)$ ▷*Sample random noise*
5: $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$ ▷*Rand. step of rand. trajectory*
6: Take gradient descent step on $\nabla_\theta ||\epsilon - \epsilon_\theta(x_t, t)||^2$ ▷*Optimisation*
7: **until** converged



Rocca, 2022

To sample/generate

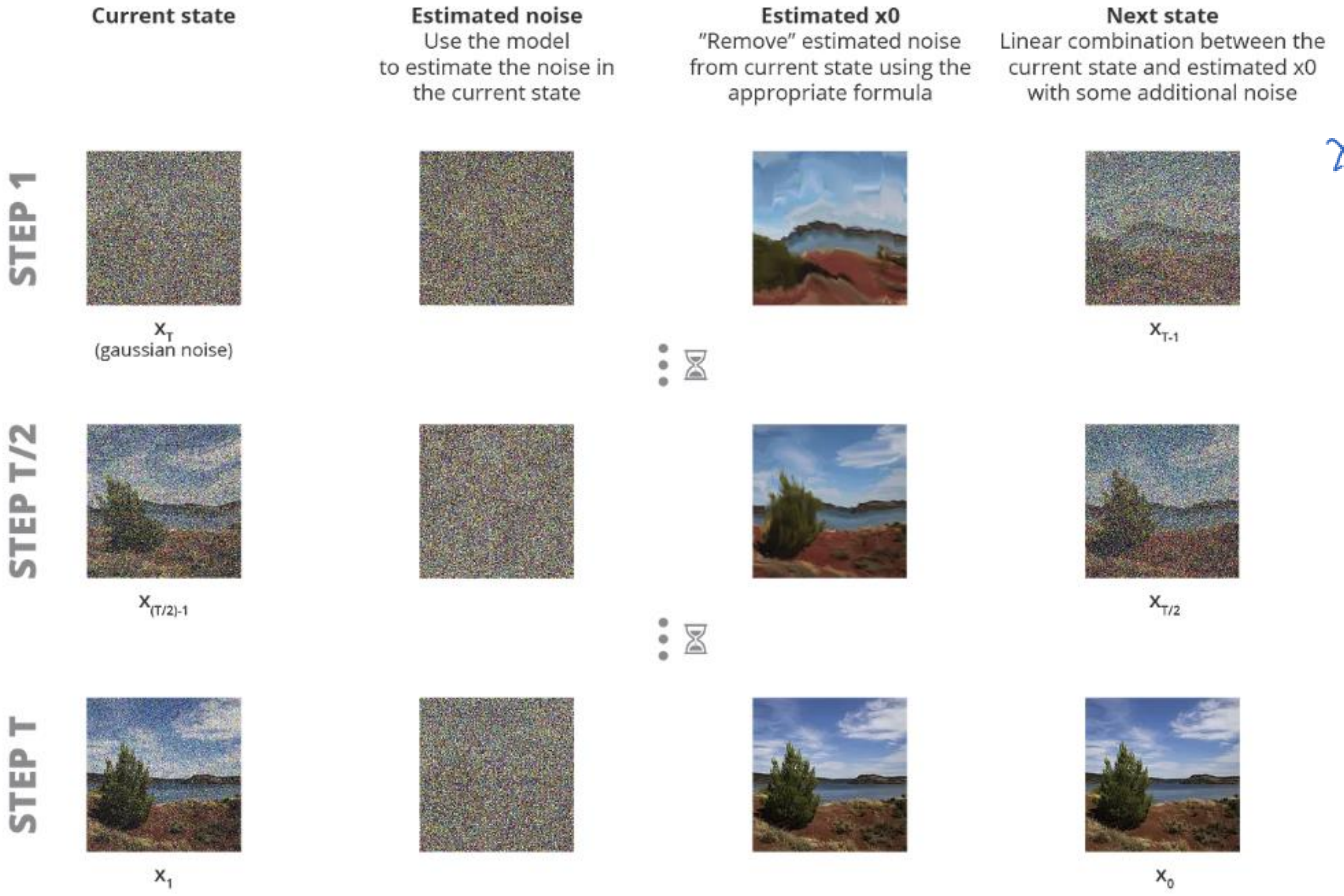


Illustration of the sampling process of a denoising diffusion probabilistic model.

x_0

$x_T \to \tilde{x}_0$

To sample/generate

Algorithm 2 Sampling

$x_T \sim \mathcal{N}(0, I)$ ▷*Initial isotropic gaussian noise sampling*

for t = T, ..., 1 **do**

$\quad z \sim \mathcal{N}(0, I)$ if $t > 1$ else $z = 0$ ▷*Sample random noise (if not last step)*

$\quad \tilde{\epsilon} = \epsilon_\theta(x_t, t)$ ▷*Estimated noise in current noisy data*

$\quad \tilde{x_0} = \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1 - \bar{\alpha}_t}\tilde{\epsilon})$ ▷*Estimated x_0 from estimated noise*

$\quad \tilde{\mu} = \mu_t(x_t, \tilde{x_0}) \left(= \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right)\right)$ ▷*Mean for previous step sampling*

$\quad x_{t-1} = \tilde{\mu} + \sigma_t z$ ▷*Previous step sampling*

end for

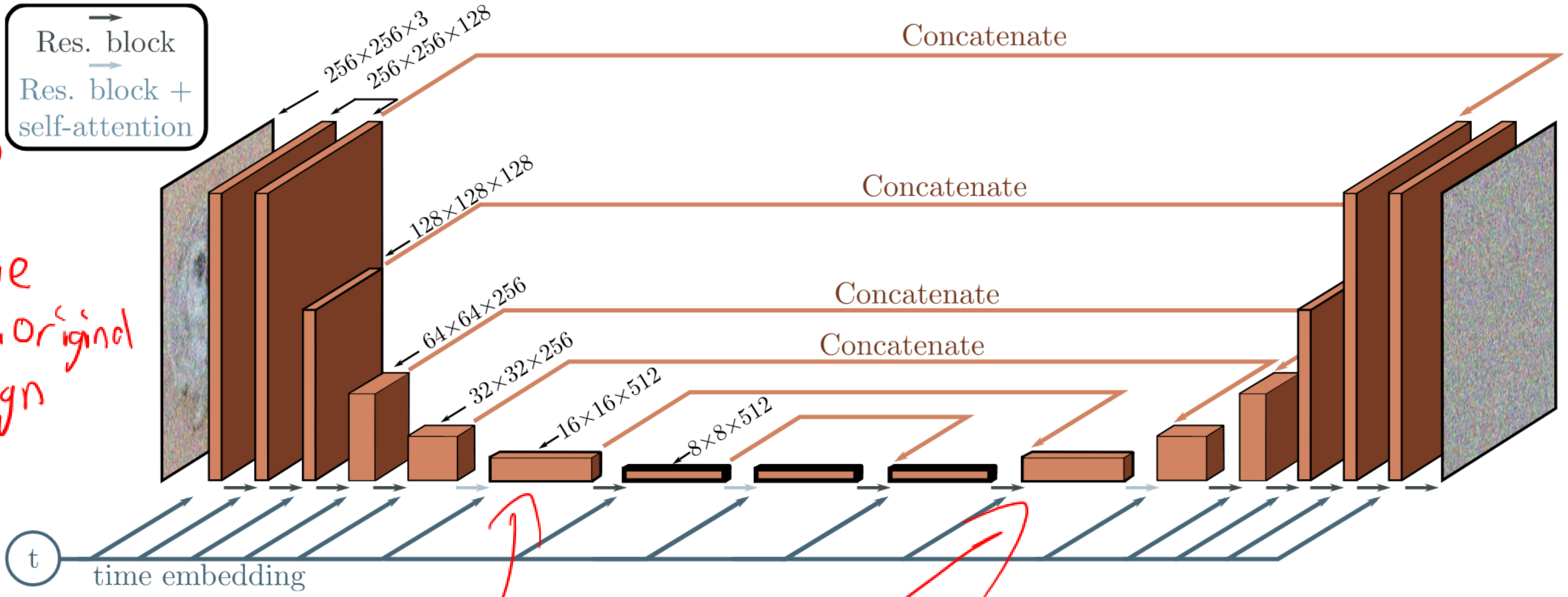
return x_0

Rocca, 2022

U-Net (2016) as the Model

Output is same size as the input.

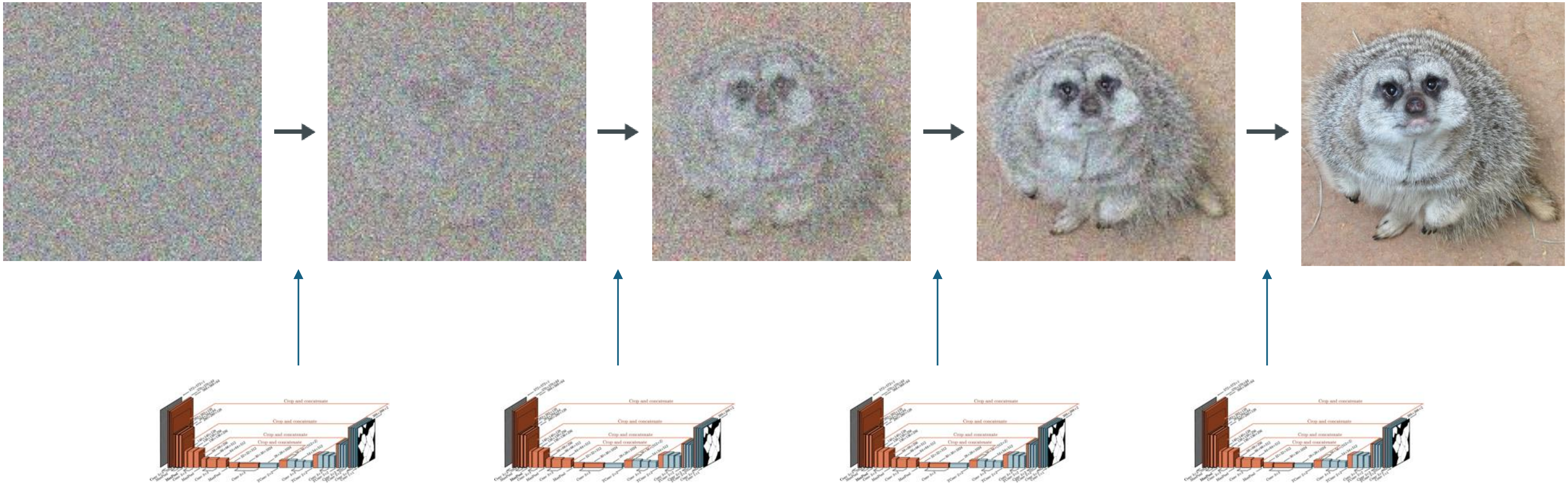
main change from original design



attention here usually

U-Net for reverse diffusion

dog or weasel?



Any Questions?

???

Moving on

- Diffusion model math
- Training process
- Motivation for latent diffusion
- Latent diffusion
- Conditional generation

Hi-Res Conditional Generation via Multi-Resolution Generation

Positive interpretation

low res output ~~specifs~~ specifies overall layout.

high res output fills in consistent details.

Negative interpretation

high resolution version was too hard to do directly

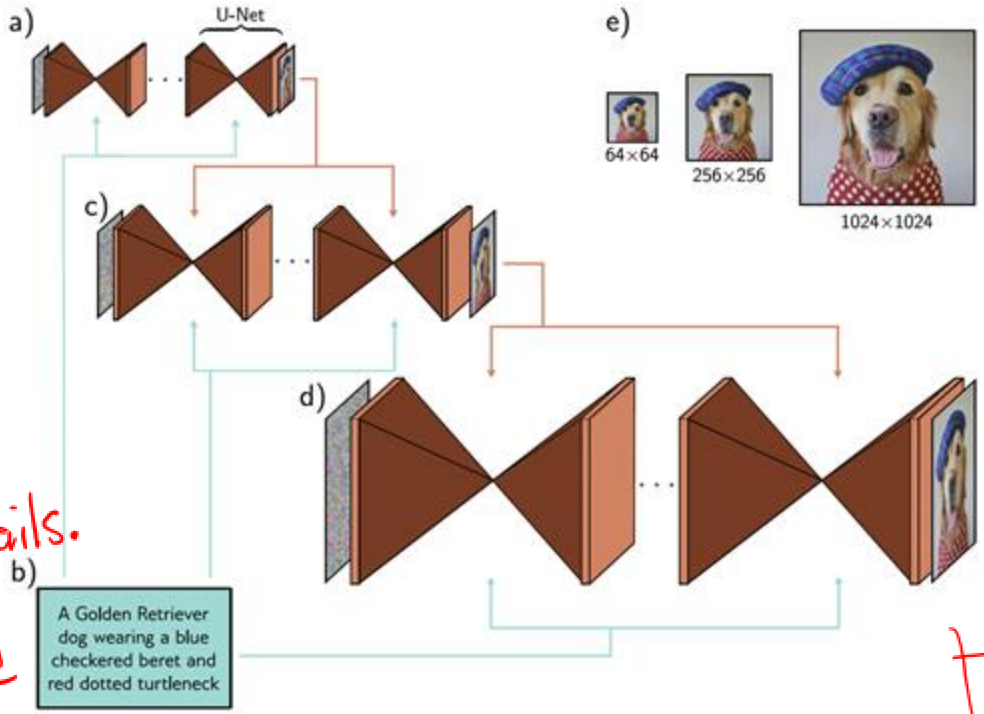


Figure 18.11 Cascaded conditional generation based on a text prompt. a) A diffusion model consisting of a series of U-Nets is used to generate a 64×64 image. b) This generation is conditioned on a sentence embedding computed by a language model. c) A higher resolution 256×256 image is generated and conditioned on the smaller image *and* the text encoding. d) This is repeated to create a 1024×1024 image. e) Final image sequence. Adapted from Saharia et al. (2022b).

leverage small working model/generation to bootstrap bigger generation.

Pyramid complexity often gives net speedup. Almost guaruntees consistency.

Diffusion Models

Eventually got quality comparable to GANs…

- Better than VAEs and normalizing flows, but very very slow…
- Every step of the reverse process is working with full size images.
  - Full size input
  - Full size output
- Multi-resolution architectures are often a sign cost is an issue.
  - But also useful for global consistency, so do not disregard.

Do we have good models?

	GANs	VAEs	Flows	Diffusion
Efficient sampling	✓	✓	✓	✗
High quality	✓	✗	✗	✓
Coverage	✗	?	?	?
Well-behaved latent space	✓	✓	✓	✗
Interpretable latent space	?	?	?	✗
Efficient likelihood	n/a	✗	✓	✗

Diffusion on pixels is slow.

Diffusion on latents will be faster.

Any Questions?

???

Moving on

- Diffusion model math
- Training process (again)
- Motivation for latent diffusion
- Latent diffusion
- Conditional generation

Different Models for Different Tradeoffs

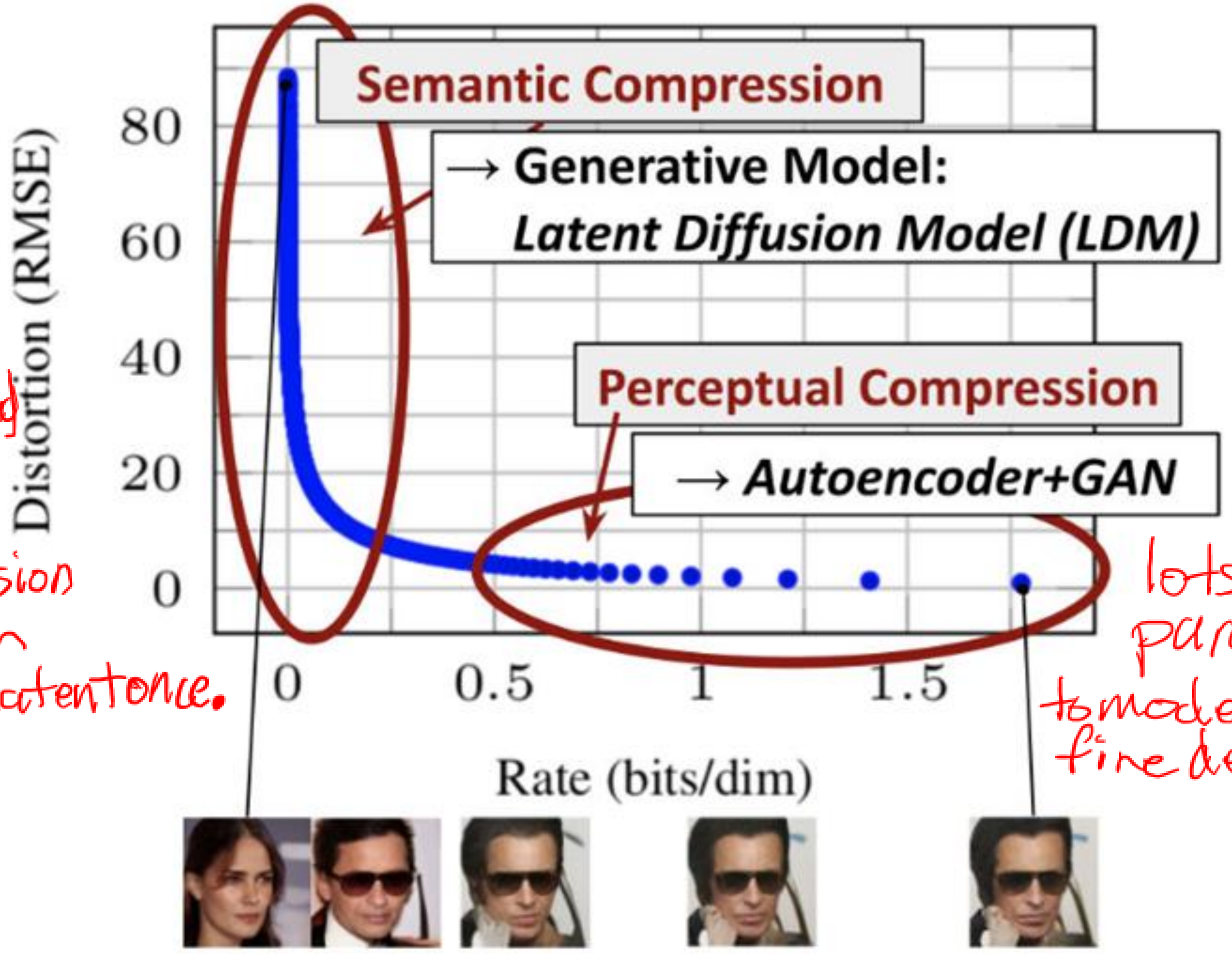
Focus on lower dimension latent model that gets the semantic details right…

- Diffusion in latent space
- Use another high quality model for image generation.

borrow a pretrained model w/small latents. do diffusion on latents. then decode latent once.

"High-Resolution Image Synthesis with Latent Diffusion Models"
By Rombach, Blattman, Lorenz, Esser and Ommer (2021)

you can see all the people/objects but blurry

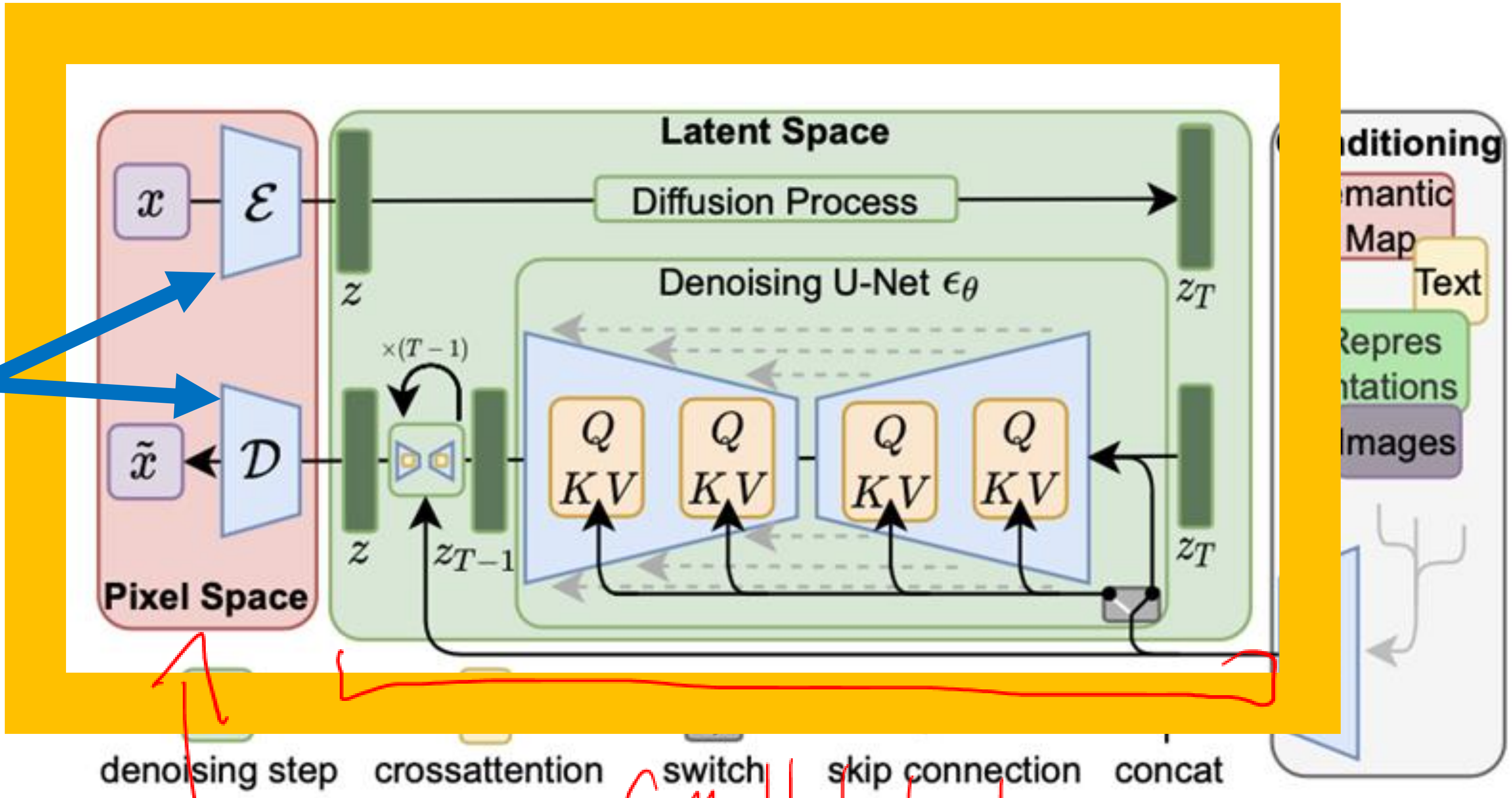


lots of parameters to model fine detail

Combining Auto-Encoder Latents with Diffusion

Pick your favorite autoencoder with a small latent space.

someone already pre-trained



big images

Small latents
1-2 orders of magnitude smaller

Rombach et al, "High-Resolution Image Synthesis with Latent Diffusion Models," 2022

Latent Diffusion

Key idea:

- Pixel space is big.
- Run diffusion process in smaller latent space.

"High-Resolution Image Synthesis with Latent Diffusion Models"
By Rombach, Blattman, Lorenz, Esser and Ommer (2021)

Latent Diffusion Components

Latent diffusion models have two main components.

- Function mapping latent codes to high quality images.
  - This function does not need to have all the qualities we want from generative models.
  - In particular, much of the latent space may not make sense. bad output ok.
- Function mapping noise to latent codes of high quality images. reverse diffusion on latents
  - This is the latent diffusion part.
  - Just this part gets retrained for different applications.

A Random Latent from Stable Diffusion

one cell of coarse grid

```
latent = torch.randn(1, 32, 32, 4)
latent_image = decode_latent(latent)
```

Image shape is 1x256x256x3.

This is that high quality image model?

~~[illegible]~~ x3

https://stability.ai/

Picking a good latent is still a hard problem.

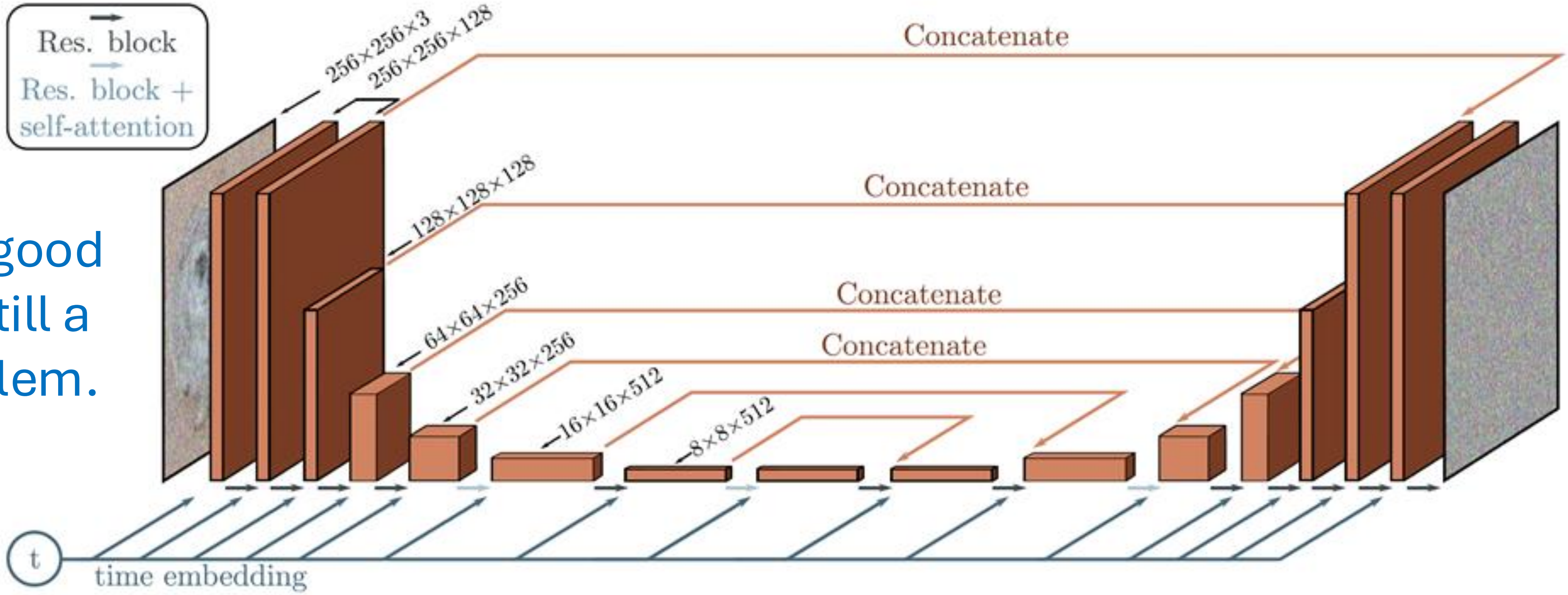


Figure 18.9 U-Net as used in diffusion models for images. The network aims to predict the noise that was added to the image. It consists of an encoder which reduces the scale and increases the number of channels and a decoder which increases the scale and reduces the number of channels. The encoder representations are concatenated to their partner in the decoder. Connections between adjacent representations consist of residual blocks, and periodic global self-attention in which every spatial position interacts with every other spatial position. A single network is used for all time steps, by passing a sinusoidal time embedding (figure 12.5) through a shallow neural network and adding the result to the channels at every spatial position at every stage of the U-Net.

Any Questions?

???

Cited paper "Latent Diffusion"

Stable Diffusion
(stability.ai)

Midjourney

Moving on

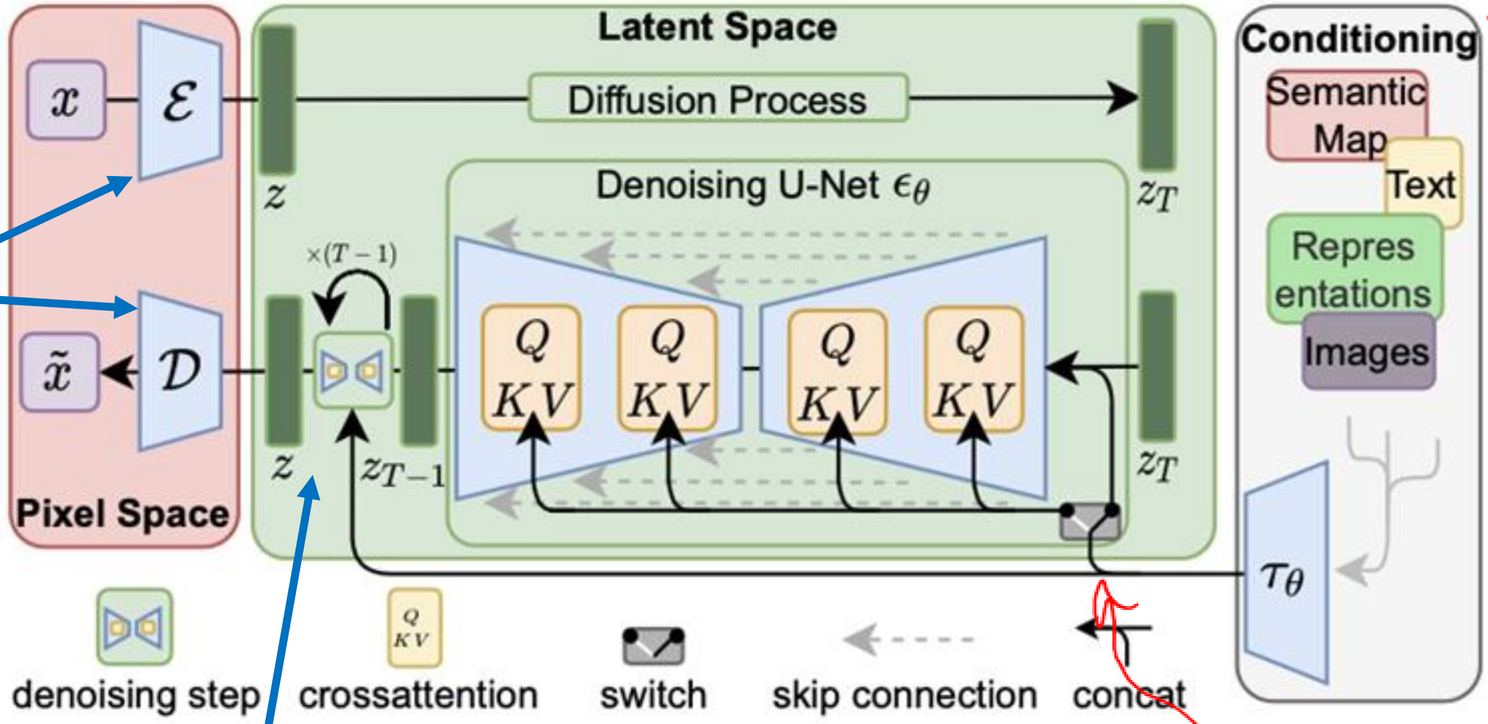
- Diffusion model math
- Training process
- Motivation for latent diffusion
- Latent diffusion
- Conditional generation

Conditioning in Latent Space

How do we control the output?

Conditioning takes over all of latent sampling.

Pick your favorite autoencoder with a small latent space.



collect everything that we want to use for control

concatenate these vectors w/attention inputs

T total repetitions of denoising with conditioning.

Conditioning affects each denoising step.

Rombach et al, "High-Resolution Image Synthesis with Latent Diffusion Models," 2022

Conditional generation using classifier guidance

Figure 18.12 Conditional generation using classifier guidance. Image samples conditioned on different ImageNet classes. The same model produces high quality samples of highly varied image classes. Adapted from Dhariwal & Nichol (2021).

Build a classifier for images, and adjust diffusion steps to also move towards target class.

Also works w/ a classifier for latents.

Limited to cases where classifiers ~~exist~~ exist or can be trained.

Dhariwal and Nichol, "Diffusion Models Beat GANs on Image Synthesis." 2021.

Conditional generation using text prompts

General Version

Stable Diffusion Conditionning

prompt ⇒ latent vector.

add noise to that latent. then diffuse like normal.

A transparent sculpture of a duck made out of glass | An angry duck doing heavy weightlifting at the gym | A brain riding a rocketship heading towards the moon

A couple of glasses sitting on a table | New York skyline with Hello World written with fireworks in the sky

Figure 18.13 Conditional generation using text prompts. Synthesized images from a cascaded generation framework, conditioned on a text prompt encoded by a large language model. The stochastic model can produce many different images compatible with the prompt. The model can count objects and incorporate text into images. Adapted from Saharia et al. (2022b).

language model embedding + latent codes need to be compatible.

CLIP is a good choice.

Text prompt

⇒ language model embedding

⇒ includes as input into diffusion process

(mostly used by attention parts?)

Must be included in training process.

C. Saharia *et al.*, "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding," 2022.

Super Resolution

- Downsample training images 4x
- Upsample with bicubic interpolation.
- Train latent diffusion model to recover the finer-grained details.

“High-Resolution Image Synthesis with Latent Diffusion Models”
By Rombach, Blattman, Lorenz, Esser and Ommer (2021)

Figure 9. 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 [70].

In Painting

Mask some of the image and reconstruct rest of the image to be consistent.

- Don’t change the unmasked part!
- Keeping unmasked part mostly easy because latent code has spatial structure.

“High-Resolution Image Synthesis with Latent Diffusion Models”
By Rombach, Blattman, Lorenz, Esser and Ommer (2021)

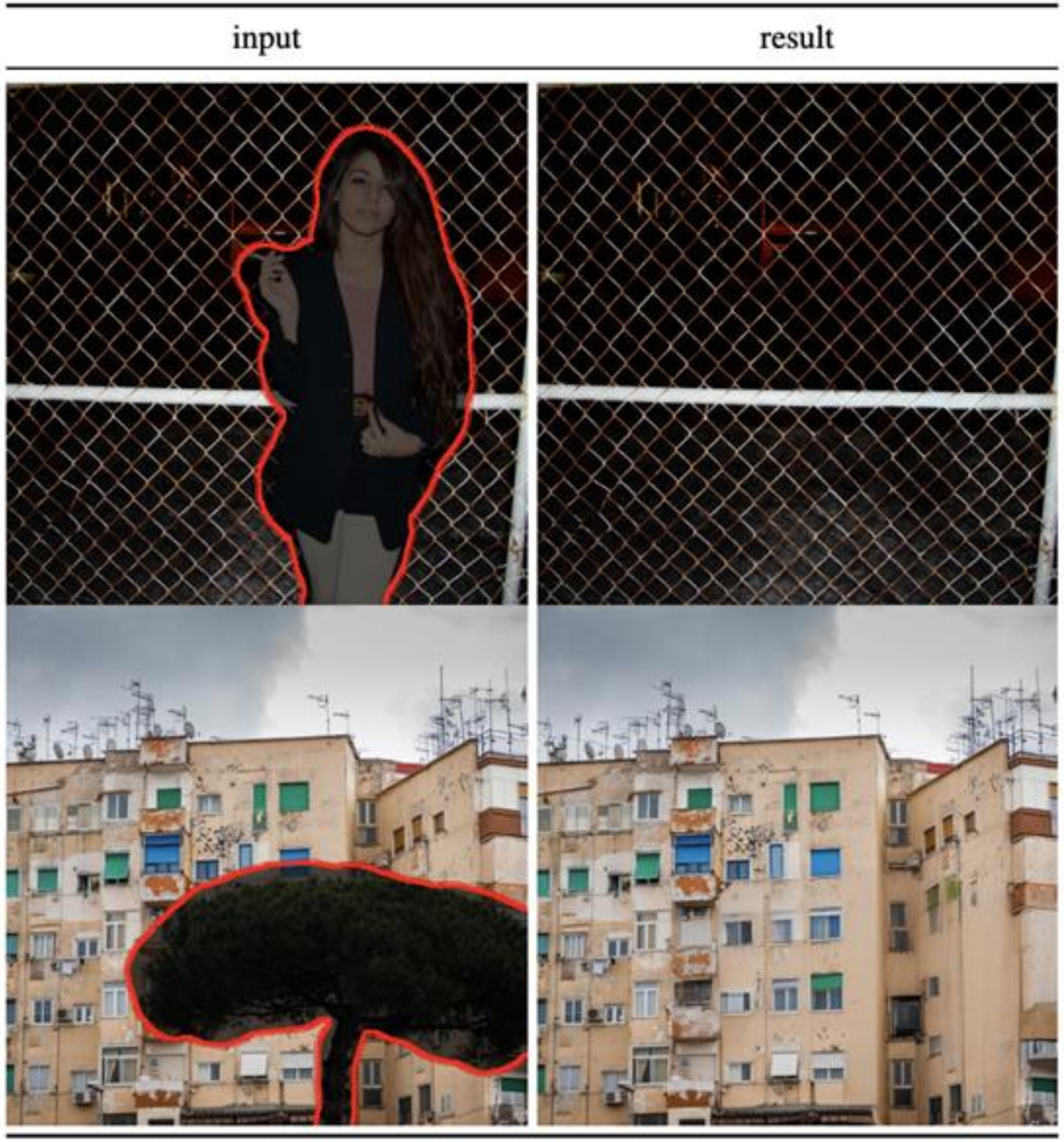


Figure 10. Qualitative results on object removal with our *big, w/ ft* inpainting model. For more results, see Fig. 21.

Example Notebooks from the Book

- Notebook 18.1 – Diffusion Encoder in 1D Open in Colab
- Notebook 18.2 – Training Decoder Open in Colab
- Notebook 18.3 – 1D Reparameterized Model (more robust) Open in Colab
- Notebook 18.4 – Families of Diffusion Models (DDIM) Open in Colab