Deep Learning for Vision & Language

Diffusion Models II

RICE UNIVERSITY

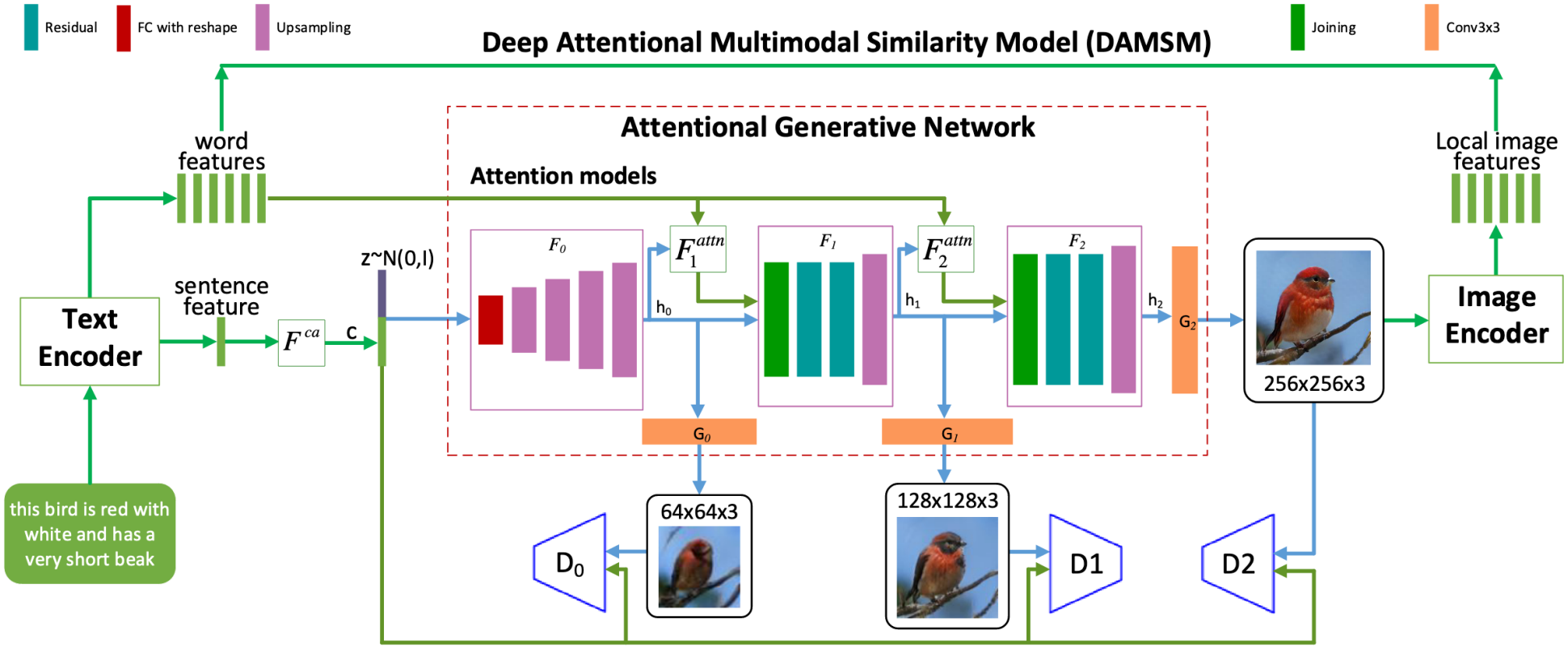
Last Class

- Text to image Models
- Sequence-to-sequence based text-to-image models
- Detour: Style Transfer – Input Feature Optimization.

Today:

- Reverse Diffusion Models
- Other Topics

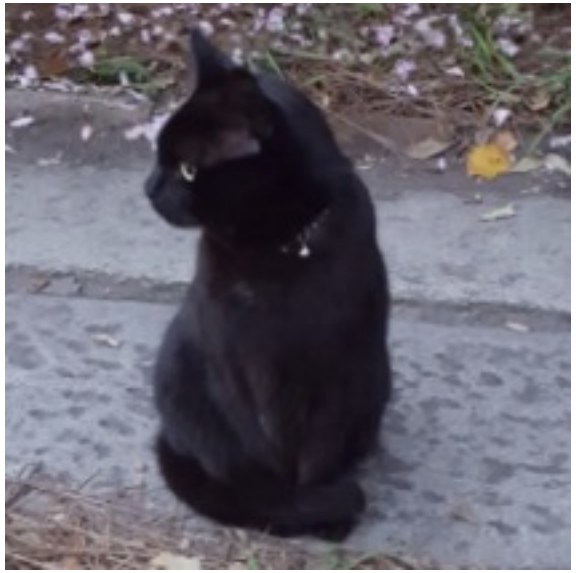
(1) Hierarchical Conditional GANs / Text-conditioned (AttnGAN)

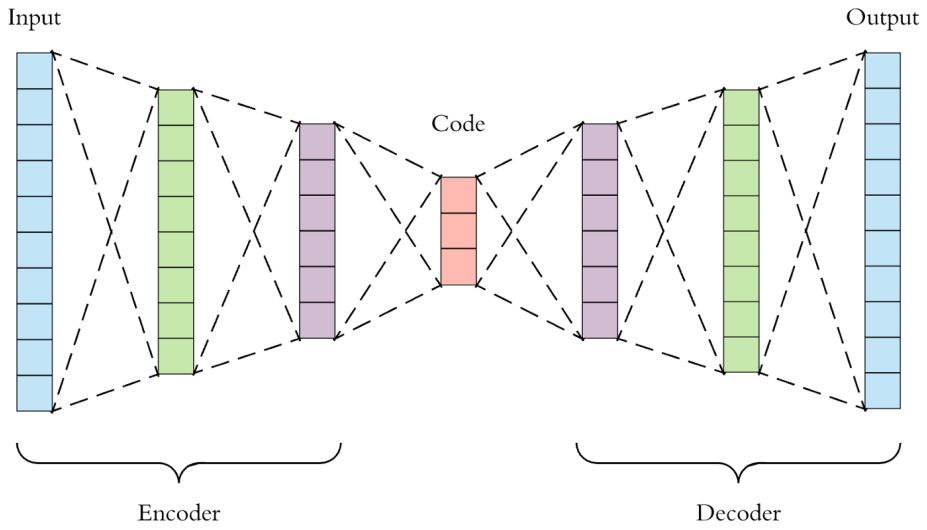


AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks

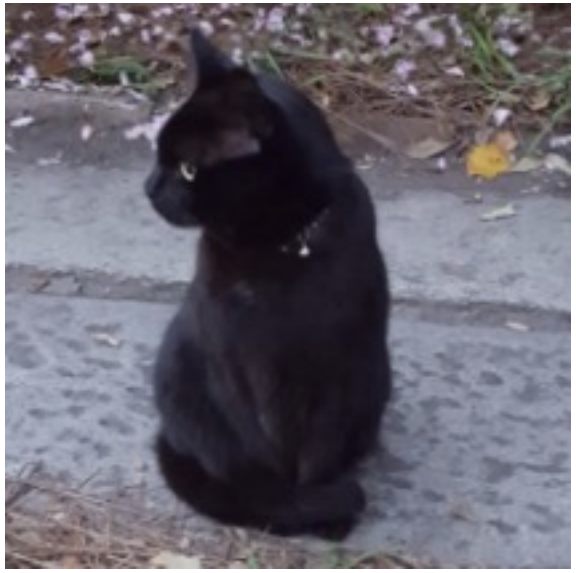
Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, Xiaodong He

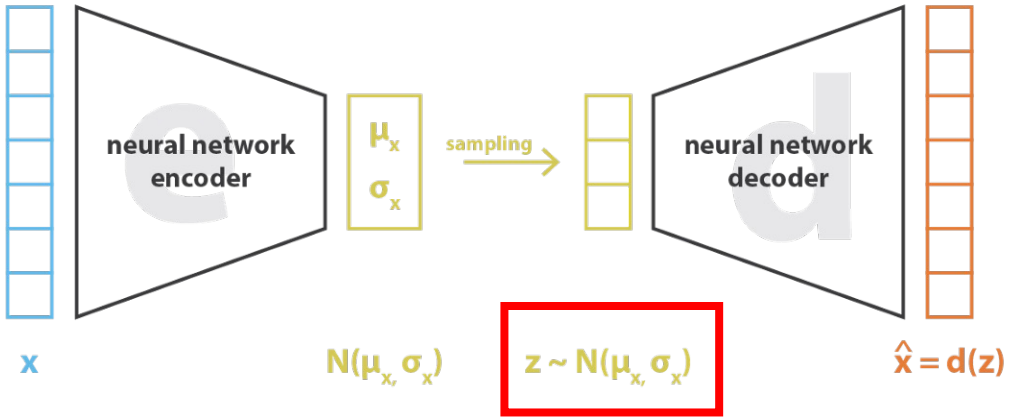
AutoEncoder Models





Variational AutoEncoder (VAE)



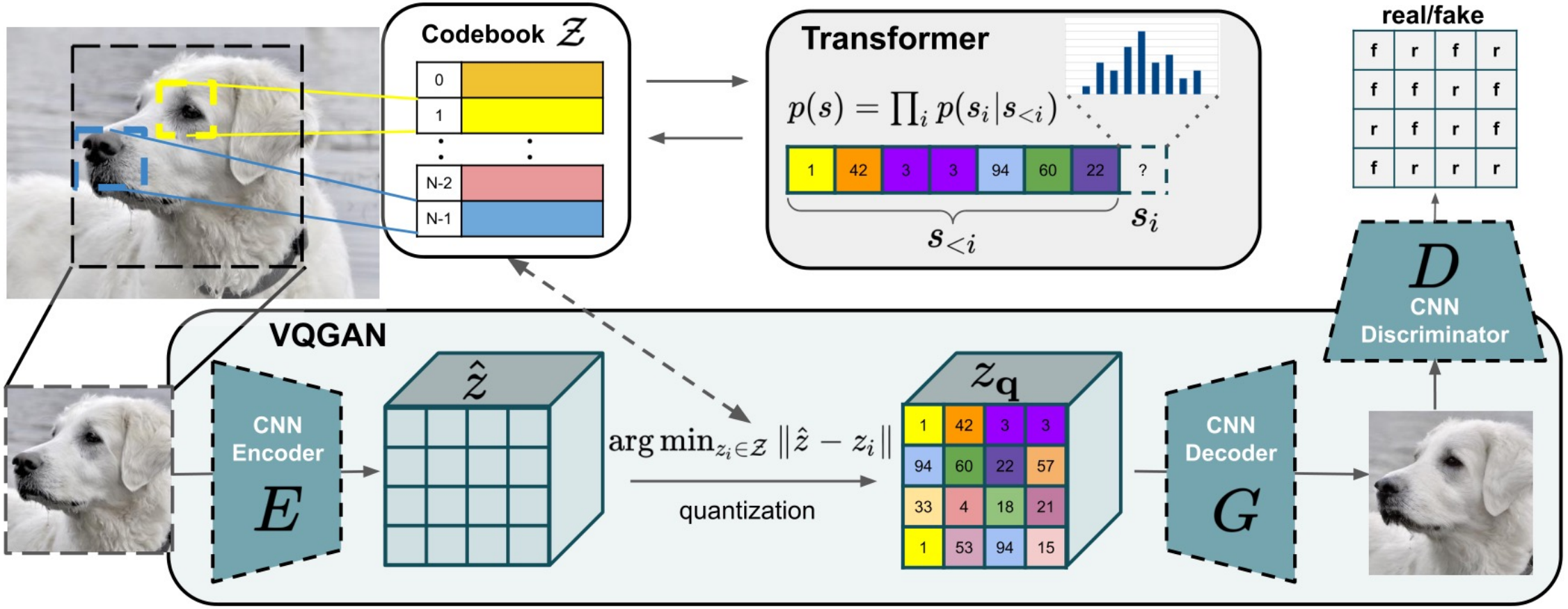


$$\text{loss} = \|x - \hat{x}\|^2 + KL[\,N(\mu_x, \sigma_x), N(0, I)\,]$$

https://pytorch.org/docs/stable/generated/torch.normal.html

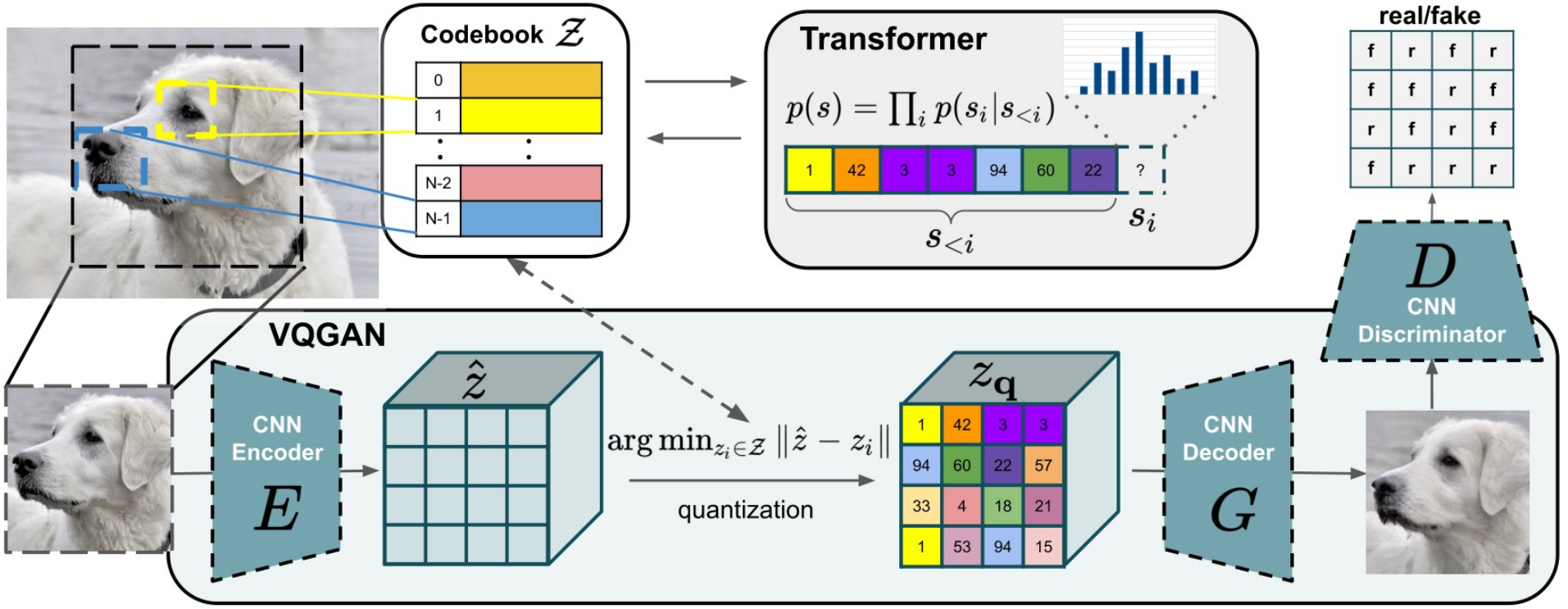
https://ai.stackexchange.com/questions/30176/are-mean-and-standard-deviation-in-variational-autoencoders-unique

Vector Quantized - GAN



https://arxiv.org/abs/2012.09841 https://arxiv.org/pdf/2110.04627.pdf

Vector Quantized GAN (VQGAN)



$$\mathcal{Q}^* = \arg\min_{E,G,\mathcal{Z}} \max_D \mathbb{E}_{x\sim p(x)} \Big[\mathcal{L}_{\text{VQ}}(E,G,\mathcal{Z}) + \lambda \mathcal{L}_{\text{GAN}}(\{E,G,\mathcal{Z}\},D) \Big]$$

$$\mathcal{L}_{\text{VQ}}(E,G,\mathcal{Z}) = \|x - \hat{x}\|^2 + \|\text{sg}[E(x)] - z_{\mathbf{q}}\|_2^2 + \|\text{sg}[z_{\mathbf{q}}] - E(x)\|_2^2.$$

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

https://arxiv.org/abs/2012.09841 https://arxiv.org/pdf/2110.04627.pdf

DALL-E (v1)

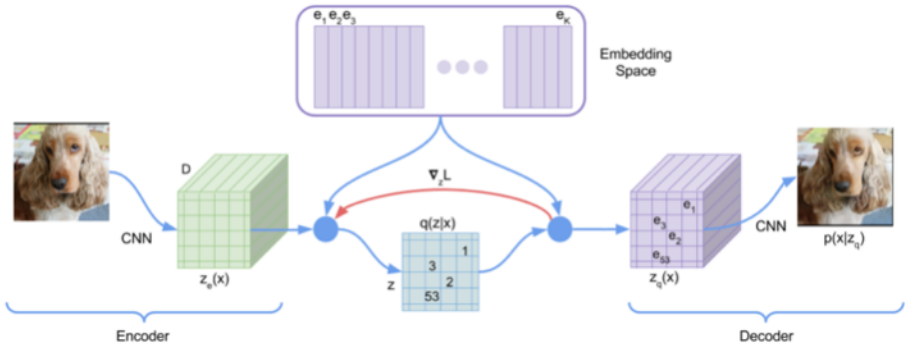
Zero-Shot Text-to-Image Generation

OpenAI Feb 2021

Aditya Ramesh [1] Mikhail Pavlov [1] Gabriel Goh [1] Scott Gray [1]
Chelsea Voss [1] Alec Radford [1] Mark Chen [1] Ilya Sutskever [1]

Step 1:

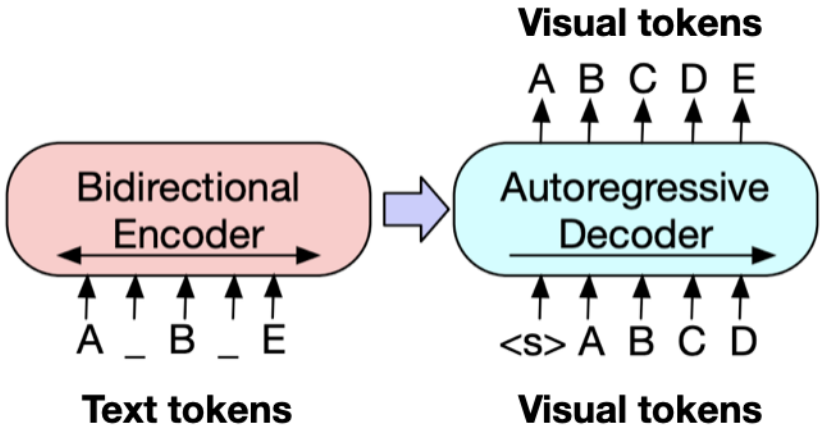
Learn Discrete Dictionary of Visual Tokens



VQVAE — Oord, Vinyals, Kavukcuoglu, 2017
VQGAN — Esser, Rombach, Ommer, 2021
dVAE - DALL-E — Ramesh et al 2021

Step 2:

Build a scene as a composition of discrete visual tokens

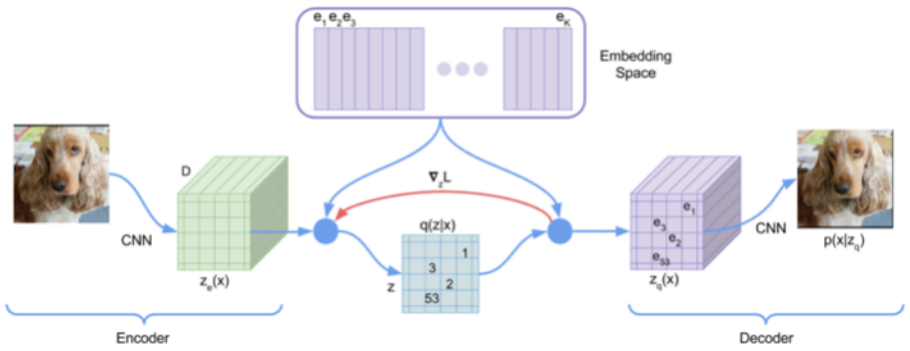


BART, GPT-3, etc

DALL-E (v1)

Step 1:

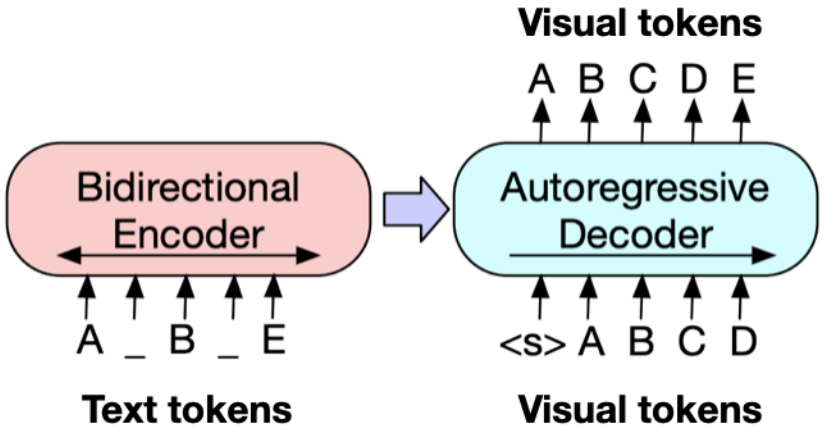
Learn Discrete Dictionary of Visual Tokens



VQVAE — Oord, Vinyals, Kavukcuoglu, 2017
VQGAN — Esser, Rombach, Ommer, 2021
dVAE - DALL-E — Ramesh et al 2021

Step 2:

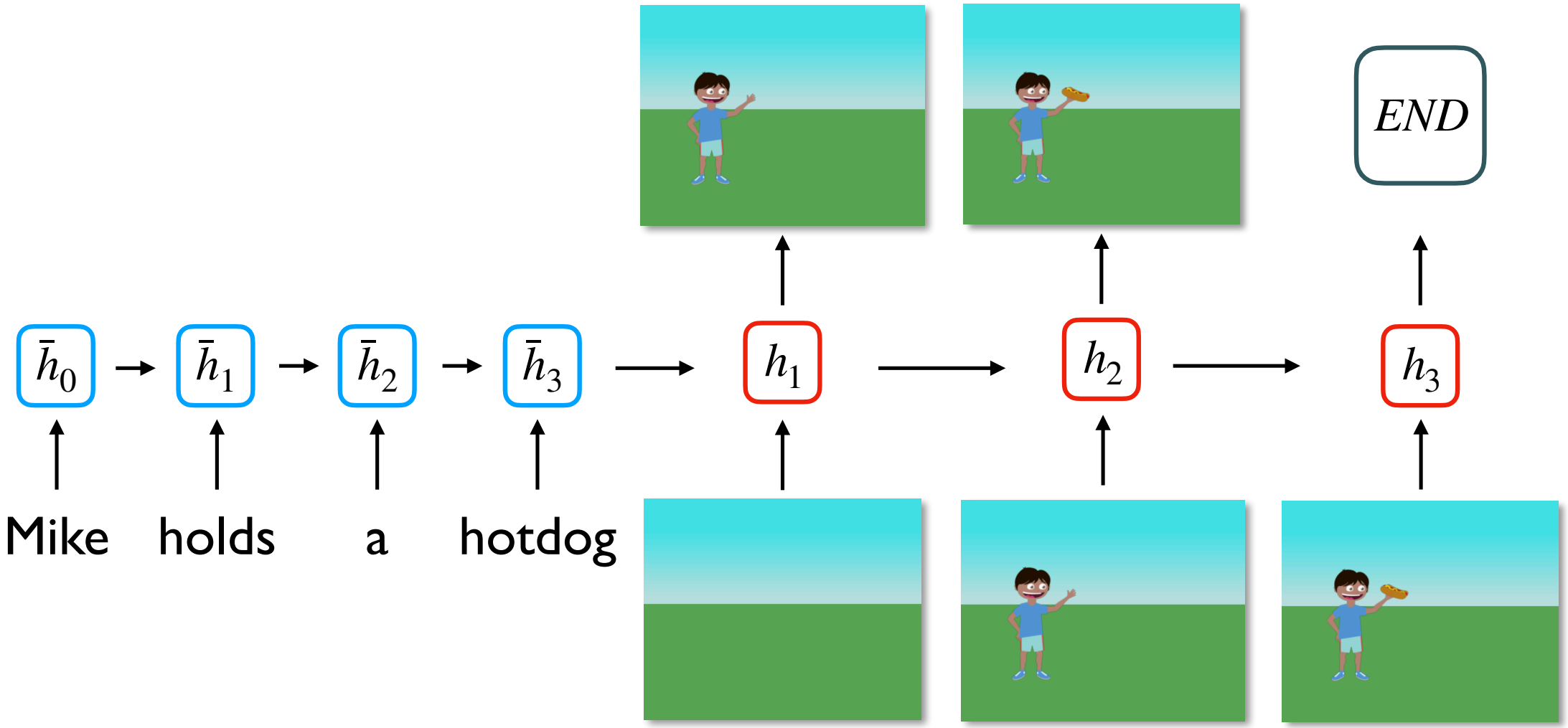
Build a scene as a composition of discrete visual tokens



BART, GPT-3, etc

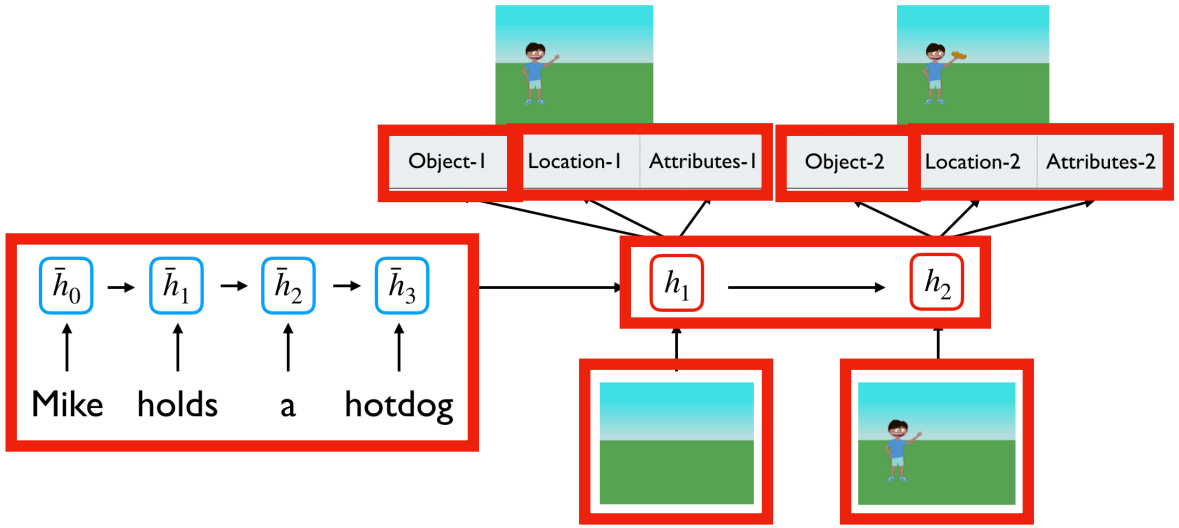
an armchair in the shape of an avocado. . . .

Text to Scene as Machine Translation!



Text2Scene: Generating Compositional Scenes from Textual Descriptions
Fuwen Tan, Song Feng, Vicente Ordonez. Intl. Conference on Computer Vision and Pattern Recognition. **CVPR 2019.**
Long Beach, California. June 2019.(~Oral presentation + Best Paper Finalist -- top 1% of submissions)

The actual model



locations

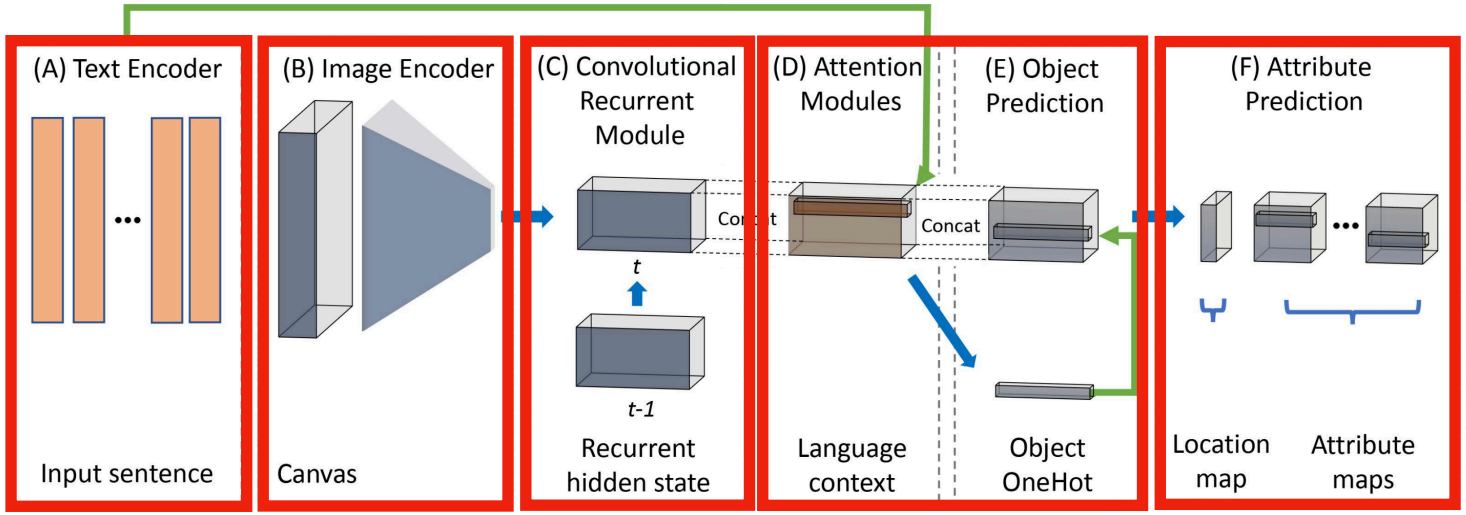
objects

Objective

$$L = -w_o \sum_t \log p(o_t) - w_l \sum_t \log p(l_t)$$
$$- \sum_k w_k \sum_t \log p(R_t^k) +$$
$$+ w_a^O L_{attn}^O + w_a^A L_{attn}^A,$$

attributes

$$h_i^E = \text{BiGRU}(x_i, h_{i-1}^E, h_{i+1}^E) \Omega(B_i \; h_t^D = \text{ConvGRU}(\Omega(. \; p(o_t)) \propto \Theta^o([u_t^o; o_{t-} \; p(l_t, \{R_t^k\}) = \Theta^a([u_t^a; o_t; c_t^a])$$



Encourage attention weights to fully use the input text.

$$L_{attn} = \sum_i [1 - \sum_t \alpha_{t,i}]^2$$

uvavision / Text2Scene

Watch 6 | Star 26 | Fork 6

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Security | Insights | Settings

[CVPR'19] Text2Scene: Generating Compositional Scenes from Textual Descriptions

Edit

Manage topics

4 commits | 1 branch | 0 releases | 1 contributor

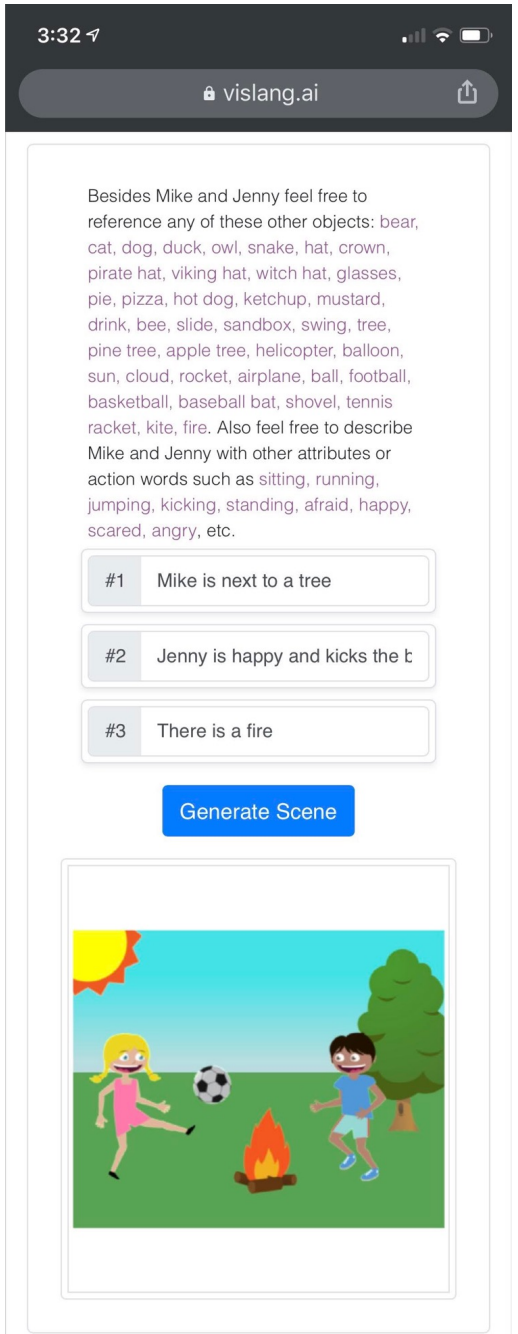
Branch: master | New pull request | Create new file | Upload files | Find File | Clone or download

fwtan Update README.md — Latest commit 5681f67 4 days ago

data	cleaning up the codes, alpha version	19 days ago
examples	cleaning up the codes, alpha version	19 days ago
experiments/scripts	cleaning up the codes, alpha version	19 days ago
lib	cleaning up the codes, alpha version	19 days ago
tools	cleaning up the codes, alpha version	19 days ago
README.md	Update README.md	4 days ago

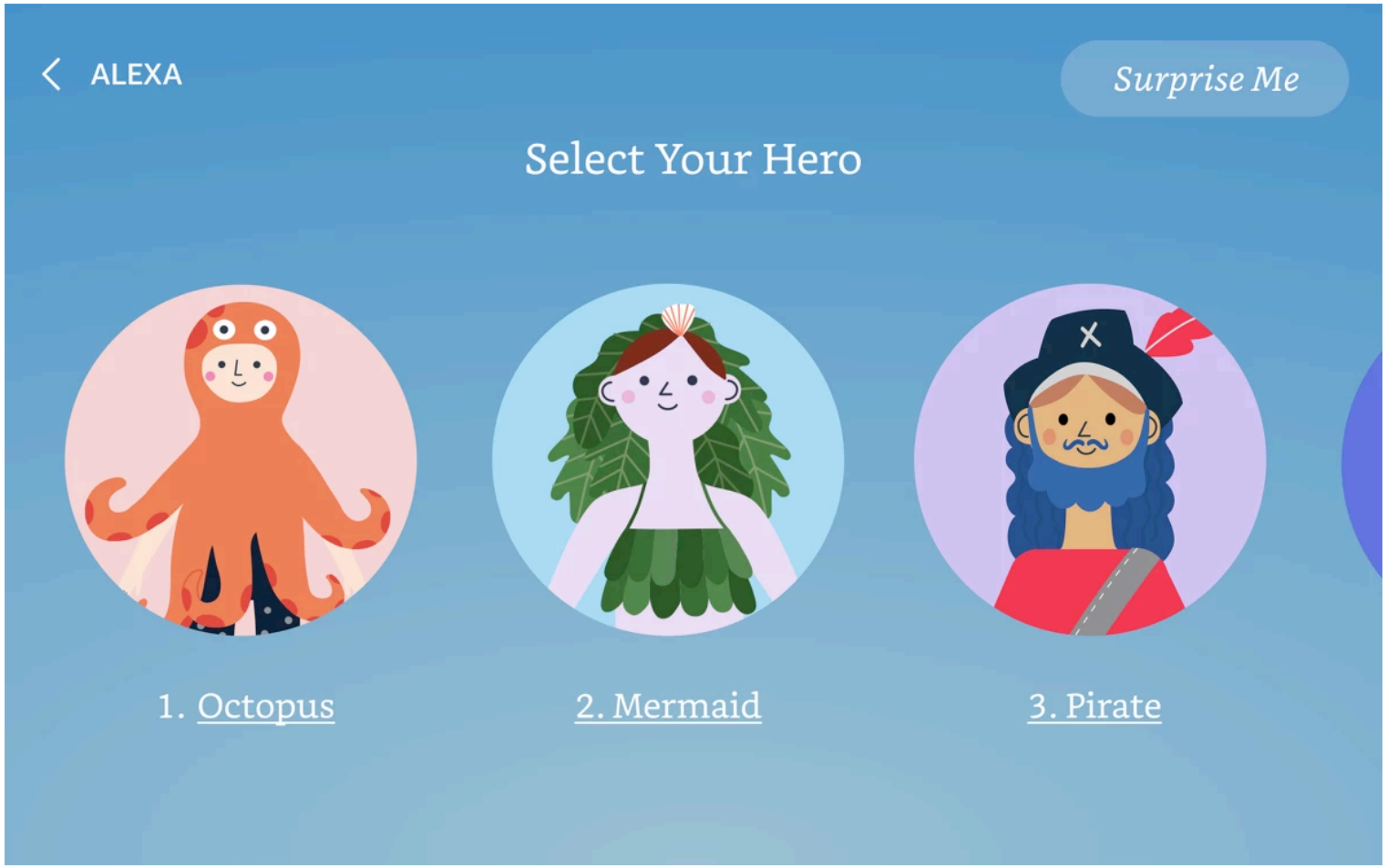
README.md

Text2Scene: Generating Compositional Scenes from Textual Descriptions



https://www.vislang.ai/text2scene

Amazon Alexa AI



https://www.amazon.science/blog/the-science-behind-alexas-new-interactive-story-creation-experience

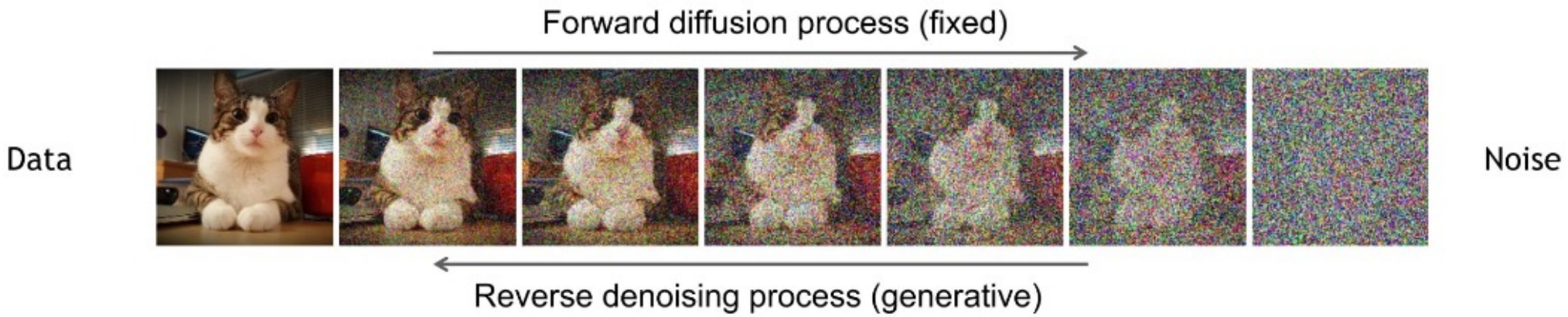
Amazon Alexa AI

https://www.amazon.science/blog/the-science-behind-alexas-new-interactive-story-creation-experience

Denoising Diffusion Probabilistic Models (DDPM)

Forward diffusion: Markov chain of diffusion steps to slowly add gaussian noise to data

Reverse diffusion: A model is trained to generate data from noise by iterative denoising



Denoising Diffusion Probabilistic Models

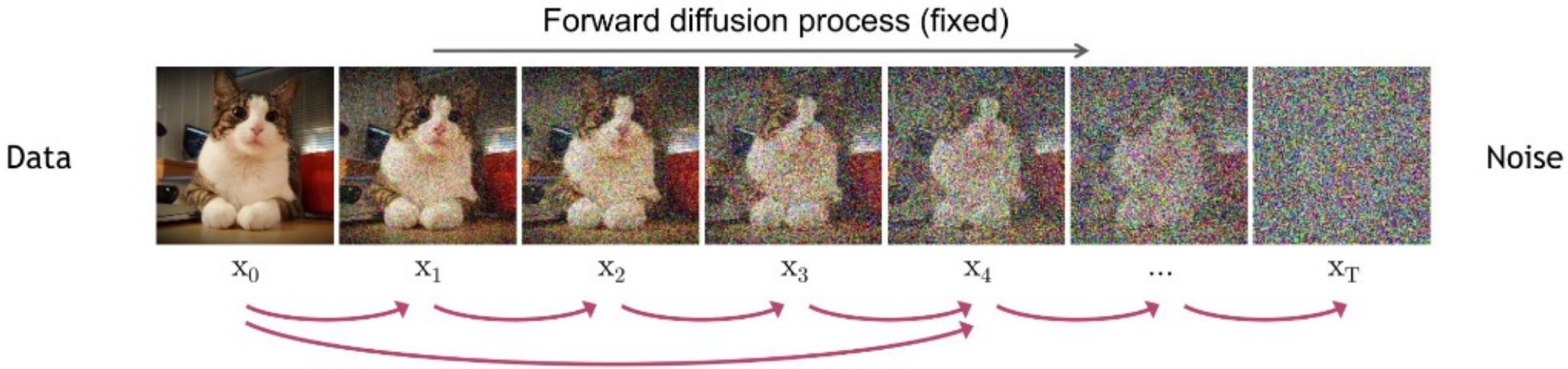
Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

https://cvpr2022-tutorial-diffusion-models.github.io/
Slides compiled by my student Aman Shrivastava

DDPM | Forward diffusion



We add a small amount of gaussian noise to a sample $\mathbf{x}_0$ in $\mathbf{T}$ timesteps to produces noised samples, $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T\}$. The steps are controlled by the noise schedule as follows:

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

https://cvpr2022-tutorial-diffusion-models.github.io/

Forward diffusion process (fixed)

Data

Noise

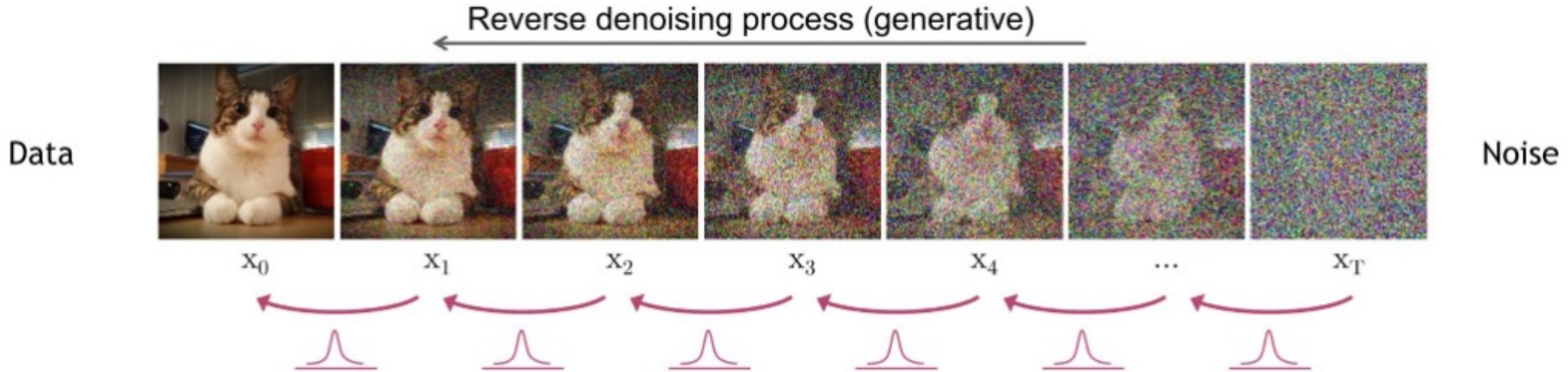
x_0 x_1 x_2 x_3 x_4 ... x_T

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

Define $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ ➡ $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$ (Diffusion Kernel)

For sampling: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

DDPM | Reverse Diffusion

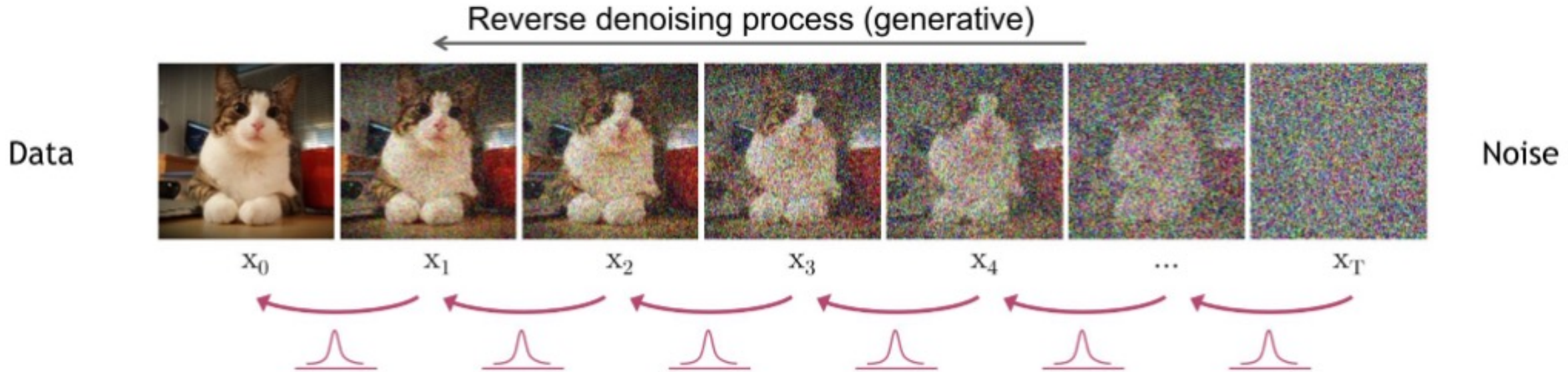


We learn a neural network model **(p_θ)** to approximate these conditional probabilities **$q(x_{(t-1)} \mid x_t)$** in order to run the reverse diffusion process as follows:

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

https://cvpr2022-tutorial-diffusion-models.github.io/

DDPM | Reverse Diffusion



We learn a neural network model **(p_θ)** to approximate these conditional probabilities **$q(x_{(t-1)} \mid x_t)$** in order to run the reverse diffusion process as follows:

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

https://cvpr2022-tutorial-diffusion-models.github.io/

U-Net

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$$

Objective:

$$L_{t-1} = D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))$$

$$= \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}||\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)||^2\right] + C$$

Given that: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$$

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

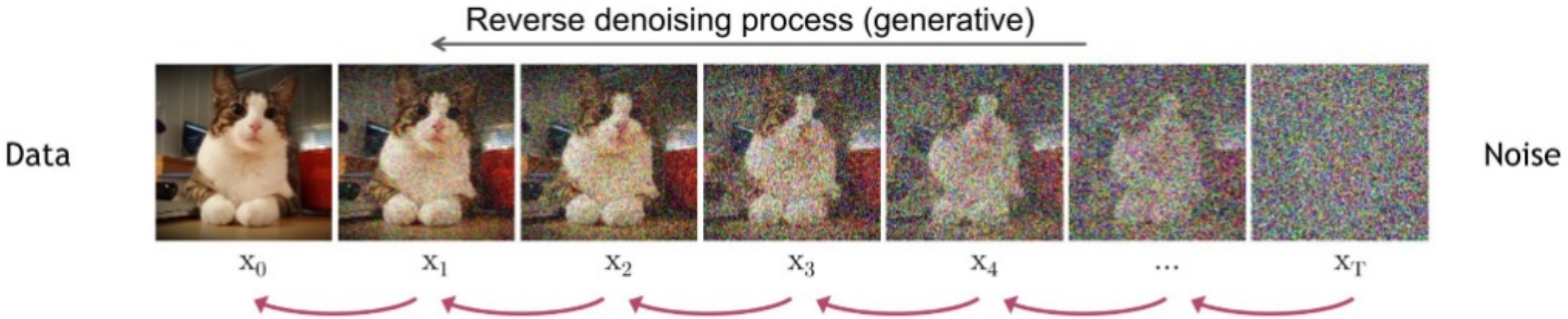
Replacing on the equation

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \Bigg[\underbrace{\frac{\beta_t^2}{2\sigma_t^2(1-\beta_t)(1-\bar{\alpha}_t)}}_{\lambda_t} ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\ \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\ \epsilon, t)||^2 \Bigg]$$

Used in practice:

$$L_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t \sim \mathcal{U}(1,T)} \left[||\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\ \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\ \epsilon}_{\mathbf{x}_t}, t)||^2 \right]$$

How do we train?



Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on

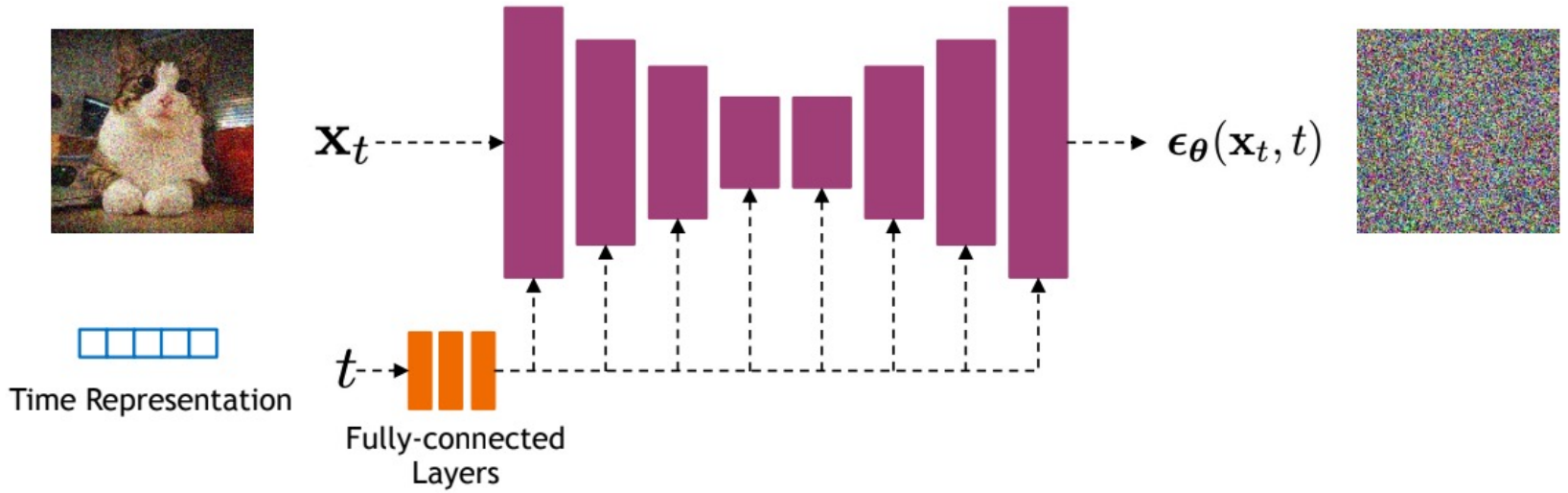
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

6: **until** converged

https://cvpr2022-tutorial-diffusion-models.github.io/

Unet to model transition

Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent $\epsilon_\theta(\mathbf{x}_t, t)$

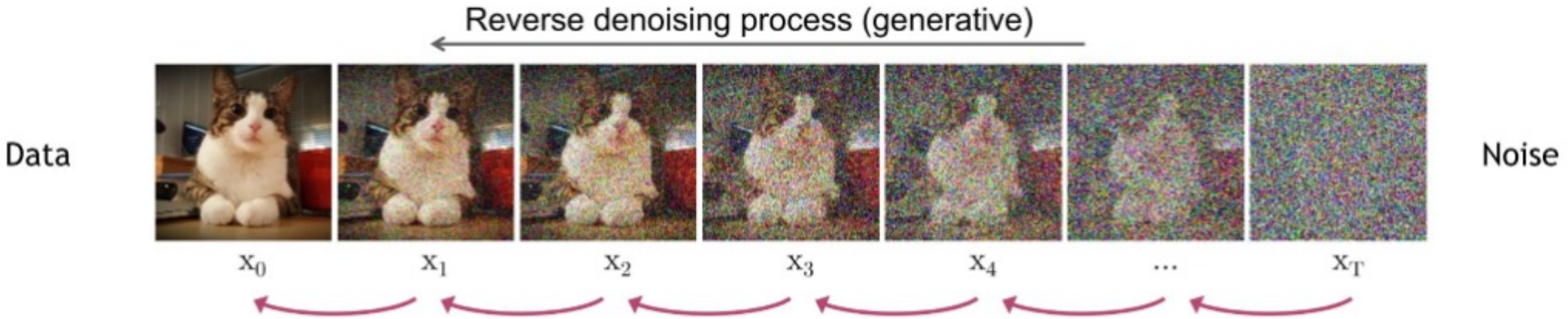


Time representation: sinusoidal positional embeddings or random Fourier features.

Time features are fed to the residual blocks using either simple spatial addition or using adaptive group normalization layers. (see Dharivwal and Nichol NeurIPS 2021)

https://cvpr2022-tutorial-diffusion-models.github.io/

How do we train?

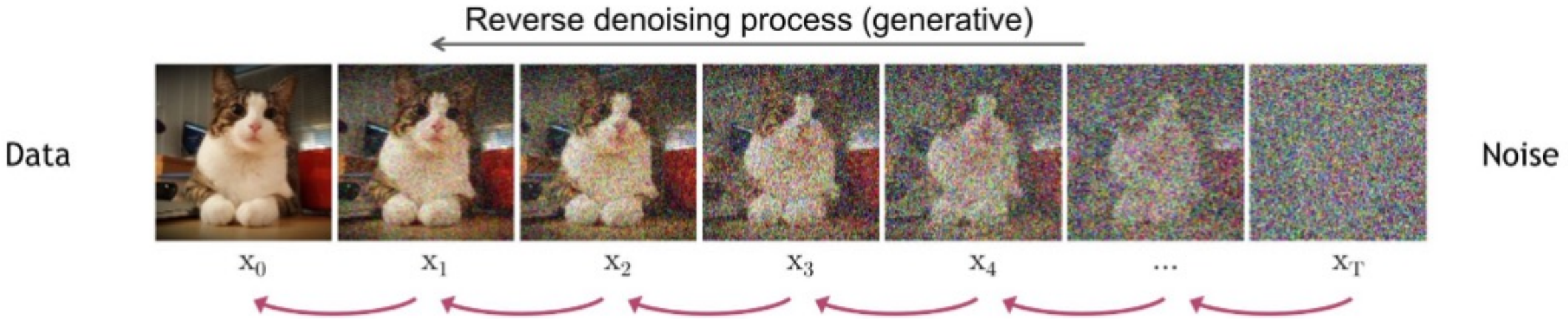


Algorithm 2 Sampling

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

https://cvpr2022-tutorial-diffusion-models.github.io/

How do we train?



Algorithm 1 Training

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

Algorithm 2 Sampling

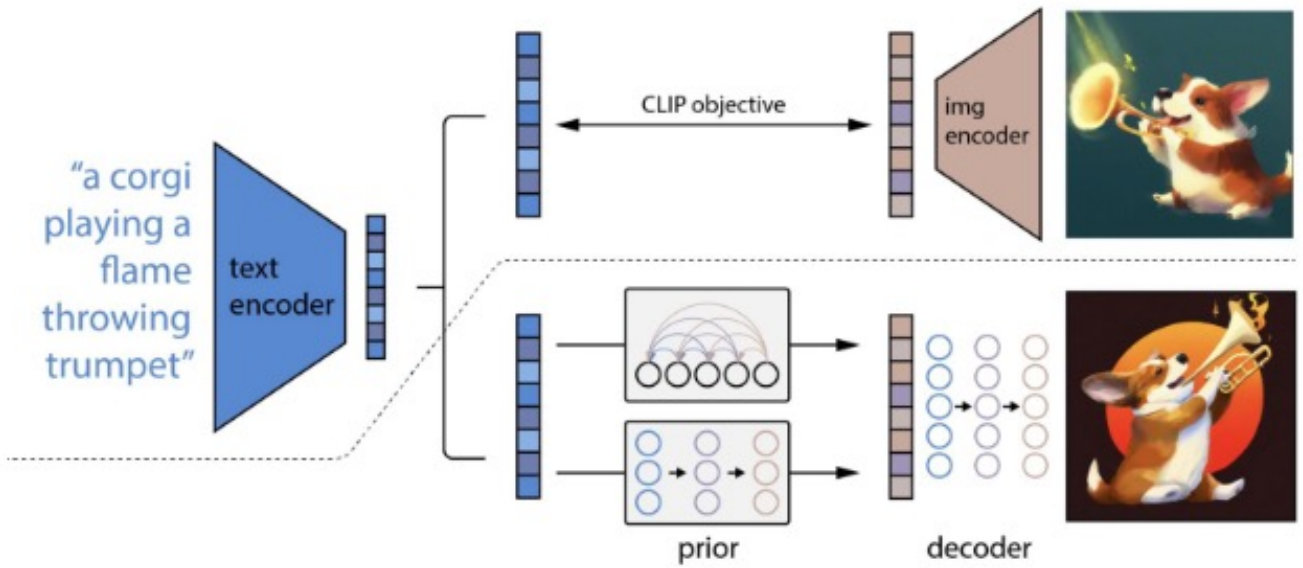
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}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

https://cvpr2022-tutorial-diffusion-models.github.io/

DALL.E 2 | Open AI

Conditioning on CLIP-embeddings

- Helps capture multimodal representations
- The bi-partite latent enables several text-controlled image manipulation tasks



Ramesh, Aditya, et al. "Hierarchical text-conditional image generation with clip latents." *arXiv preprint arXiv:2204.06125* (2022).

DALL.E 2 | OpenAI

- 1kx1k text-conditioned image generation
- Uses a **prior** to produce CLIP embeddings conditioned on the text-caption
- Uses a **decoder** to produce images conditioned on the CLIP embeddings

Slides compiled by my student Aman Shrivastava

https://cvpr2022-tutorial-diffusion-models.github.io/

Imagen by Google

A cute corgi lives in a house made out of sushi.

A cute sloth holding a small treasure chest. A bright golden glow is coming from the chest.

Imagen by Google

2.2 Diffusion models and classifier-free guidance

Here we give a brief introduction to diffusion models; a precise description is in Appendix A. Diffusion models [63, 28, 65] are a class of generative models that convert Gaussian noise into samples from a learned data distribution via an iterative denoising process. These models can be conditional, for example on class labels, text, or low-resolution images [e.g. 16, 29, 59, 58, 75, 41, 54]. A diffusion model $\hat{\mathbf{x}}_\theta$ is trained on a denoising objective of the form

$$\mathbb{E}_{\mathbf{x},\mathbf{c},\boldsymbol{\epsilon},t}\left[w_t \|\hat{\mathbf{x}}_\theta(\alpha_t \mathbf{x} + \sigma_t \boldsymbol{\epsilon}, \mathbf{c}) - \mathbf{x}\|_2^2\right] \tag{1}$$

where $(\mathbf{x}, \mathbf{c})$ are data-conditioning pairs, $t \sim \mathcal{U}([0,1])$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and α_t, σ_t, w_t are functions of t that influence sample quality. Intuitively, $\hat{\mathbf{x}}_\theta$ is trained to denoise $\mathbf{z}_t := \alpha_t \mathbf{x} + \sigma_t \boldsymbol{\epsilon}$ into $\mathbf{x}$ using a squared error loss, weighted to emphasize certain values of t. Sampling such as the ancestral sampler [28] and DDIM [64] start from pure noise $\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and iteratively generate points $\mathbf{z}_{t_1}, \ldots, \mathbf{z}_{t_T}$, where $1 = t_1 > \cdots > t_T = 0$, that gradually decrease in noise content. These points are functions of the $\mathbf{x}$-predictions $\hat{\mathbf{x}}_0^t := \hat{\mathbf{x}}_\theta(\mathbf{z}_t, \mathbf{c})$.

https://arxiv.org/pdf/2205.11487.pdf

Now

- Stable Diffusion: Diffusion in the Latent Space
- DALLE-3: Larger Model, Larger Resolution, Recaptioned Data
- SORA: Text-to-Video Generation
- Stable Diffusion Video
- Other modalities: Audio, Audio+Video

Questions