Deep Learning for Vision & Language

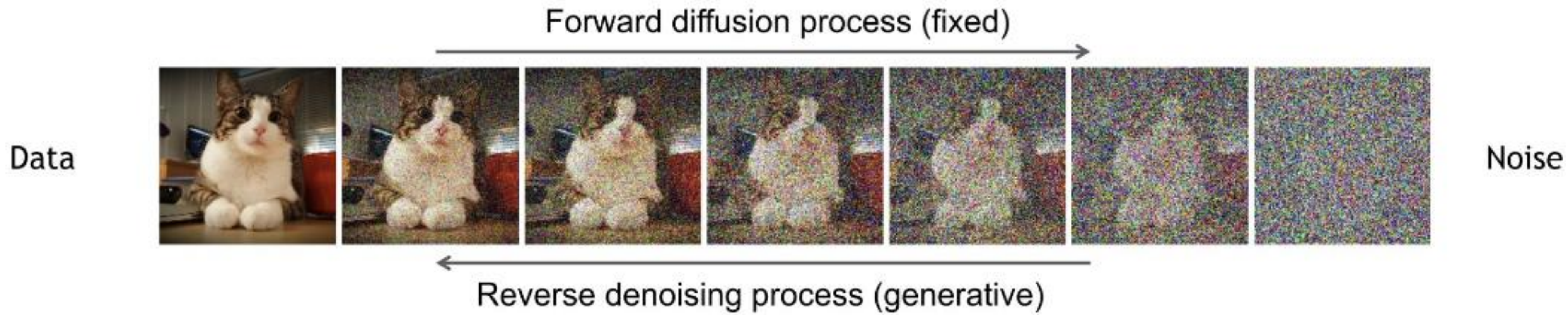
Segmentation, AutoEncoders, Variational AutoEncoders, Introduction to Diffusion Models

RICE UNIVERSITY

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

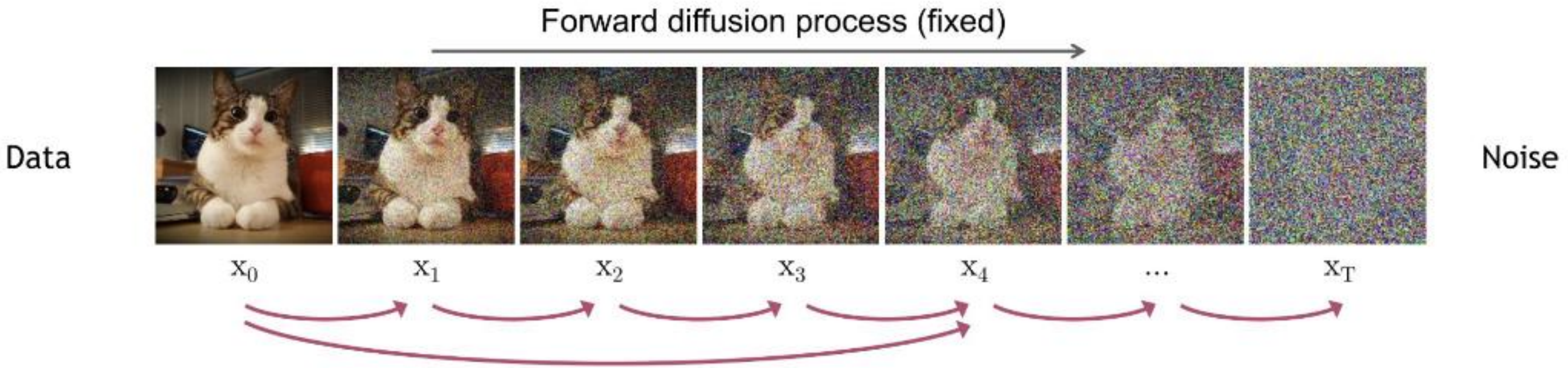
Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

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

Slides compiled by my student Aman Shrivastava

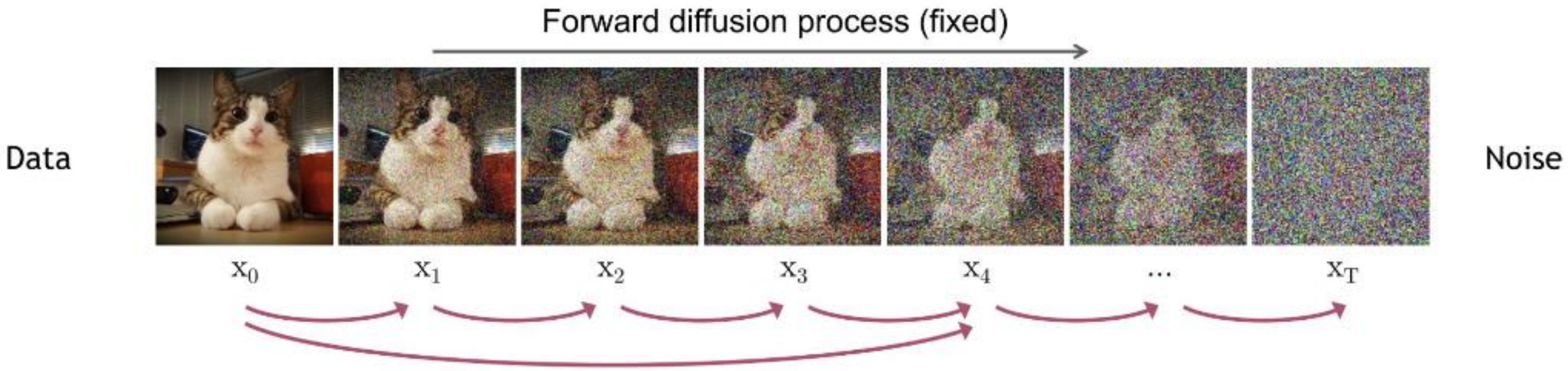
DDPM | Forward diffusion



We add a small amount of gaussian noise to a sample $\mathbf{x_0}$ in **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/

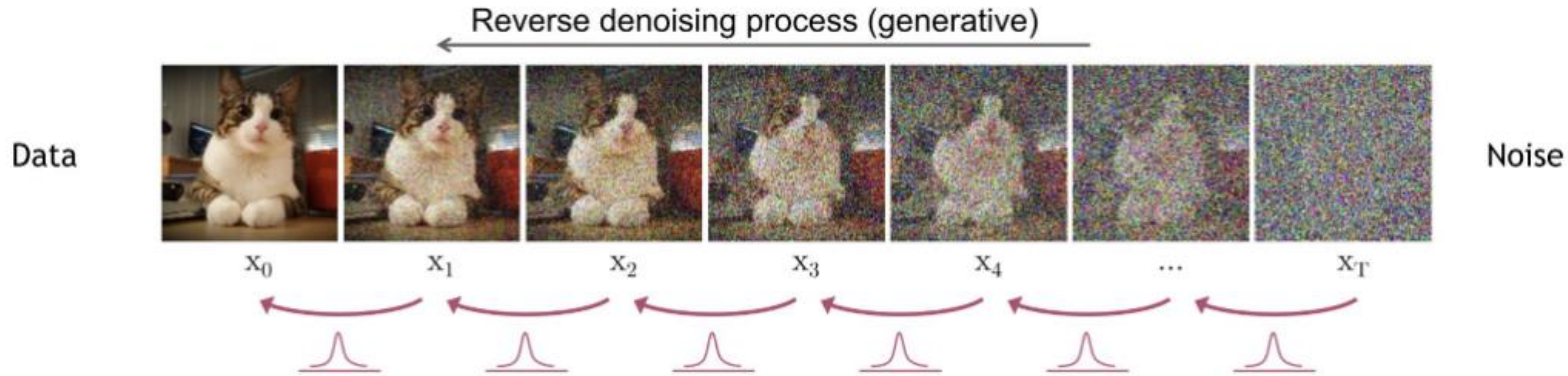


$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

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

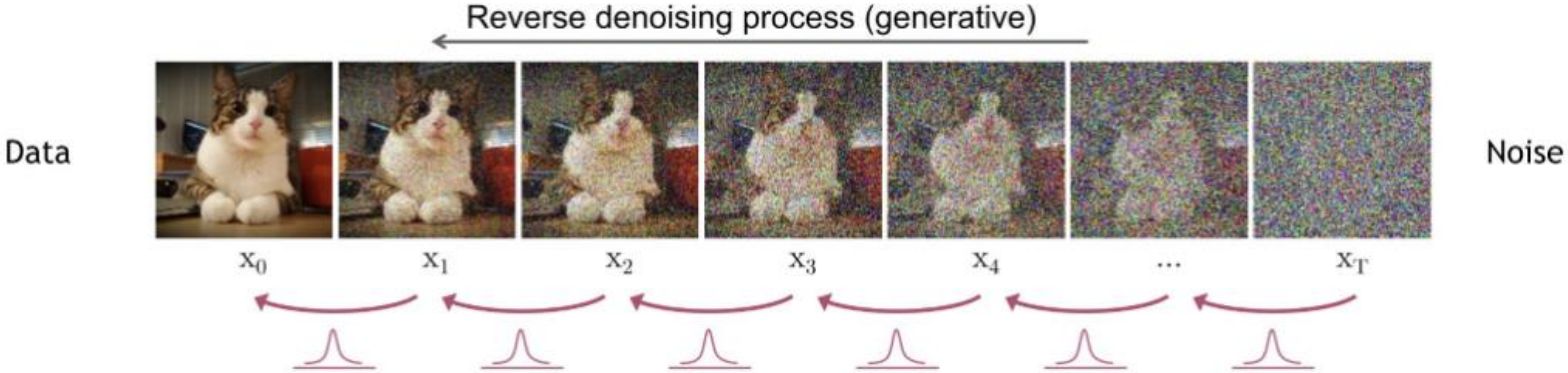


We learn a neural network model **(p_θ)** to approximate these conditional probabilities **$q(x_{(t-1)} \mid x_t)$** in order to run the reverse diffusion process as follows:

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

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

DDPM | Reverse Diffusion

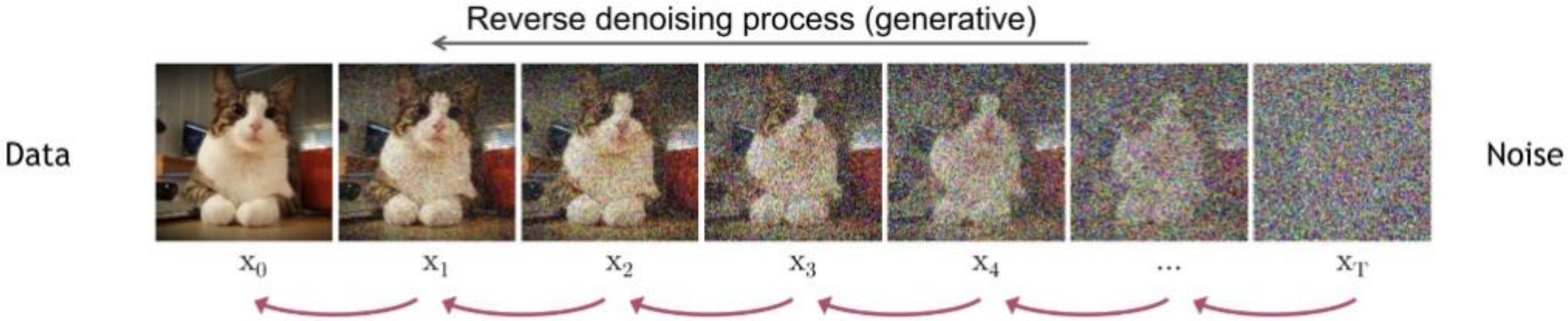


We learn a neural network model **(p_θ)** to approximate these conditional probabilities **$q(x_{(t-1)} | x_t)$** in order to run the reverse diffusion process as follows:

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

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

How do we train?



Algorithm 1 Training

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

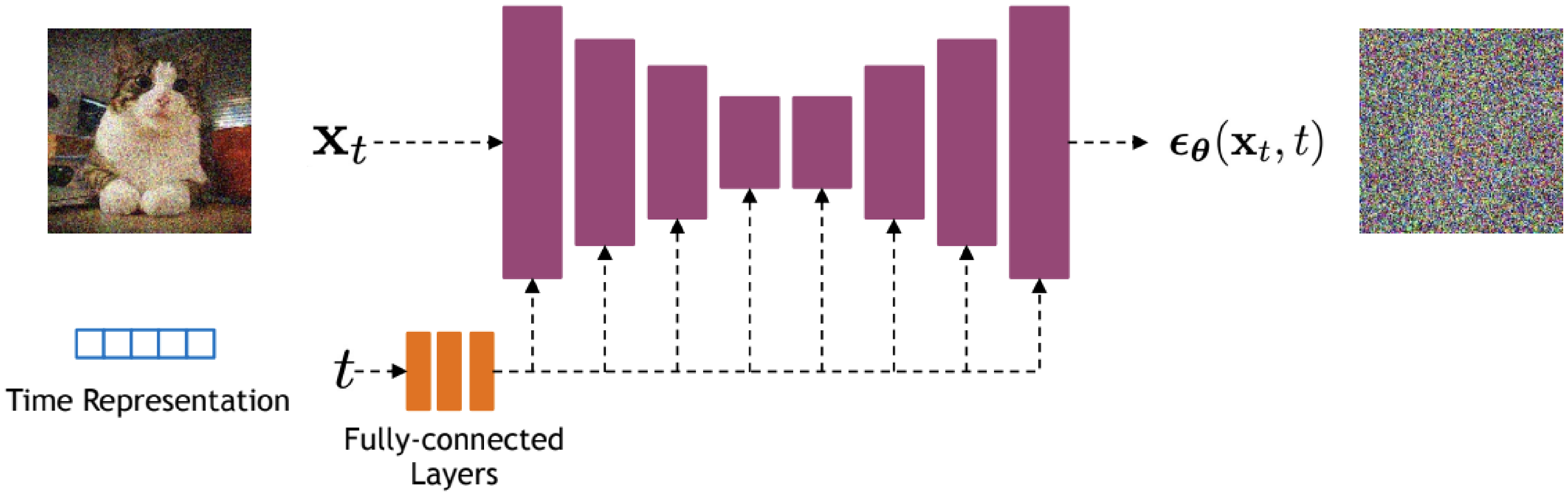
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

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 $\epsilon_\theta(\mathbf{x}_t, t)$

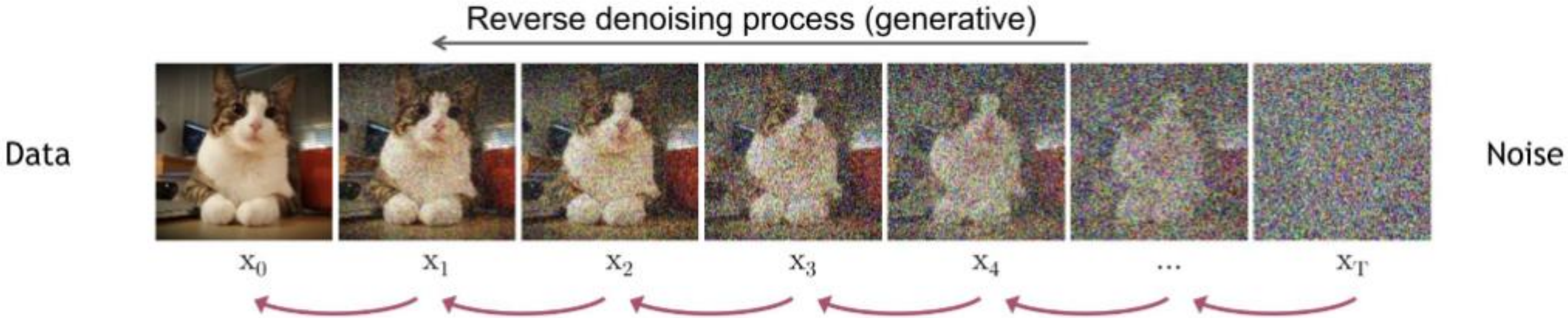


Time representation: sinusoidal positional embeddings or random Fourier features.

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?

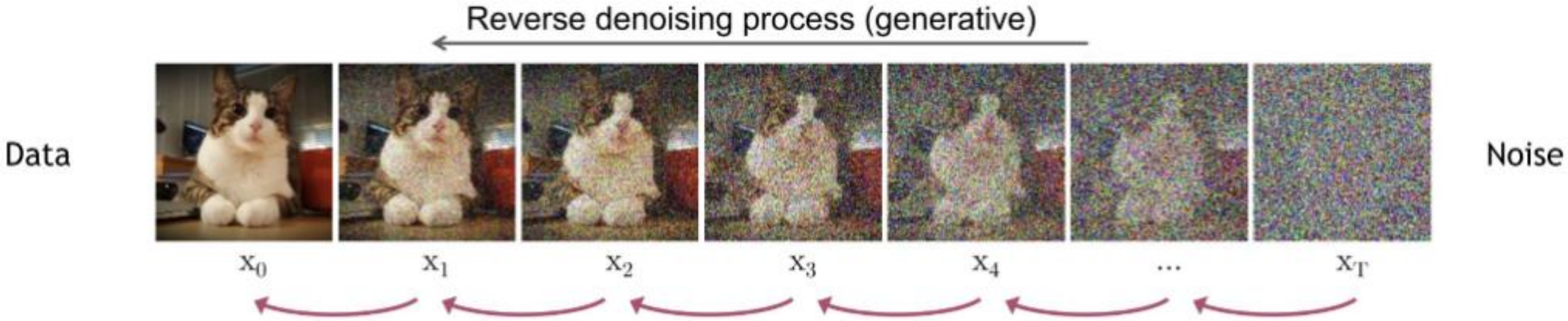


Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **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 $\mathbf{x}_0$

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

How do we train?



Algorithm 1 Training

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

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

Latent Diffusion Models (Stable Diffusion)

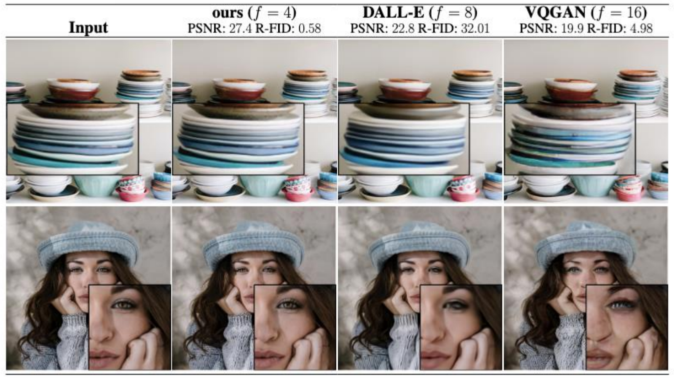
High-Resolution Image Synthesis with Latent Diffusion Models

Robin Rombach[1] * Andreas Blattmann[1] * Dominik Lorenz[1] Patrick Esser[R] Björn Ommer[1]
[1]Ludwig Maximilian University of Munich & IWR, Heidelberg University, Germany [R]Runway ML
https://github.com/CompVis/latent-diffusion

13 Apr 2022

Abstract

By decomposing the image formation process into a sequential application of denoising autoencoders, diffusion models (DMs) achieve state-of-the-art synthesis results on image data and beyond. Additionally, their formulation allows for a guiding mechanism to control the image generation process without retraining. However, since these models typically operate directly in pixel space, optimization of powerful DMs often consumes hundreds of GPU days and inference is expensive due to sequential evaluations. To enable DM training on limited computational



Latent Diffusion Models

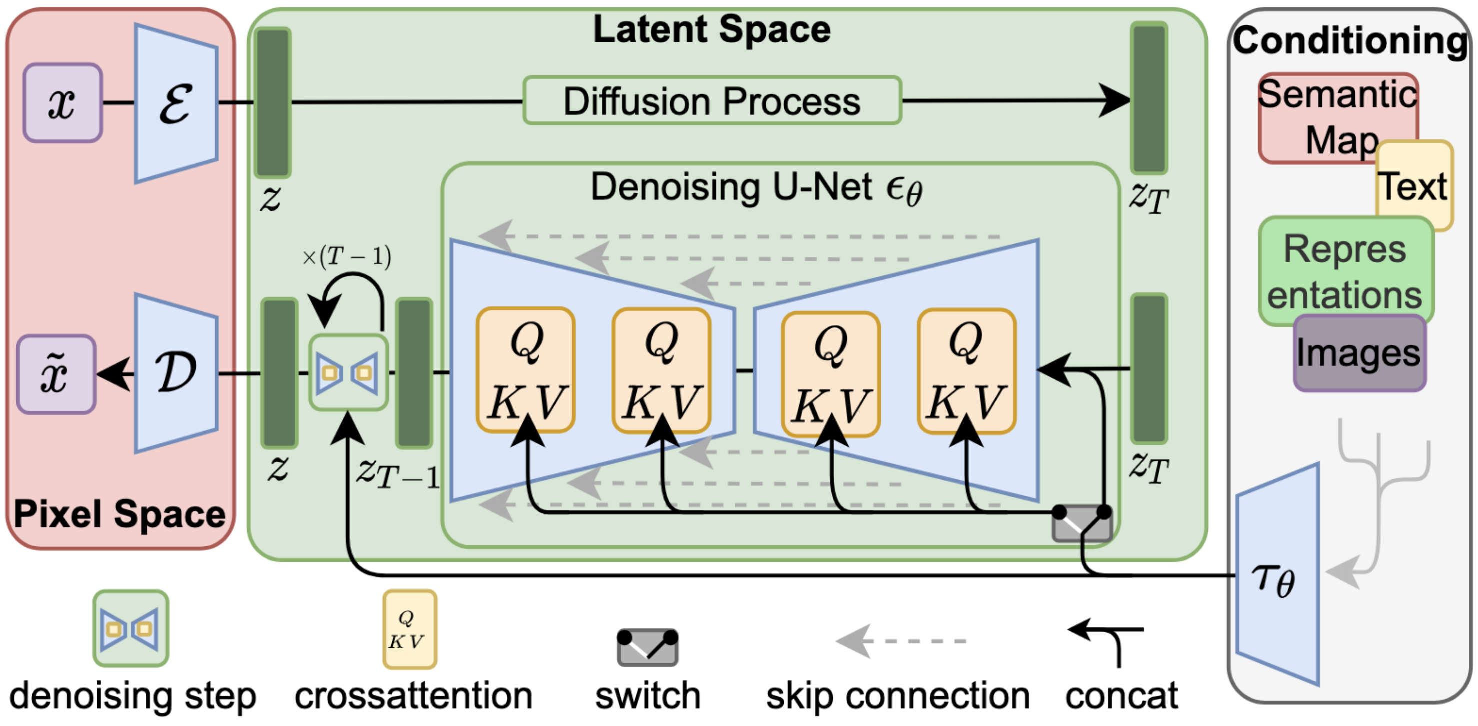
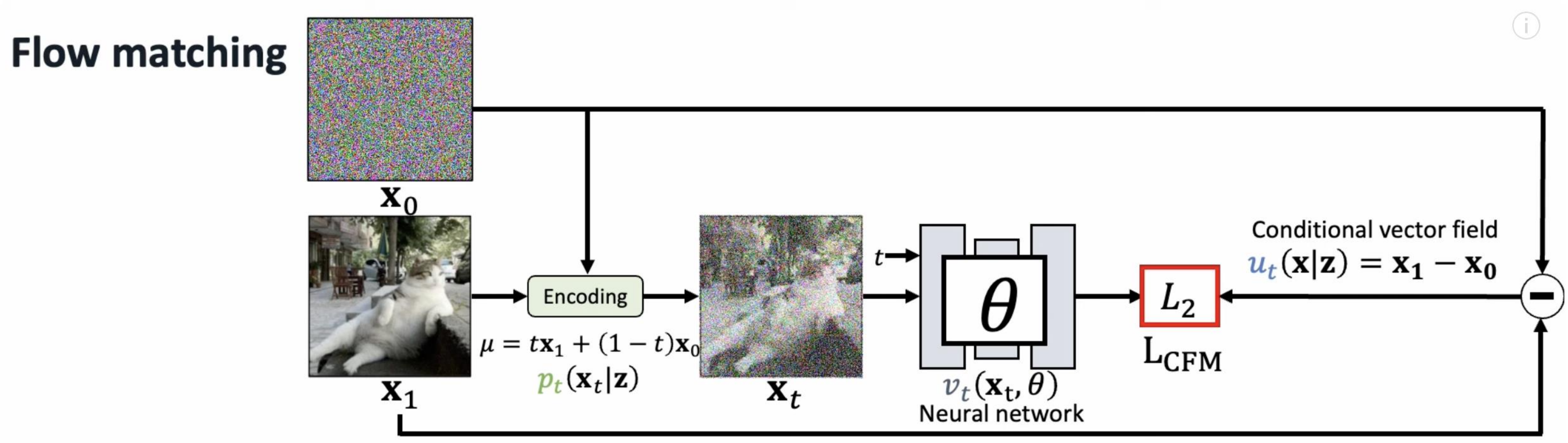


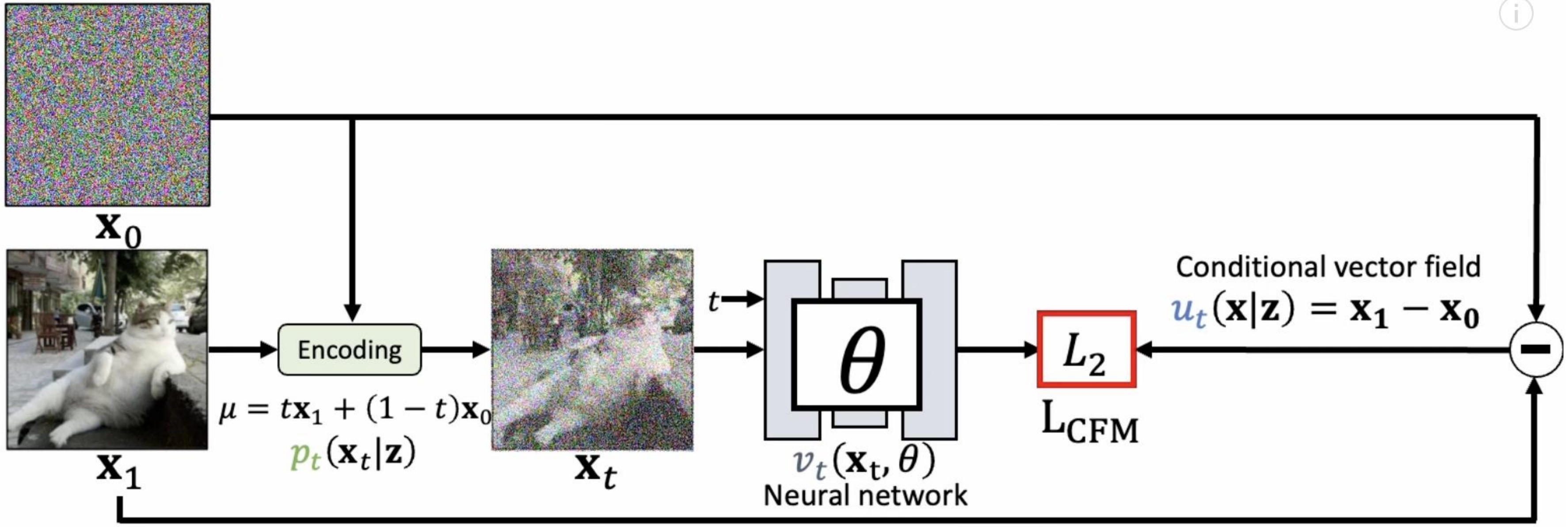
Figure 3. We condition LDMs either via concatenation or by a more general cross-attention mechanism. See Sec. 3.3

Another More Recent Improvement: Flow Matching

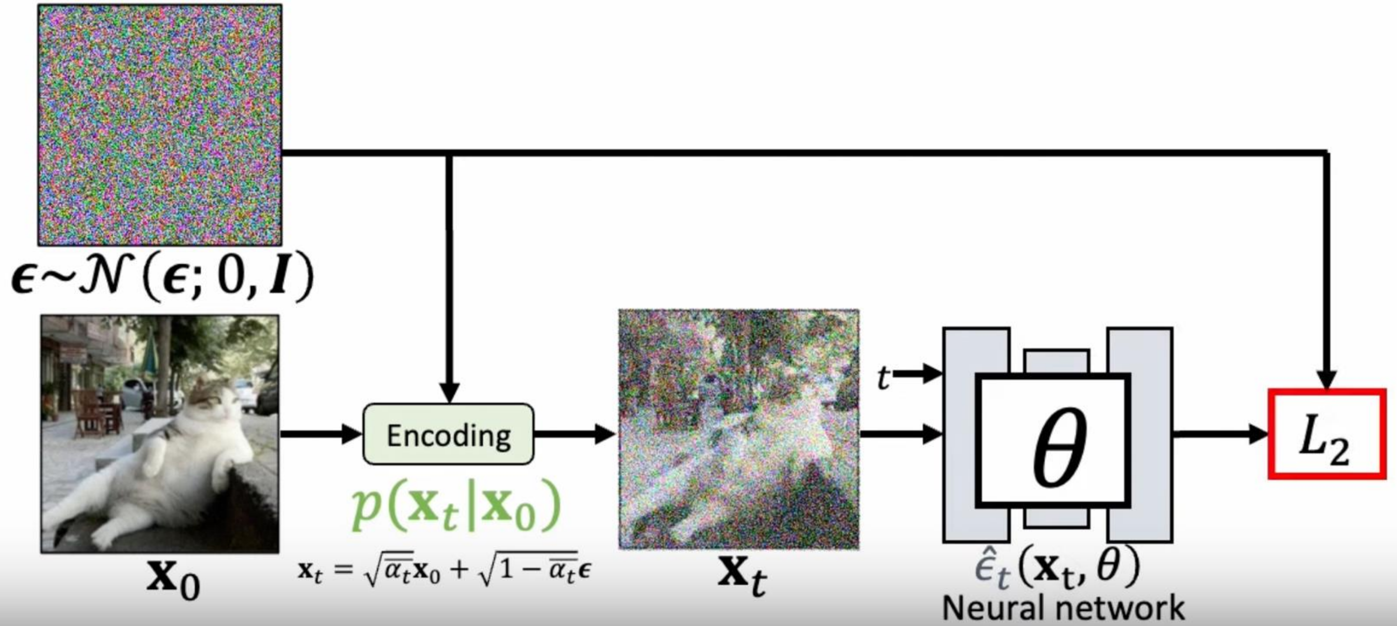


See video by Jia-bin Huang (U of Maryland) https://www.youtube.com/watch?v=DDq_pIfHqLs

Flow matching



Diffusion



AV-Link: Temporally-Aligned Diffusion Features for Cross-Modal Audio-Video Generation

Moayed Haji-Ali[1,2,*] Willi Menapace[2] Aliaksandr Siarohin[2] Ivan Skorokhodov[2]
Alper Canberk[2] Kwot Sin Lee[2] Vicente Ordonez[1] Sergey Tulyakov[2]

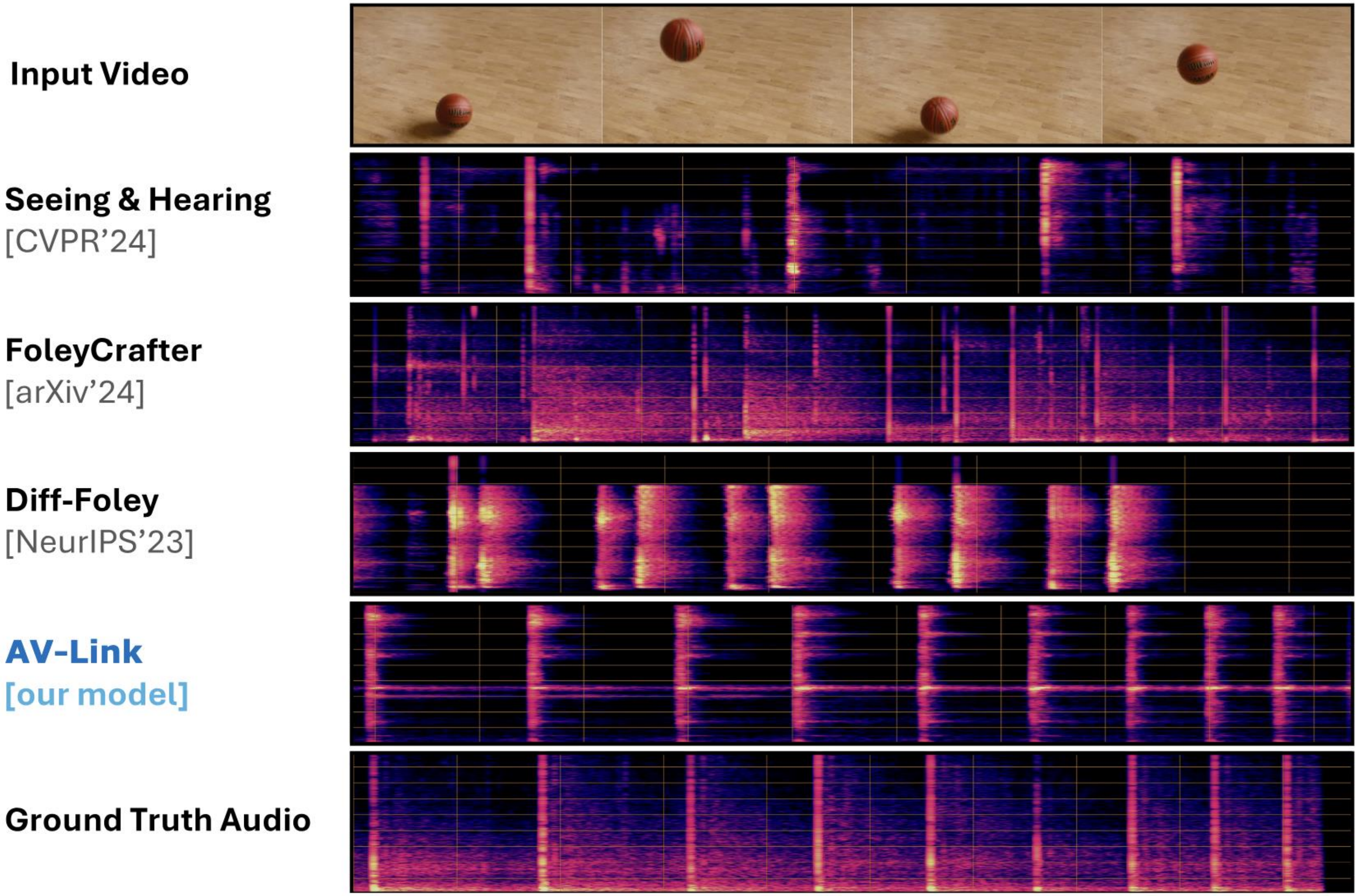
[1]Rice University [2]Snap Inc

Project Webpage: https://snap-research.github.io/AVLink

https://arxiv.org/pdf/2412.15191

AV-Link: Temporally-Aligned Diffusion Features for Cross-Modal Audio-Video Generation

Moaye
okhodov[2]
kov[2]



3.1. Background

We base our generative models on the Flow Matching framework [47, 50]. Flow Matching expresses generation of data $\mathbf{X}_1 \sim p_d$ as the progressive transformation of $\mathbf{X}_0$ following a path connecting samples from the two distributions. In its simplest formulation [50], the path is instantiated as a linear interpolation between the samples:

$$\mathbf{X}_t = t\mathbf{X}_1 + (1-t)\mathbf{X}_0, \tag{1}$$

and $\mathbf{X}_0 \sim p_n = \mathcal{N}(0, \mathbb{I})$ originate from a noise distribution.

We can move along the path following the velocity $v_t = \frac{d\mathbf{X}_t}{dt} = \mathbf{X}_1 - \mathbf{X}_0$ approximated by learnable $\mathcal{G}$ minimizing:

$$\mathcal{L} = \mathbb{E}_{t\sim p_t, \mathbf{X}_1 \sim p_d, \mathbf{X}_0 \sim p_n} \left\| \mathcal{G}(\mathbf{X}_t, t) - v_t \right\|_2^2, \tag{2}$$

where p_t indicates a training distribution over time t, which we instantiate as a logit normal distribution [17].

3.1. Background

We base our generative models on the Flow Matching framework [47, 50]. Flow Matching expresses generation of data $\mathbf{X}_1 \sim p_d$ as the progressive transformation of $\mathbf{X}_0$ following a path connecting samples from the two distributions. In its simplest formulation [50], the path is instantiated as a linear interpolation between the samples:

$$\mathbf{X}_t = t\mathbf{X}_1 + (1-t)\mathbf{X}_0, \tag{1}$$

and $\mathbf{X}_0 \sim p_n = \mathcal{N}(0, \mathbb{I})$ originate from a noise distribution.

We can move along the path following the velocity $v_t = \frac{d\mathbf{X}_t}{dt} = \mathbf{X}_1 - \mathbf{X}_0$ approximated by learnable $\mathcal{G}$ minimizing:

$$\mathcal{L} = \mathbb{E}_{t\sim p_t, \mathbf{X}_1\sim p_d, \mathbf{X}_0\sim p_n} \left\|\mathcal{G}(\mathbf{X}_t, t) - v_t\right\|_2^2, \tag{2}$$

where p_t indicates a training distribution over time t, which we instantiate as a logit normal distribution [17].

Assume we have the following:

$\mathbf{X}_0$ and $\frac{d\mathbf{X}_t}{dt} = \mathcal{G}(\mathbf{X}_t, t)$

How do we find $\mathbf{X}_t$?

3.1. Background

We base our generative models on the Flow Matching framework [47, 50]. Flow Matching expresses generation of data $\mathbf{X}_1 \sim p_d$ as the progressive transformation of $\mathbf{X}_0$ following a path connecting samples from the two distributions. In its simplest formulation [50], the path is instantiated as a linear interpolation between the samples:

$$\mathbf{X}_t = t\mathbf{X}_1 + (1-t)\mathbf{X}_0, \quad (1)$$

and $\mathbf{X}_0 \sim p_n = \mathcal{N}(0, \mathbb{I})$ originate from a noise distribution.

We can move along the path following the velocity $v_t = \frac{d\mathbf{X}_t}{dt} = \mathbf{X}_1 - \mathbf{X}_0$ approximated by learnable $\mathcal{G}$ minimizing:

$$\mathcal{L} = \mathbb{E}_{t\sim p_t, \mathbf{X}_1\sim p_d, \mathbf{X}_0\sim p_n} \left\| \mathcal{G}(\mathbf{X}_t, t) - v_t \right\|_2^2, \quad (2)$$

where p_t indicates a training distribution over time t, which we instantiate as a logit normal distribution [17].

At inference time, an ODE solver such as first-order Euler can be employed to produce samples $\mathbf{X}_1$ starting from Gaussian noise $\mathbf{X}_0$ using the model's velocity estimates.

Assume we have the following:

$\mathbf{X}_0$ and $\frac{d\mathbf{X}_t}{dt} = \mathcal{G}(\mathbf{X}_t, t)$

How do we find $\mathbf{X}_t$?

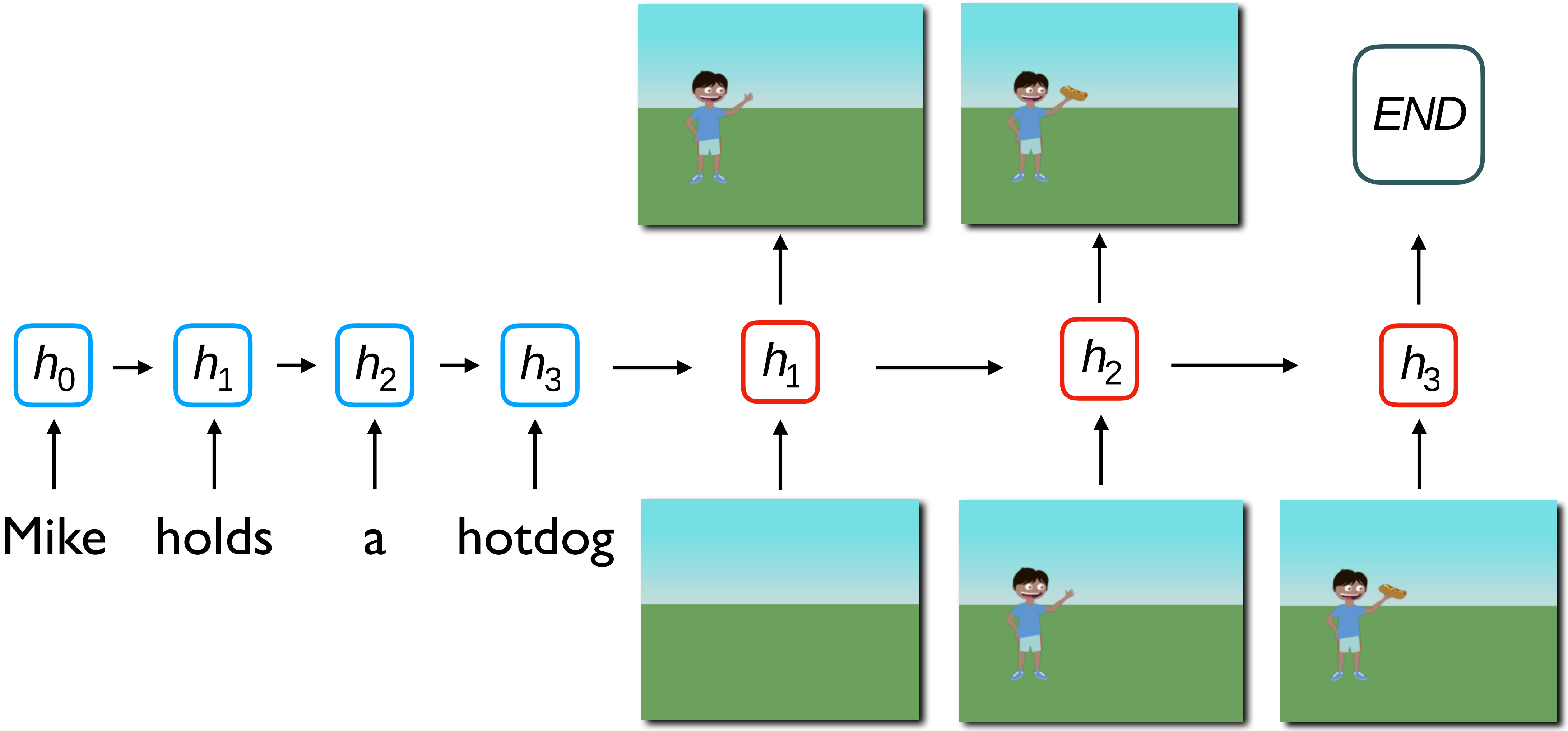
Also check this material to know more

- Entire class on Flow Matching and Diffusion Models: https://diffusion.csail.mit.edu/

Alternative Methods to Diffusion

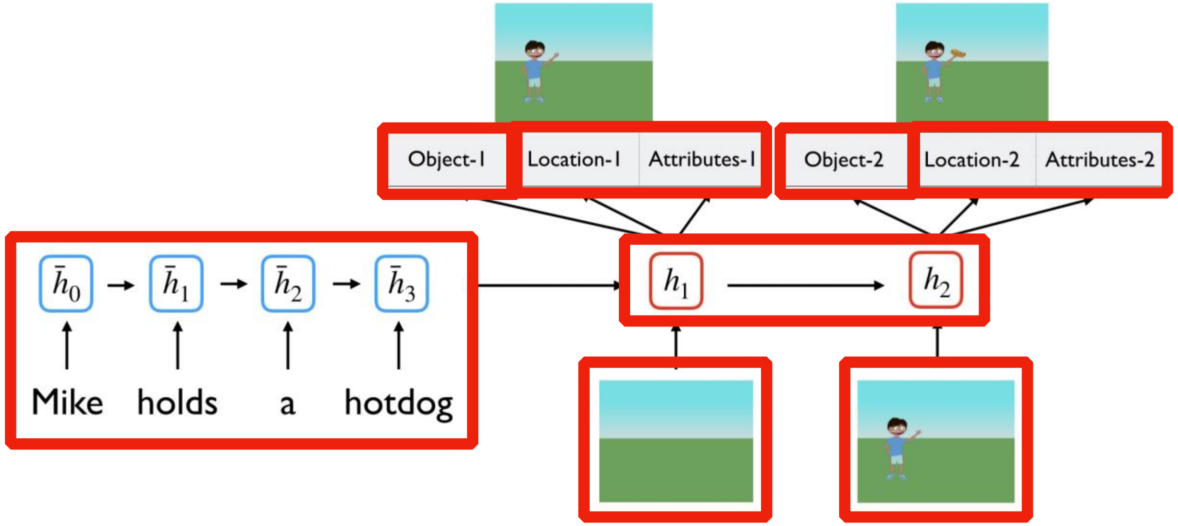
- Auto-Regressive models (LLMs to Generate Images!)

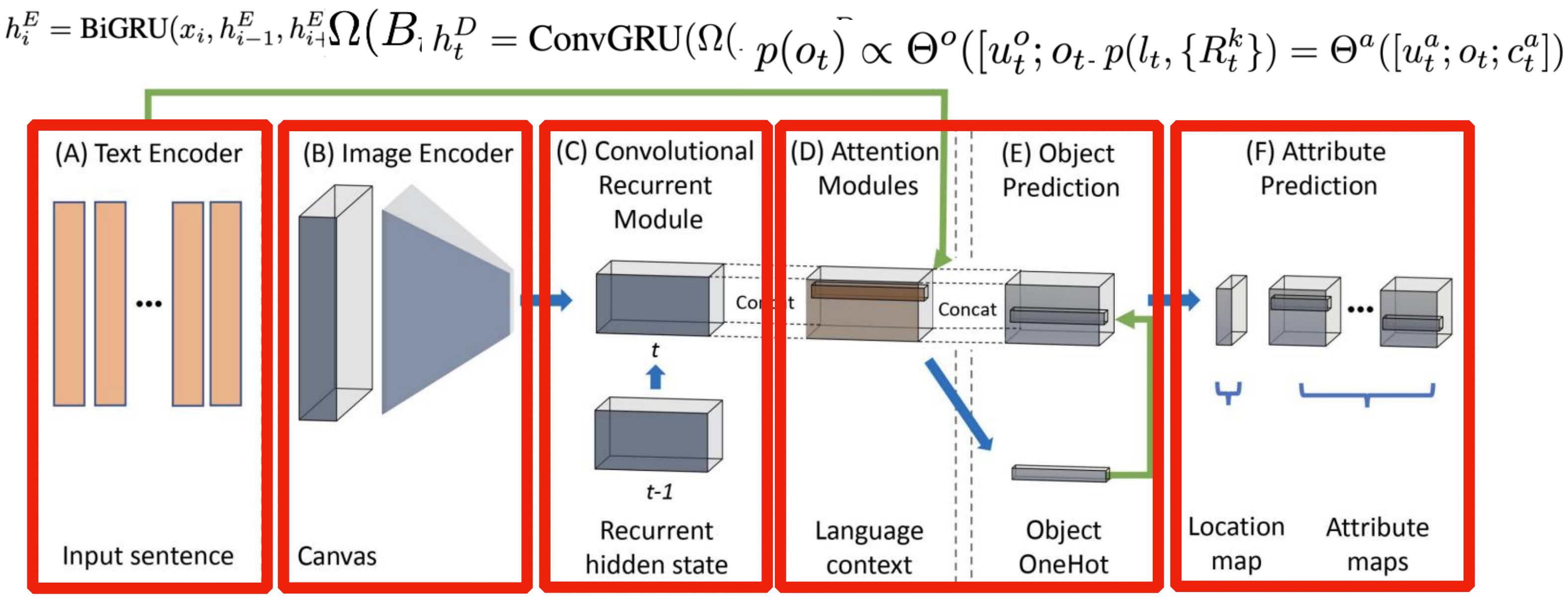
Text to Scene as Machine Translation!



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





Objective

objects

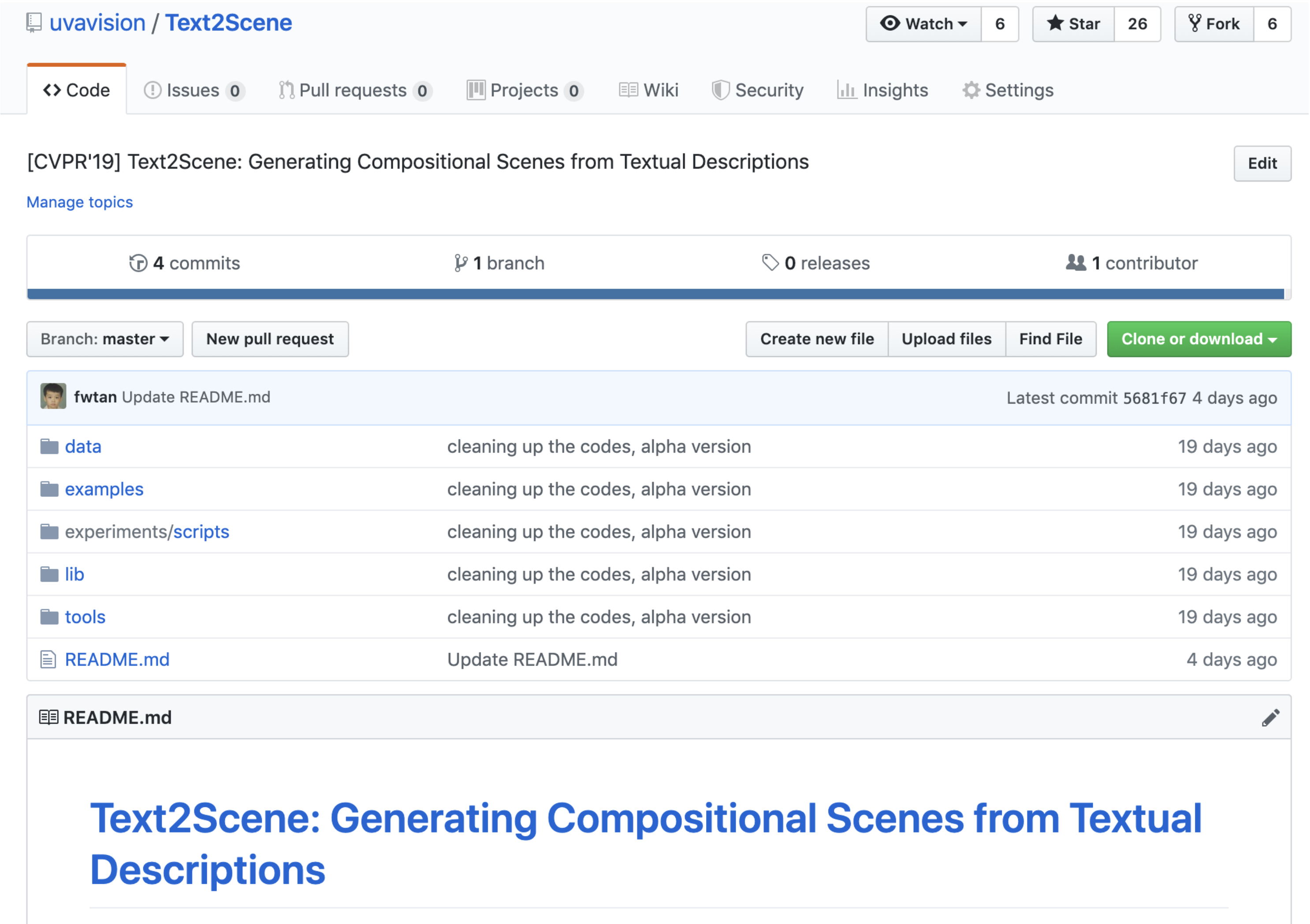
locations

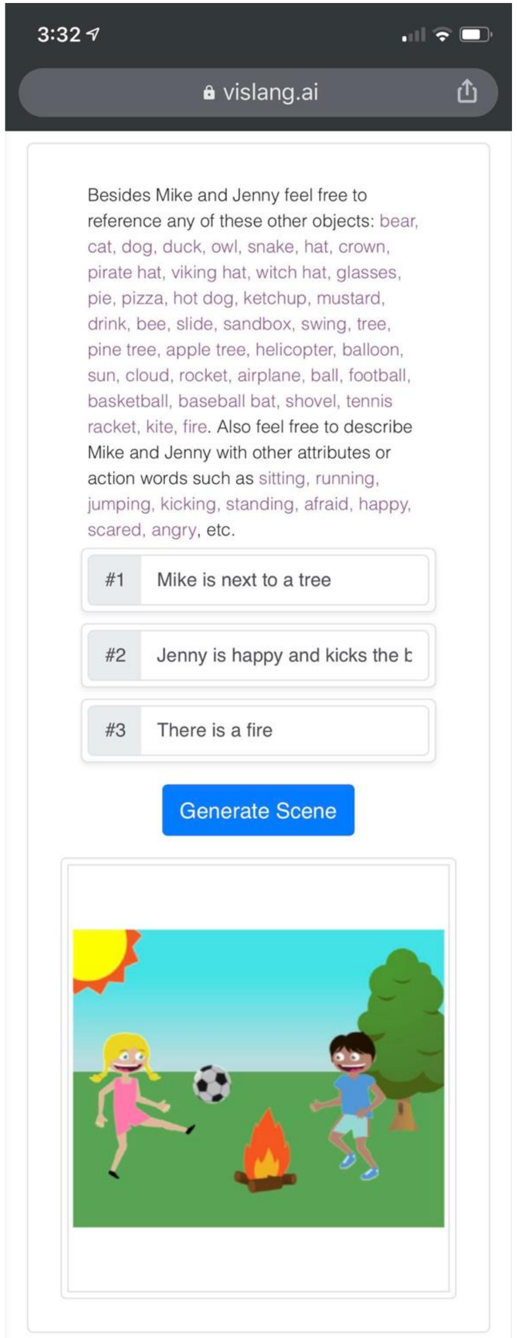
$$L = -w_o \sum_t \log p(o_t) - w_l \sum_t \log p(l_t) - \sum_k w_k \sum_t \log p(R_t^k) + w_a^O L_{attn}^O + w_a^A L_{attn}^A,$$

attributes

Encourage attention weights to fully use the input text.

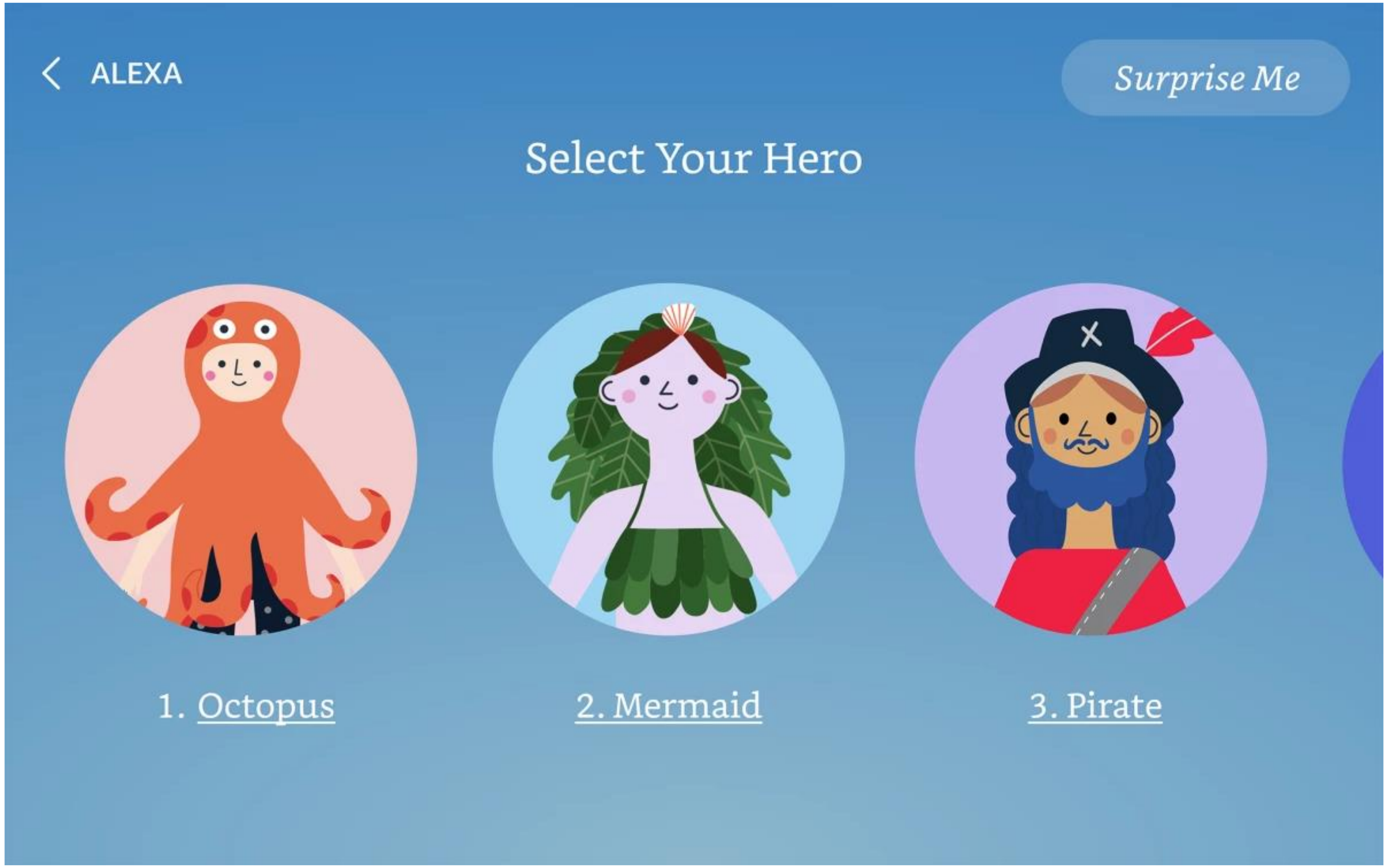
$$L_{attn} = \sum_i [1 - \sum_t \alpha_{t,i}]^2$$





https://www.vislang.ai/text2scene

Amazon Alexa AI

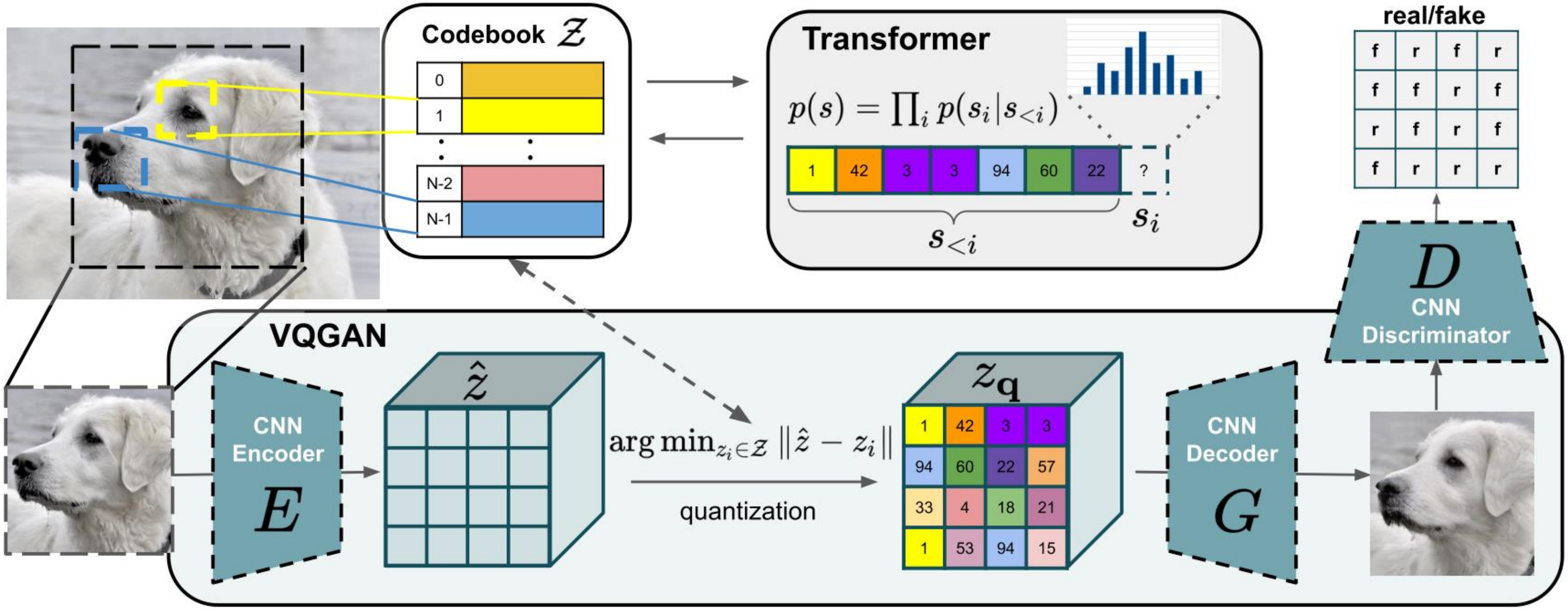


https://www.amazon.science/blog/the-science-behind-alexas-new-interactive-story-creation-experience

Amazon Alexa AI

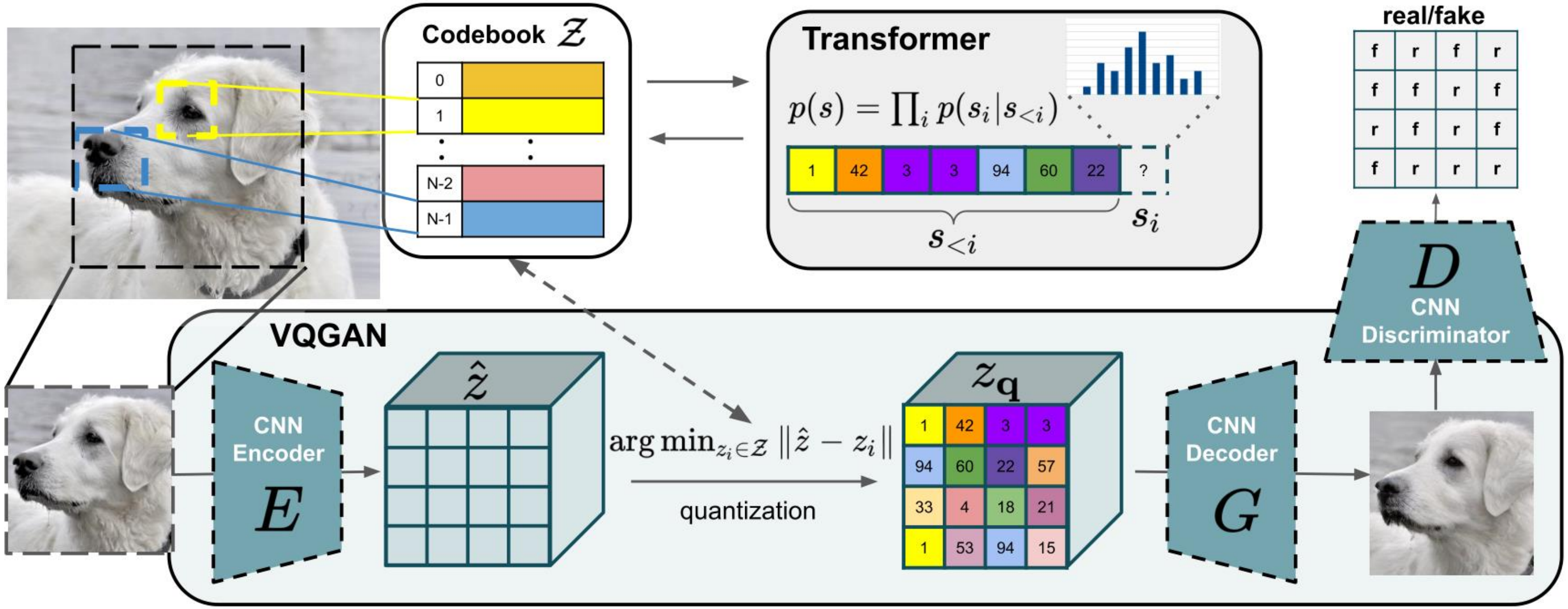
https://www.amazon.science/blog/the-science-behind-alexas-new-interactive-story-creation-experience

Vector Quantized - GAN



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

Vector Quantized GAN (VQGAN)



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

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

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

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

DALL-E (v1)

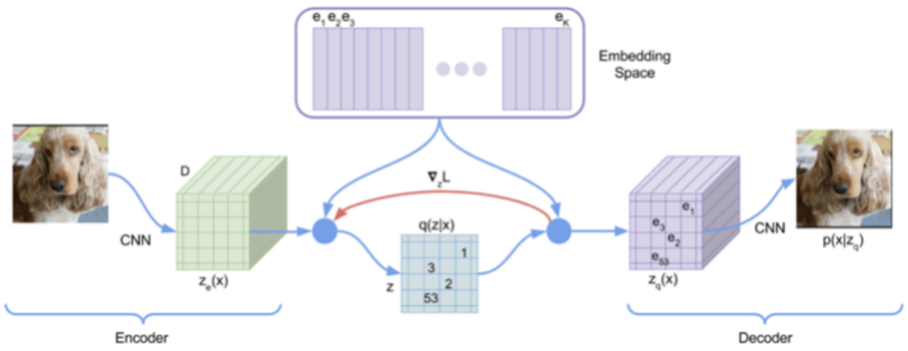
Zero-Shot Text-to-Image Generation

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

OpenAI Feb 2021

Step 1:

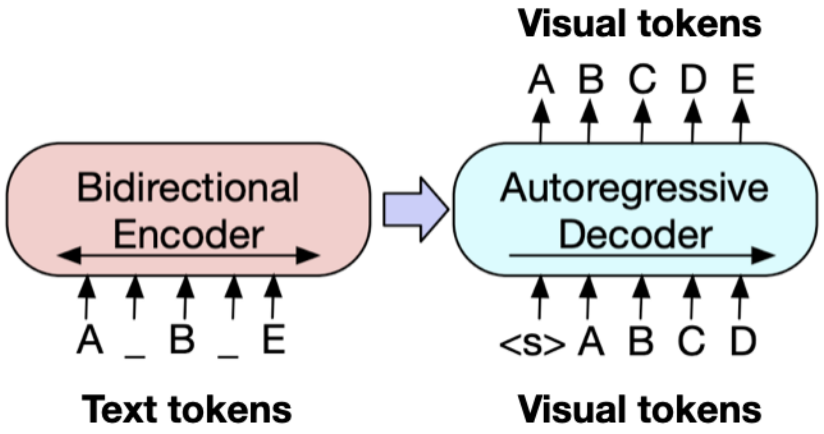
Learn Discrete Dictionary of Visual Tokens



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

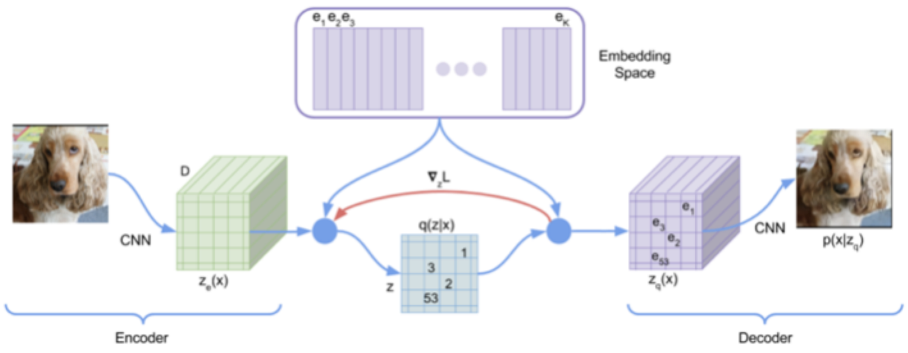


BART, GPT-3, etc

DALL-E (v1)

Step 1:

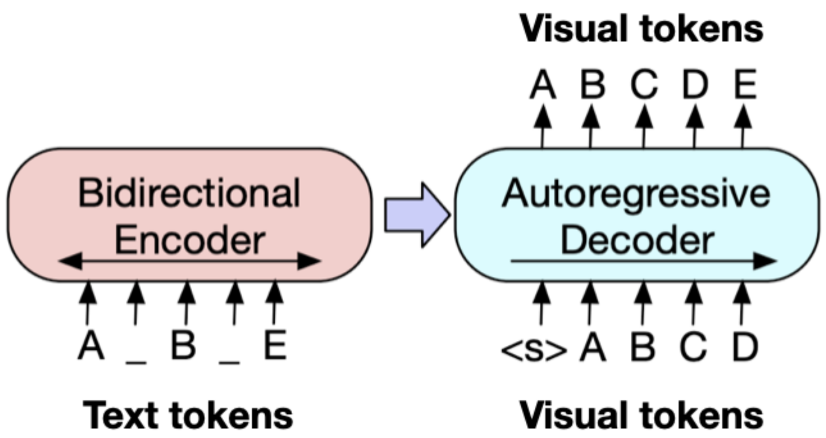
Learn Discrete Dictionary of Visual Tokens



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. . . .

Questions