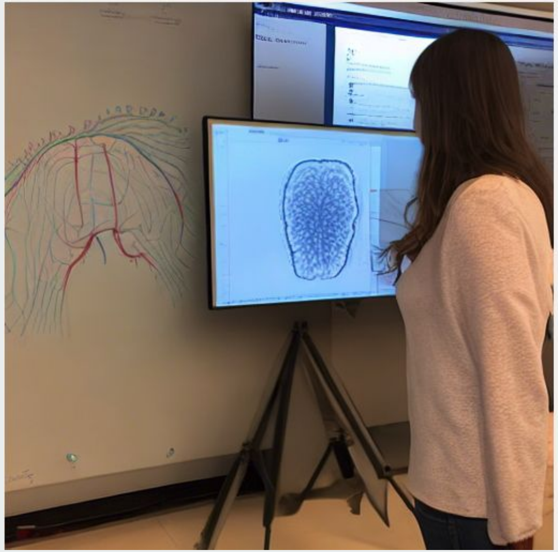
Stable Diffusion

An explosion of interest in Generative AI!



Prompt: Student presenting neuroscience data in Georgia

Generated on my local M2 machine in about 45 seconds using DiffusionBee [4]

What can it do?

1. Class conditional image synthesis
2. Image in-painting
3. **Text-to-image**
4. Unconditional image generation
5. Super-resolution

Focus on 3 because most interesting!

History

Year	Model	Key ideas	Shortcomings
2013	VAE	Encoder-Decoder + prior gives good latents	Blurry images compared to GANs [2] due to surrogate loss (ELBO)
2014	GAN	Noise -> Decoder -> Image + Discriminator	Notorious to train (mode collapse)
2016	Flow	Exact likelihood computation using invertible mapping	Specialized architectures for reversible transforms
2021 - Present	DALL-E2, Imagen	Diffusion + Text encoder	Resource hungry; not feasible on simple machines

Stable Diffusion: Overview

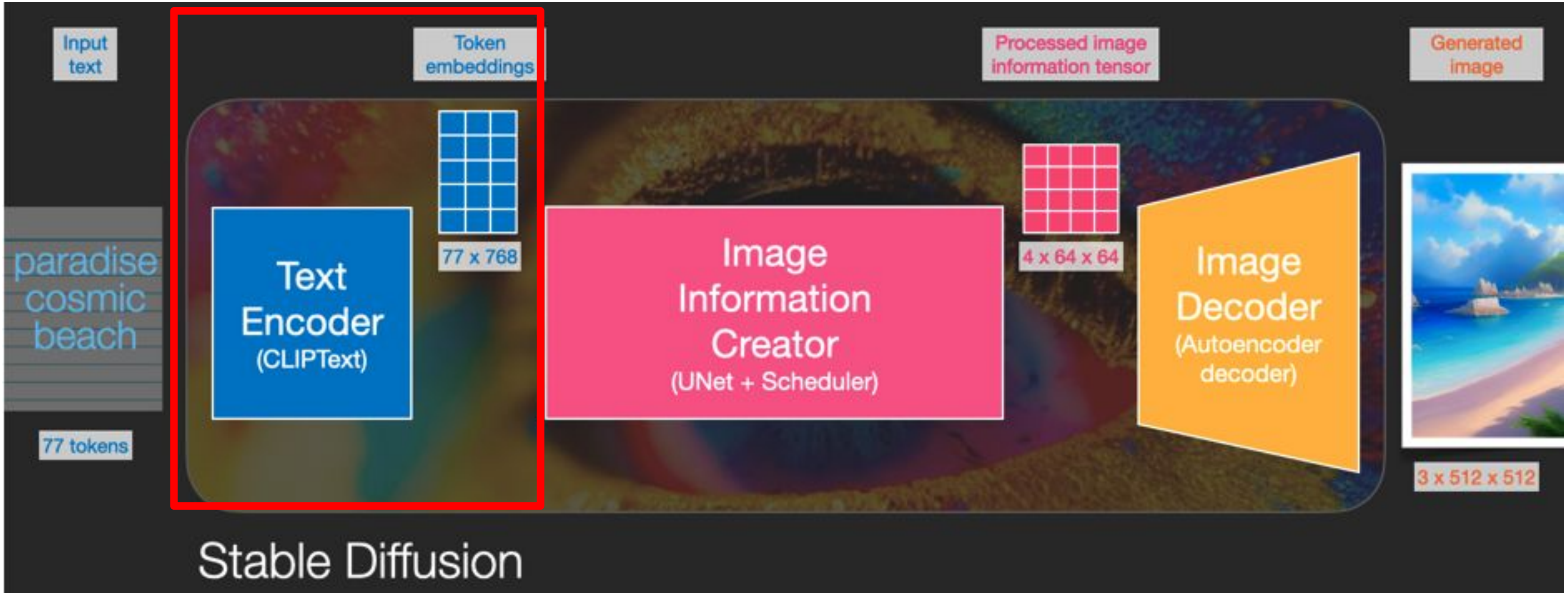


Image from [1]

Text Encoder

- Goal is to **encode relevant details** in the prompt into **numbers**
- How? Transformer language model (of course!)
    - Paper uses BERT (only text)
    - Released model uses ClipText (text component of Contrastive Language-Image Pre-training or CLIP by OpenAI)
- Larger language models do better (shown by Imagen, Google's generative text-to-image model)

How is CLIP trained?

- Take database of image and their captions
- Encodings of text and image of pair should be close (cosine similarity)
- Why is CLIP better?
  -> It has information more fine-tuned for our task, as opposed to generic pre-trained LLMs

Frozen for our task

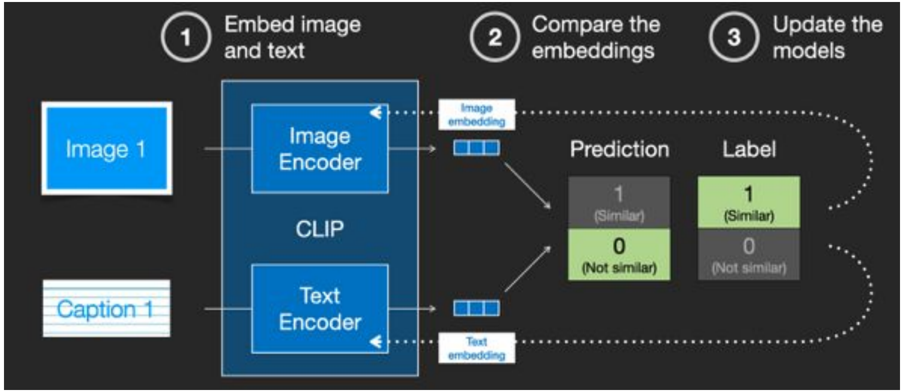


Image from [1]

Stable Diffusion: Overview

Image from [1]

Image Information Creator

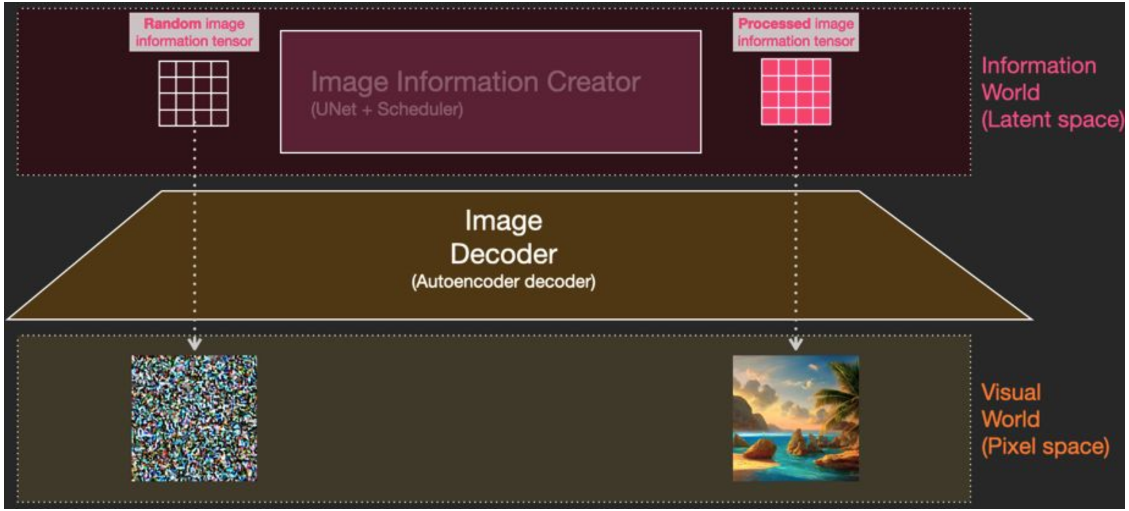


Image from [1]

- Forget text for now
- Start with random noise
- Denoise it using UNet iteratively
- Run for a fixed number of steps (hyperparameter)
- Compare with GAN, which does one shot generation using some decoder -> very ambitious
- Works purely in **latent space (and not pixel space)**

Image Information Creator

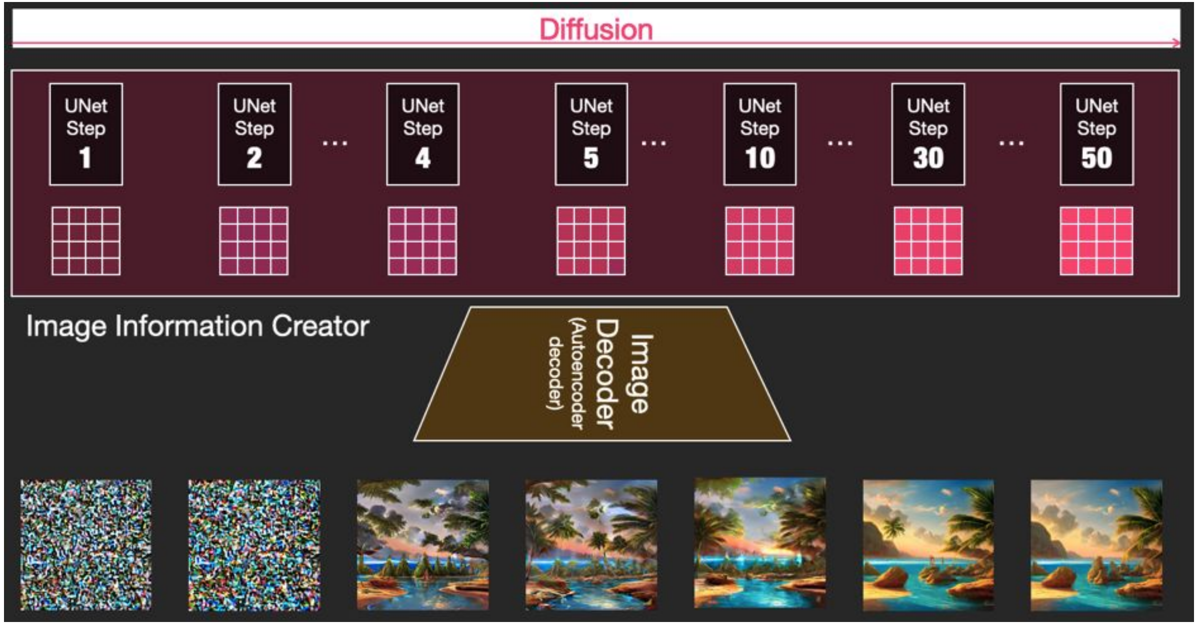


Image from [1]

- Decode it to track sample inside latent space!
- Sudden jump from 2 to 4!

Diffusion: What

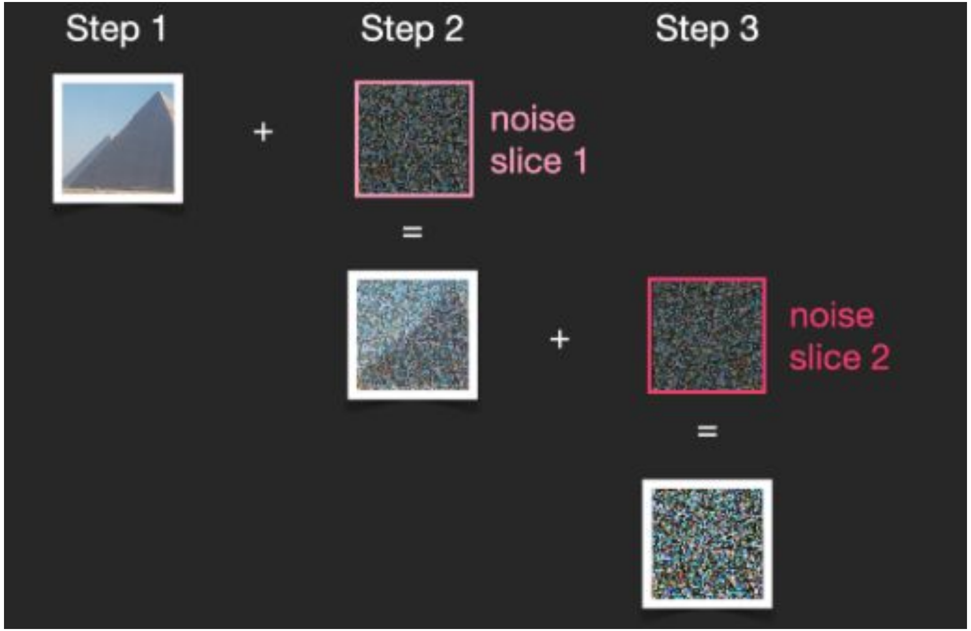


Image from [1]

How do we train the denoiser?
Diffusion!

- Take an image
- Add noise to it from some distribution (Gaussian in SD)
- Keep doing it for some fixed number of steps
- Garbage at the end

Diffusion: What

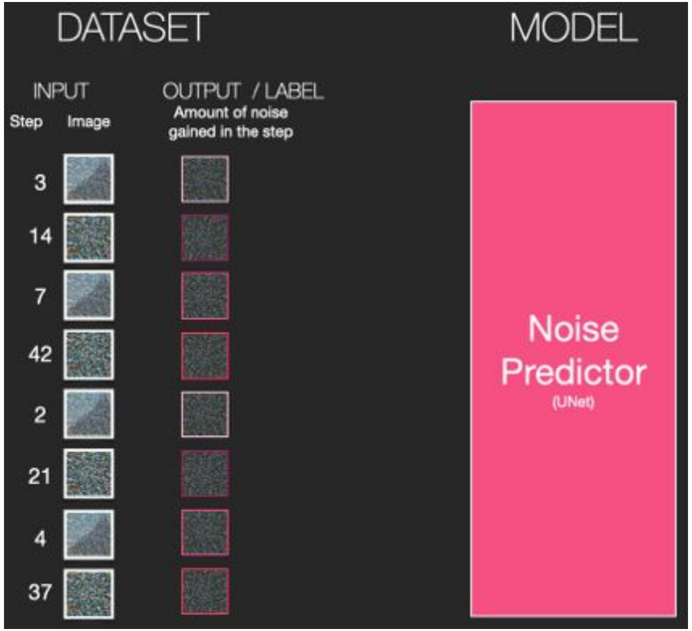


Image from [1]

- Start with clean image X(0)
- Add noise N(t) at time t
- Create dataset with:
  - Input: X(t+1), t
  - Output: N(t)
- UNet predicts noise gained in that step
- Need to also track step number

  Why? Indicates 'stage' we are in

Inside the UNet

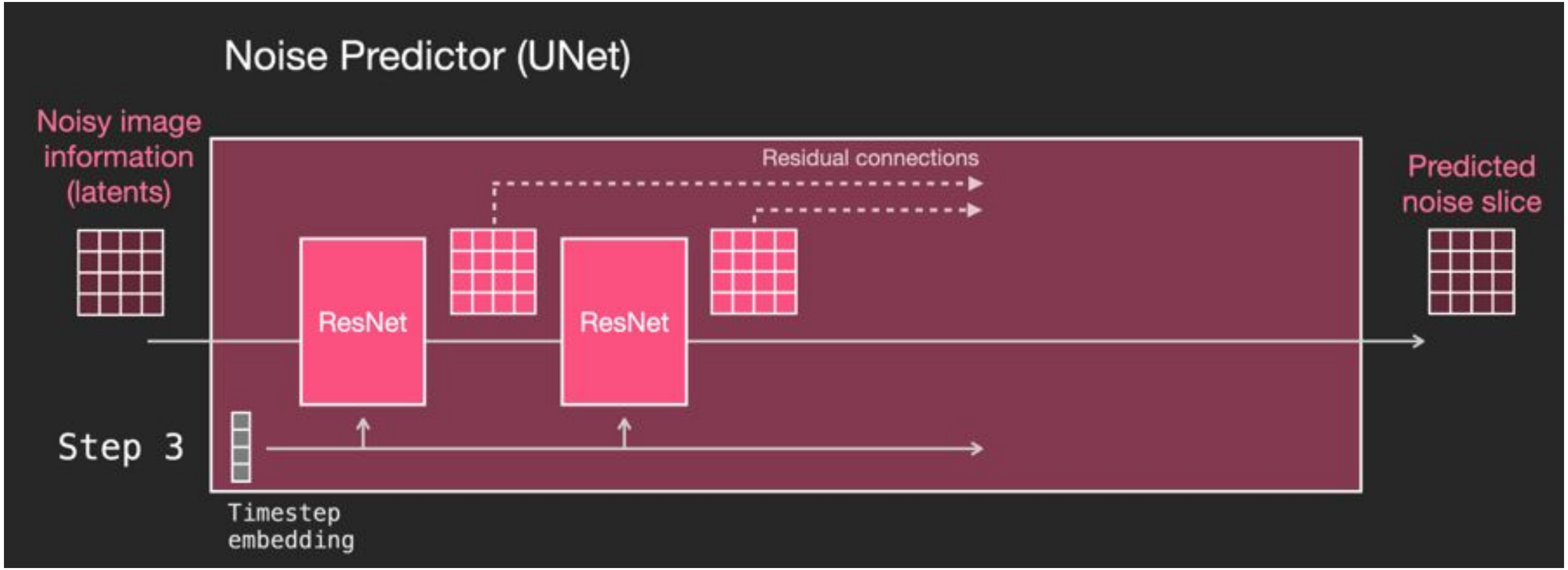


Image from [1]

More details

- UNet because we have a encoder-decoder style bottleneck
- Skip connections help get rid of max pooling
- Sinusoidal timestep embedding

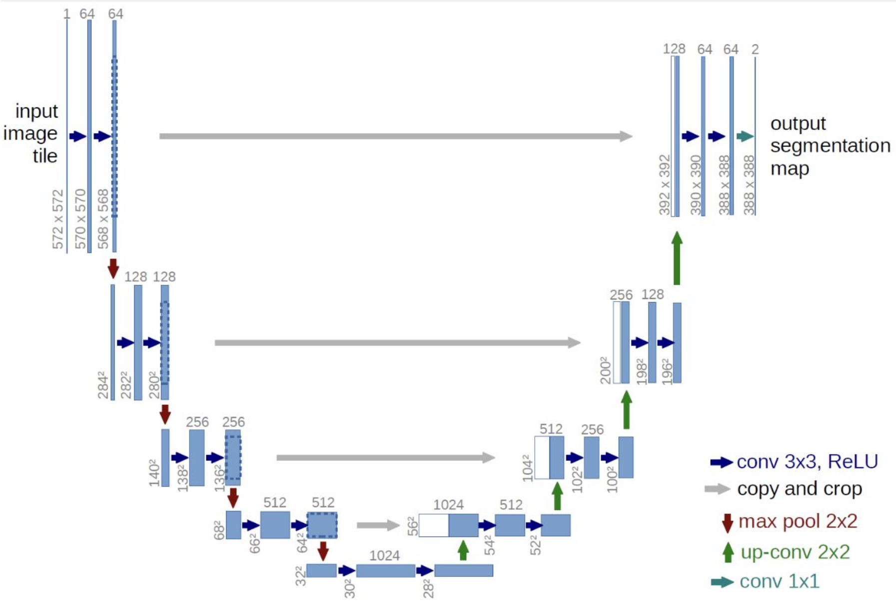


Image from U-Net: Convolutional Networks for Biomedical Image Segmentation

Diffusion: Why?

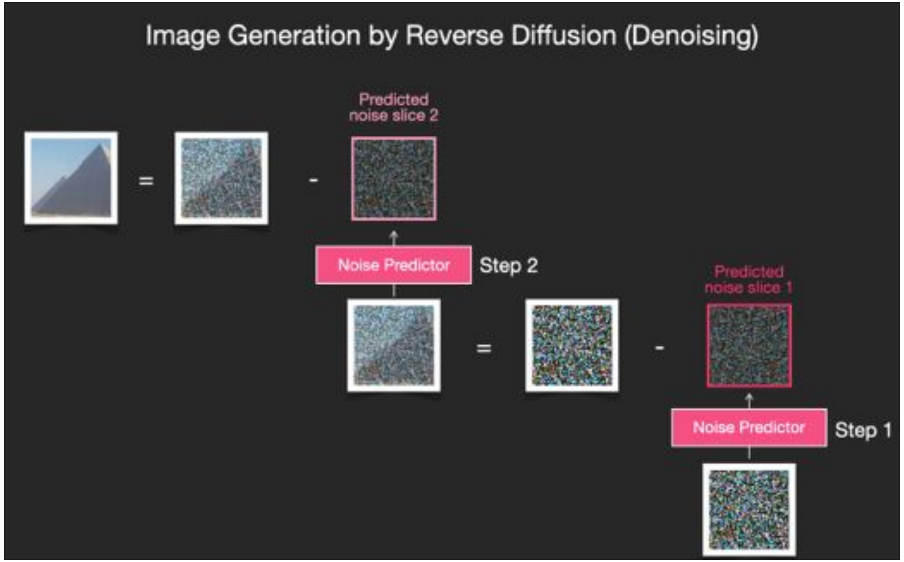


Image from [1]

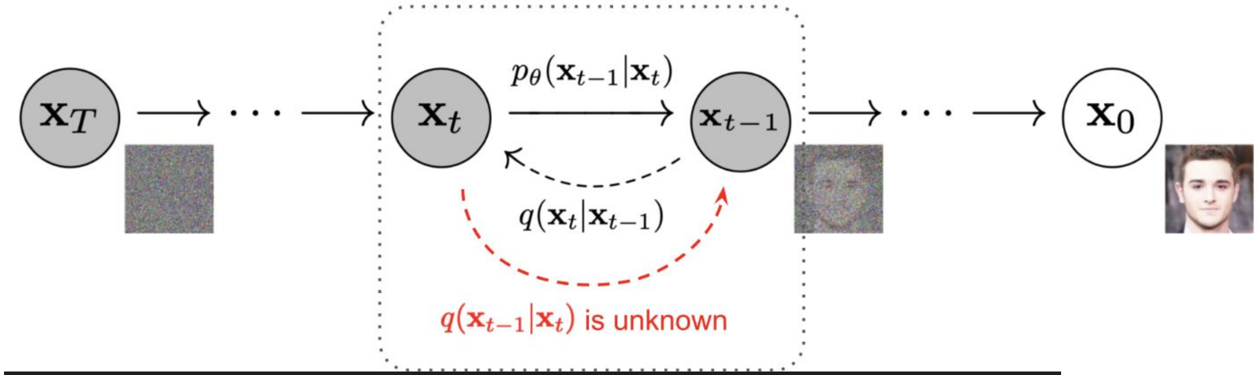
- Start with random noise
- Denoise iteratively using UNet
- If trained properly, image fidelity should improve at each step!
- Diffusion at the core of both Dall-E 2 and Google Imagen!

More on Diffusion

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

Forward process

Use variational lower bound



From [What are Diffusion Models? | Lil'Log](https://lilianweng.github.io/posts/2021-07-11-diffusion-models/)

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

Backward process parameterized by NN

From [2]

Diffusion Loss

$$\text{Let } L_{\text{VLB}} = \mathbb{E}_{q(\mathbf{x}_{0:T})}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] \geq -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0)$$

[Can be proven using Jensen's]

From [2]

$$= \mathbb{E}_q[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^{T} \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}]$$

- L_T: can be ignored since q has no learnable params & x_T is Gaussian noise
- q(x_{t-1}|x_t, x_0) can be computed in closed form:
  Intuitively makes sense since we know starting point
  -> L_{t-1} can be computed using closed form since both Gaussians

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I})$$

Diffusion Loss

$$\begin{aligned}\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} \\ &= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}_{t-2} \\ &= \dots \\ &= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}\end{aligned}$$

- It turns out that:

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

$$\text{Thus } \mathbf{x}_{t-1} = \mathcal{N}\Big(\mathbf{x}_{t-1}; \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\Big), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)\Big)$$

- So, instead of predicting the entire mean, we only predict the noise, making the task *easier* since we know x_t

- Covariance is only a function of alphas and betas (known beforehand):

$$\tilde{\beta}_t = 1/\Big(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\Big) = 1/\Big(\frac{\alpha_t - \bar{\alpha}_t + \beta_t}{\beta_t(1-\bar{\alpha}_{t-1})}\Big) = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\cdot\beta_t$$

From [2]

Summary

Algorithm 1 Training

repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim \mathrm{Uniform}(\{1, \dots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Take gradient descent step on
  $$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
until converged

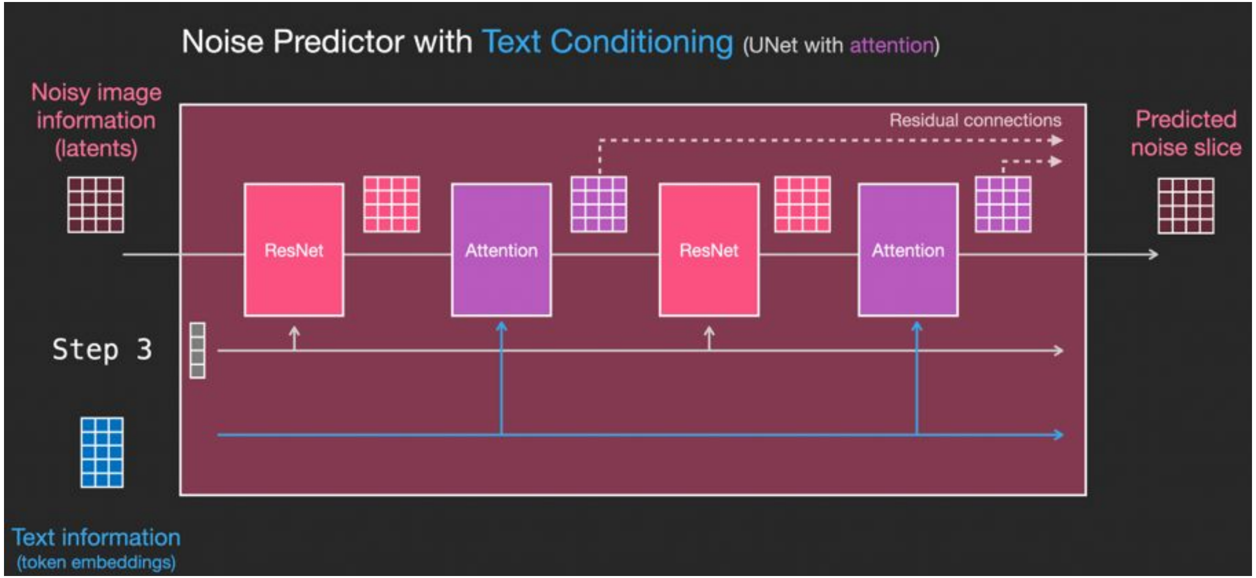
Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$

From [2]

Q. Surrogate loss! Why is it better than VAEs?

Condition on text



From [1]

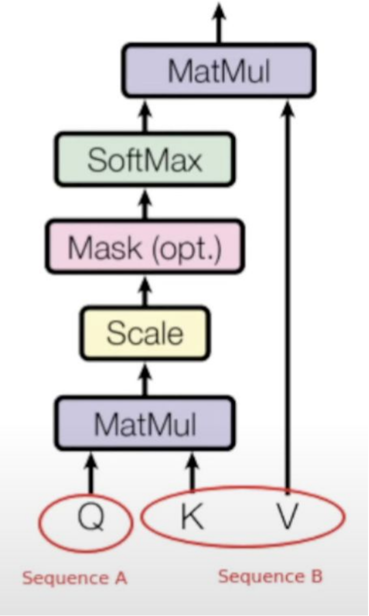
- We don't want to generate **any image;** it should correspond to the text
- Condition on **encoded** prompt using **Cross-Attention**

Cross-Attention

What:

- Way to combine information from two sequences A and B (could be of different lengths)
- Called self-attention when A = B
- A: Query
  E.g. video of ‘cat’
- B: Key and Values
  E.g. ‘title’ and ‘video id’

Generic: can be done for any modalities!
E.g. Text and Image in our case



Cross-Attention

In our case:

- Q = image [what to retrieve]
- K = τ(y)
- V = τ(y)

According to me, Q should've been τ(y) and K, V should've been image?

- Map y (prompt) to an intermediate representation τ (using CLIP in our case)
- Phi(z_t) obtained by flattening the latent

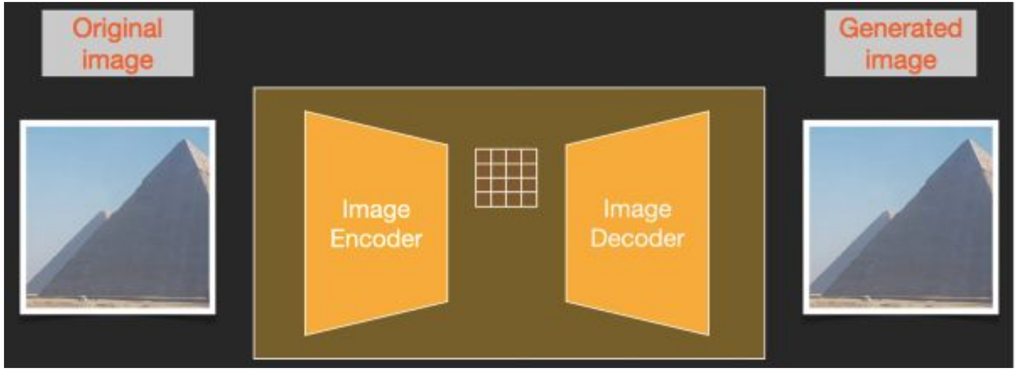
$$Q = W_Q^{(i)} \cdot \varphi_i(z_t), \; K = W_K^{(i)} \cdot \tau_\theta(y), \; V = W_V^{(i)} \cdot \tau_\theta(y)$$

From [3]

Image Decoder

From [1]

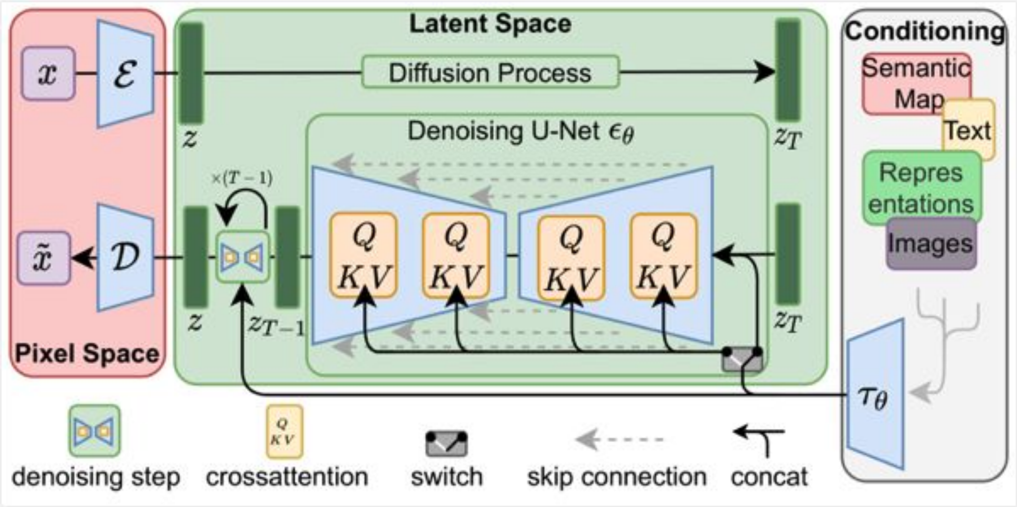
Image Decoder



From [1]

- During training, we need image latent to start with
- How do we obtain it? -> Autoencoder!
- Downsample by factor f along both dims
- 2 types of penalties:
    - KL-reg (standard)
    - VQ-reg

Putting it together



From [3]

Evaluation Metrics

Fréchet inception distance (FID):

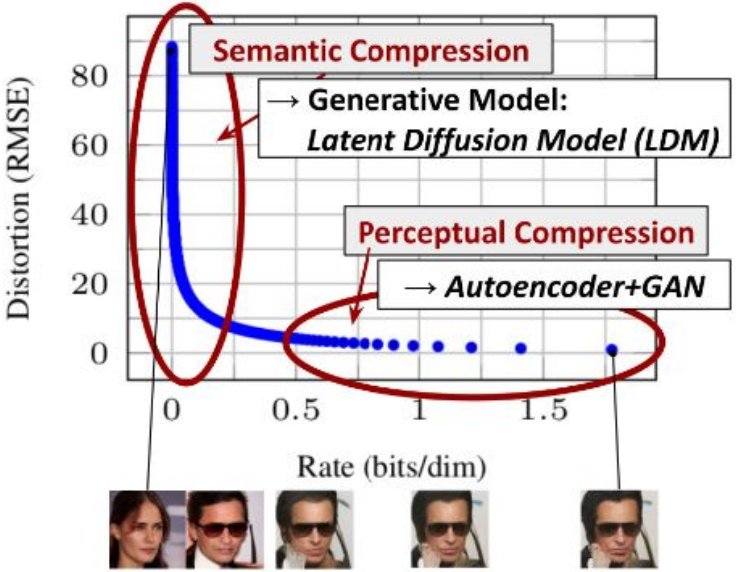
- Take set of real images X and generated images Y
- Extract output of penultimate layer of pre-trained InceptionV3
- X = {x_1, x_2, ...}, Y = {y_1, y_2, ...} where r_i and g_j are 2048-dimensional
- Fit Gaussians for both
- Fréchet distance between both (Why Frechet? IS uses KL-div)

Intuition:

- Mean - Close to real space
- Only mean -> collapse -> need diversity -> 2nd term

$$\mathrm{FID} = ||\mu_X - \mu_Y||^2 - \mathrm{Tr}(\Sigma_X + \Sigma_Y - 2 \quad \Sigma_X \Sigma_Y)$$

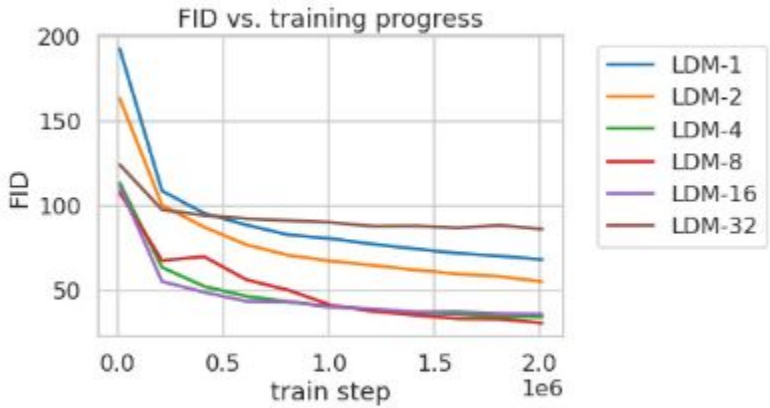
Perceptual vs Semantic compression



From [3]

- A lot of bits required to encode imperceptible details
- Think of JPEG compression: high frequency components can be left out!
- Similarly, we want to be close to the transition point of the graph

Results



From [3]

LDM-f: downsample by in autoencoder

- High factor - low FID initially but stagnates due to information loss
- Low factor - slow training

  Don't know why overall loss is lower though; could decrease if run for more steps as slope is not zero, unlike lower ones

Results

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	$\mathbf{30.29}_{\pm 0.42}$	1.45B	250 DDIM steps, c.f.g. [32] $s = 1.5$

From [3]

- DDIM steps: denoising diffusion **implicit** model - faster way of sampling
- Can compare steps across models using this
- Comparable performance with 25% parameters!

Takeaways

- Main contribution: Diffusion in **Latent Space** (not pixel space) -> heavy speedup
  Can run on simple machines (such as mine)
- Lots of sampling strategies
- Sequential sampling -> slower than GANs
- Fully open source implementation + weights (unlike Imagen and DALL-E2)
- Can do much more:
  - Image in-painting
  - Super-resolution
- What does *Stable* in Stable Diffusion refer to? The company Stability AI?

Resources

[1] The Illustrated Stable Diffusion – Jay Alammar

[2] Weng, Lilian. (Jul 2021). What are diffusion models? Lil'Log. https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

[3] Rombach, Robin, et al. "High-resolution image synthesis with latent diffusion models." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022.

[4] GitHub - divamgupta/diffusionbee-stable-diffusion-ui: Diffusion Bee

Helpful: The Annotated Diffusion Model