10-701: Introduction to Machine Learning Lecture 26 – Generative Models

Pradeep Ravikumar & Geoff Gordon

Spring 2026

Thanks to Matt Gormley, Aran Nayebi, Henry Chai for various slides

Front Matter

- Announcements
  - HW6
    - due today Wednesday, April 22
    - late due date on Friday, April 24
  - Project
    - Video Showcase due on Wednesday, April 22
    - Final report due on Thursday, April 23
    - Project video showcase this Friday, April 24! Attendance required, refreshments will be served!
    - As a reminder, no late days can be used for project deliverables.

Image Generation

What do we mean by “image generation”?

Image Generation

sea anemone

brain coral

slug

- Given a class label, sample a new image from that class
  - Image classification takes an image and predicts its label $p(y \mid \boldsymbol{x})$
  - Class-conditional generation is doing this in reverse $p(x|y)$

- **Class-conditional generation**
- Super resolution
- Image Editing
- Style transfer
- Text-to-image (TTI) generation

Source: https://arxiv.org/pdf/1906.00446.pdf

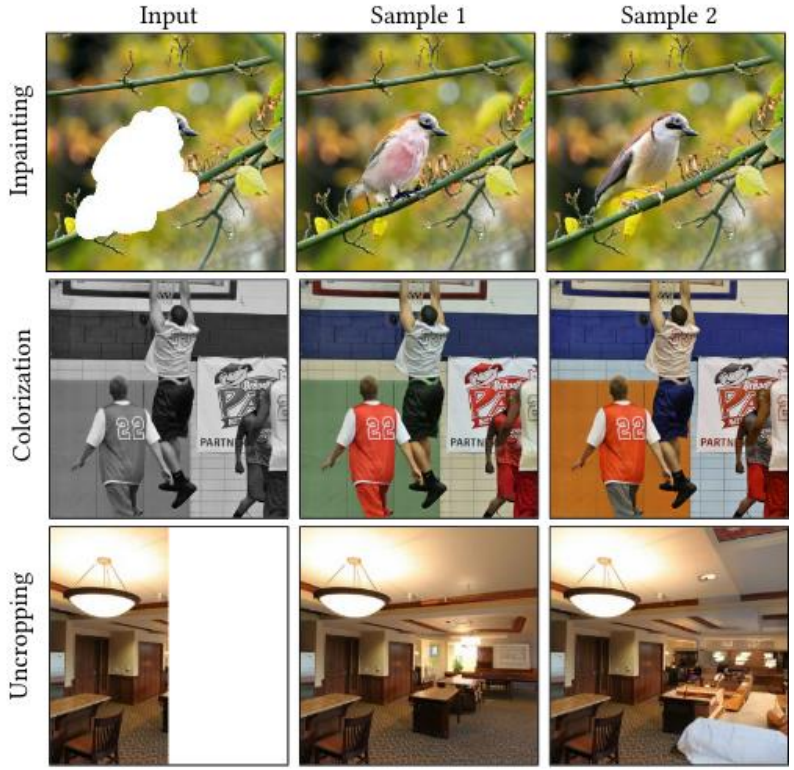
Image Generation

- Given a low-resolution image, generate a high-resolution reconstruction of the image

- Class-conditional generation
- **Super resolution**
- Image Editing
- Style transfer
- Text-to-image (TTI) generation

Source: https://arxiv.org/pdf/2104.14951.pdf

Image Generation



- Class-conditional generation
- Super resolution
- **Image Editing**

- **Inpainting** fills in the (pre-specified) missing pixels
- **Colorization** restores color to a greyscale image
- **Uncropping** creates a photo-realistic reconstruction of a missing side of an image

Source: https://arxiv.org/pdf/2111.05826.pdf

Image Generation

- Given two images, present the semantic content of the *source* image in the style of the *reference* image

- Class-conditional generation
- Super resolution
- Image Editing
- **Style transfer**
- Text-to-image (TTI) generation

Source: https://arxiv.org/pdf/1508.06576.pdf

Image Generation

Prompt: A propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese.

- Given a text description, sample an image that depicts the prompt

- Class-conditional generation
- Super resolution
- Image Editing
- Style transfer
- **Text-to-image (TTI) generation**

Source: https://arxiv.org/pdf/2307.01952.pdf

Image Generation

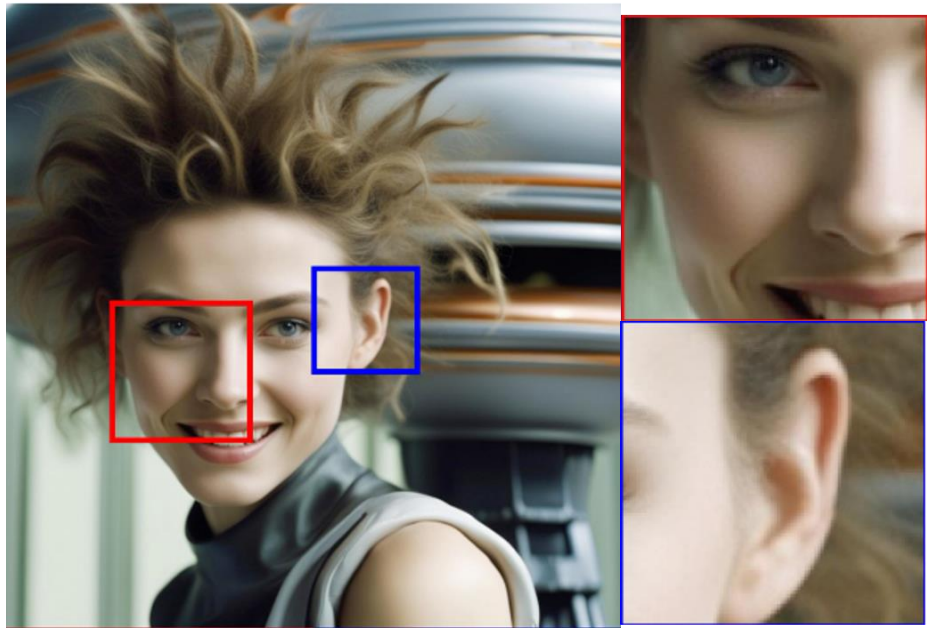
Prompt: Epic long distance cityscape photo of New York City flooded by the ocean and overgrown buildings and jungle ruins in rainforest, at sunset, cinematic shot, highly detailed, 8k, golden light

- Class-conditional generation
- Super resolution
- Image Editing
- Style transfer
- **Text-to-image (TTI) generation**

Source: https://arxiv.org/pdf/2307.01952.pdf

Image Generation

Prompt: close up headshot, futuristic young woman, wild hair sly smile in front of gigantic UFO, dslr, sharp focus, dynamic composition

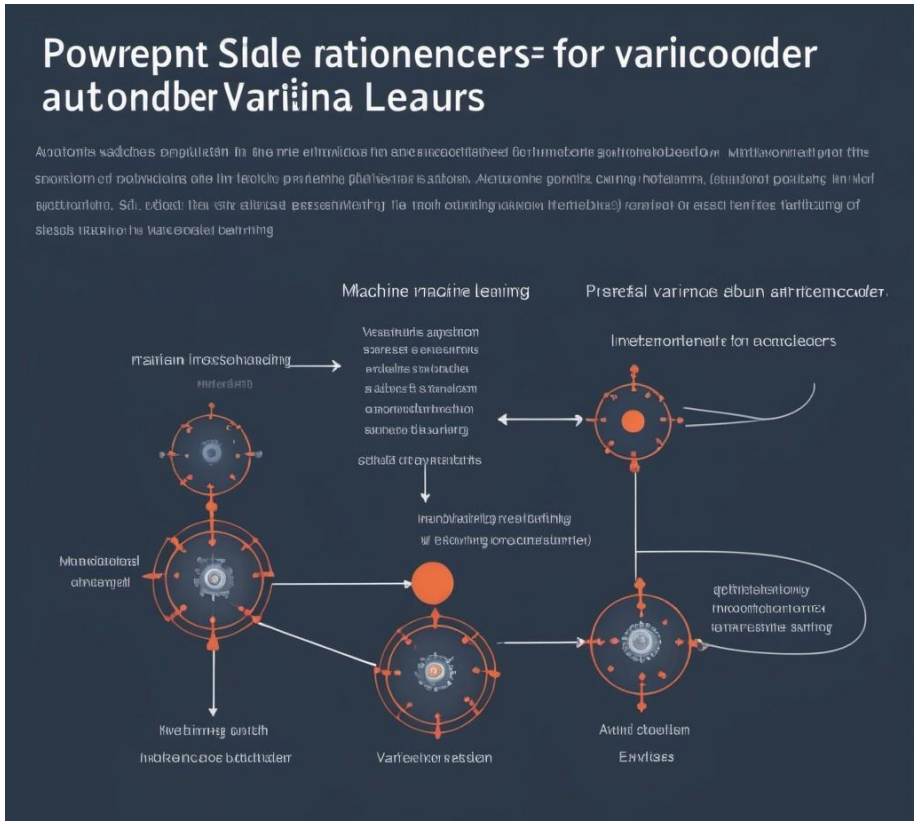


- Class-conditional generation
- Super resolution
- Image Editing
- Style transfer
- **Text-to-image (TTI) generation**

Source: https://arxiv.org/pdf/2307.01952.pdf

Slide Generation?

Prompt: powerpoint slide explaining variational autoencoders for an intro to ML course, easy to follow, with an explanation of the evidence lower bound



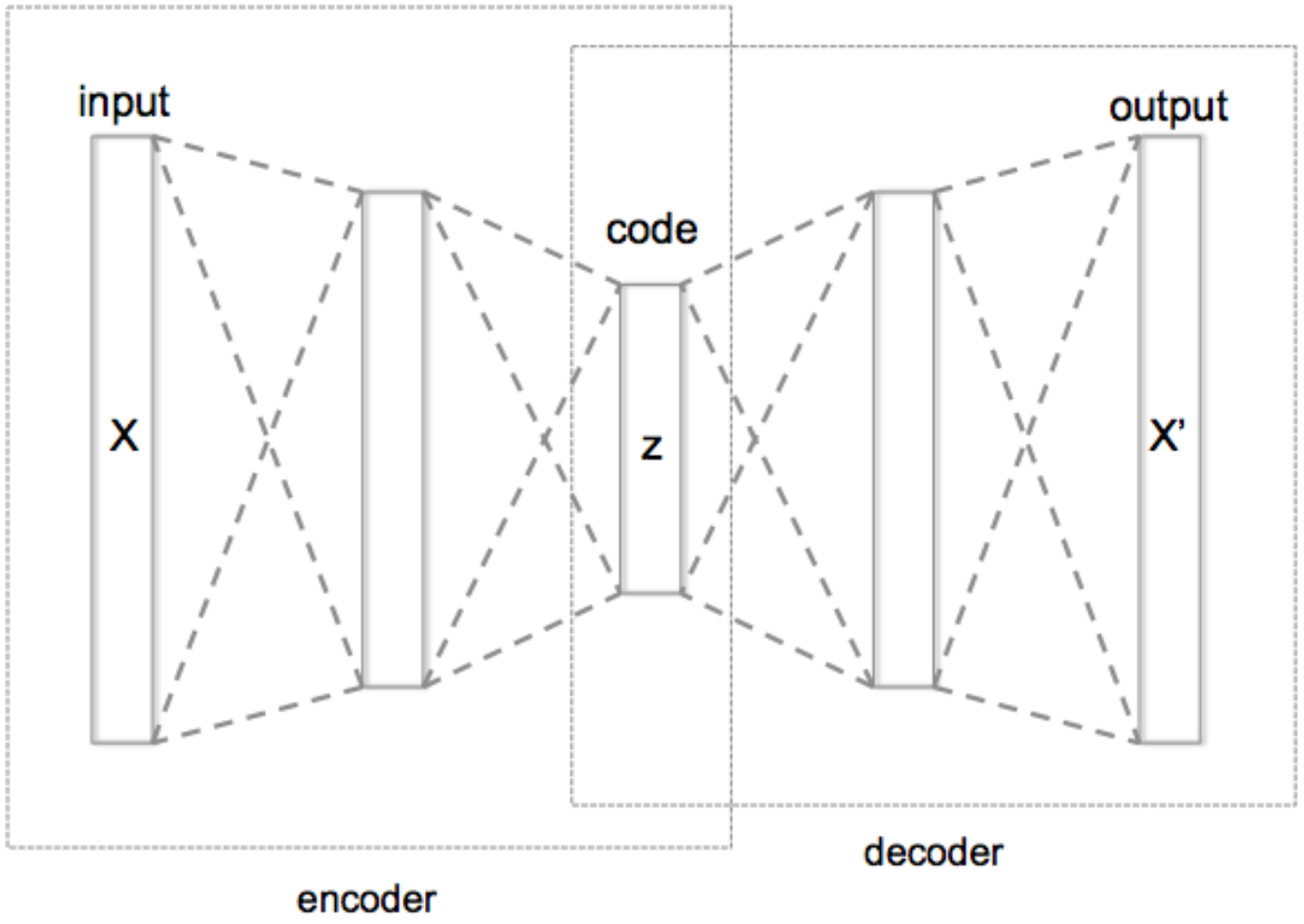
- Class-conditional generation
- Super resolution
- Image Editing
- Style transfer
- **Text-to-image (TTI) generation**

Source: https://stablediffusionweb.com/app/image-generator

Image Generation

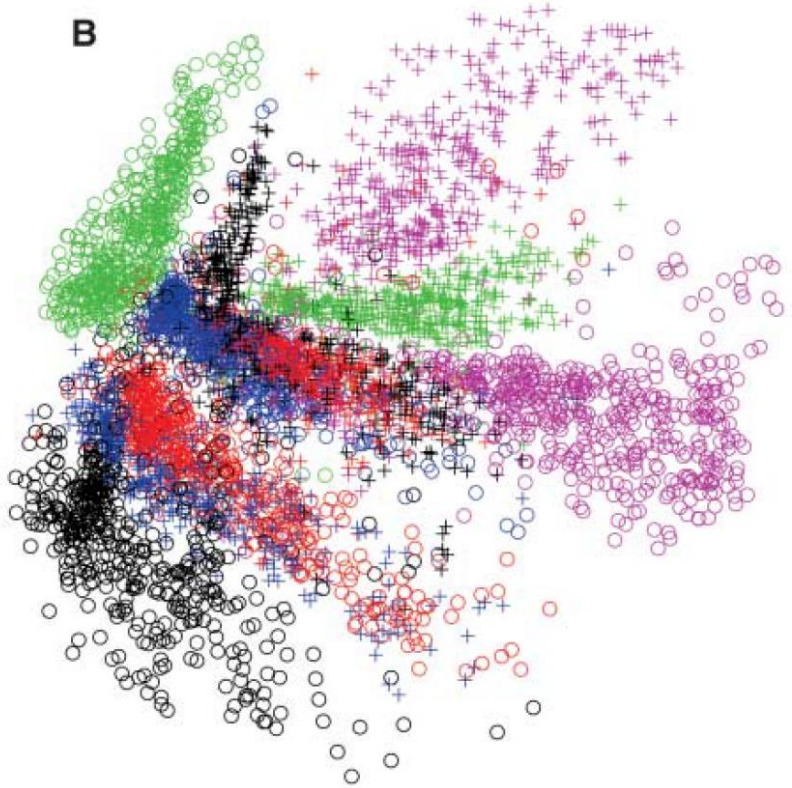
- Fundamental challenge: images are incredibly high-dimensional objects with complex relationships between elements
- Idea: learn a low-dimensional representation of images, sample points in the low-dimensional space and project them up to the original image space

Recall: Autoencoders



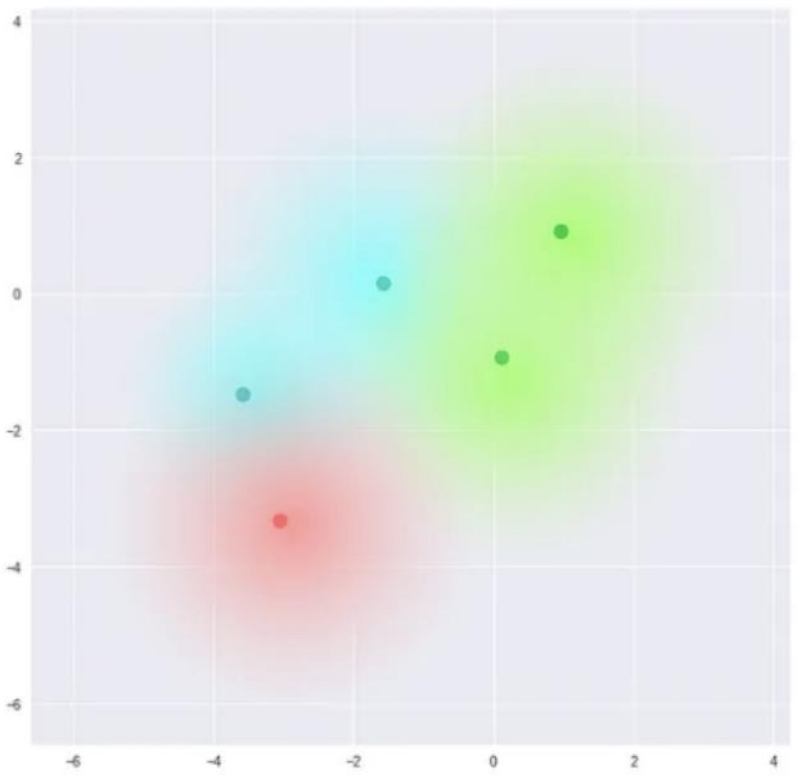
Source: https://en.wikipedia.org/wiki/Autoencoder#/media/File:Autoencoder_structure.png

- Issue: latent space is sparse…
  - Sampling from latent space of an autoencoder creates outputs that are effectively identical to images in the training dataset

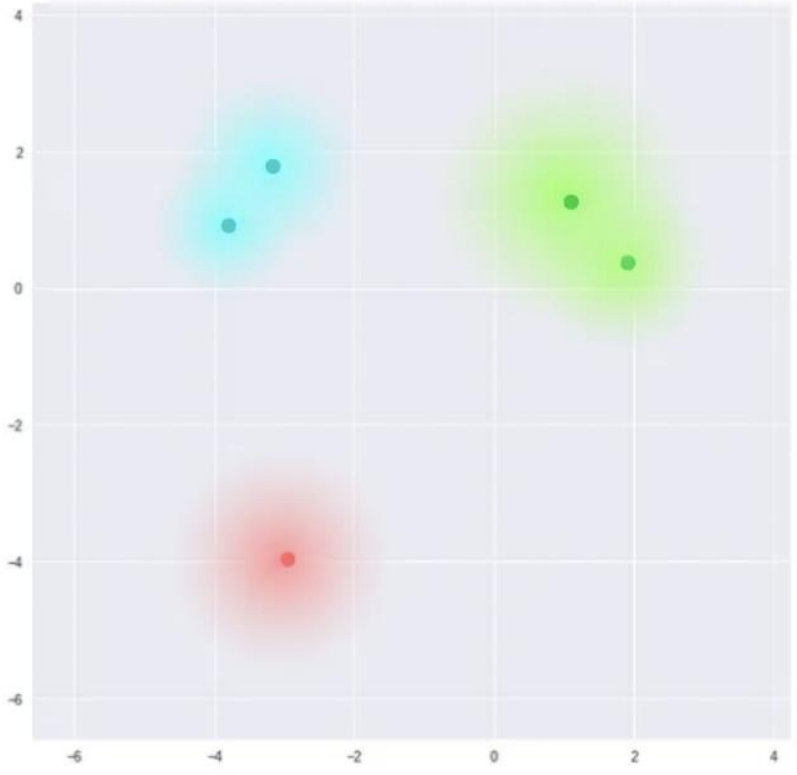


Autoencoder Latent Space

Source: https://www.science.org/doi/10.1126/science.1127647



What we require



What we may inadvertently end up with

Autoencoder Latent Space

Source: https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

Variational Autoencoder: Network Perspective

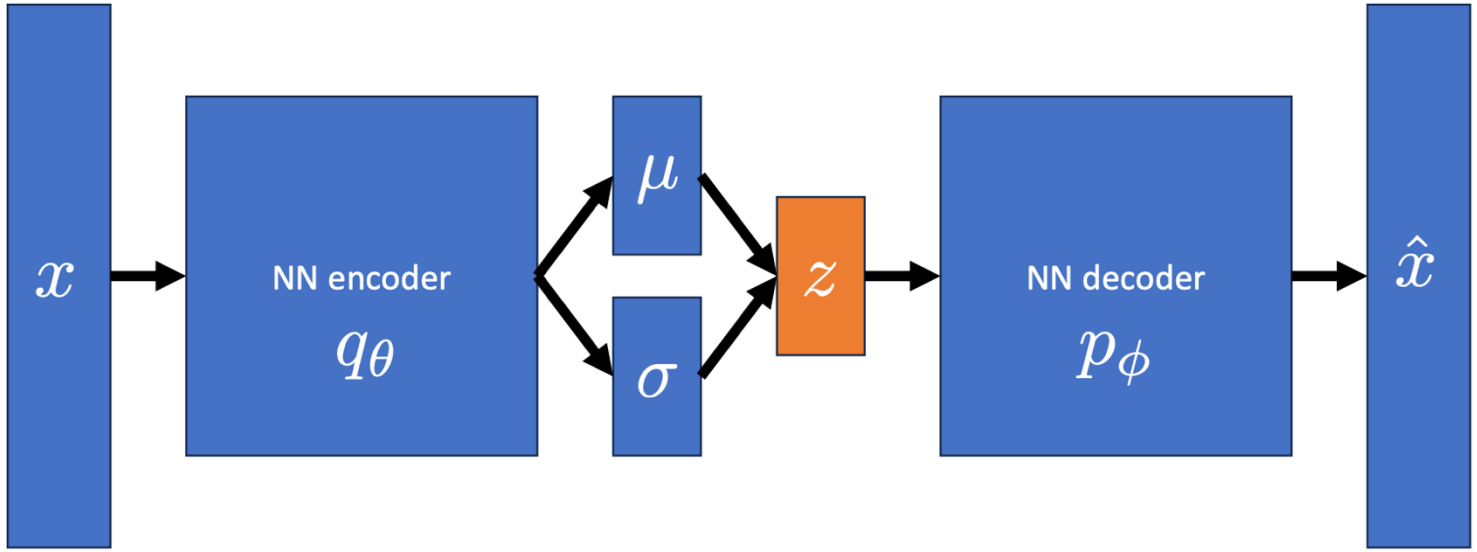
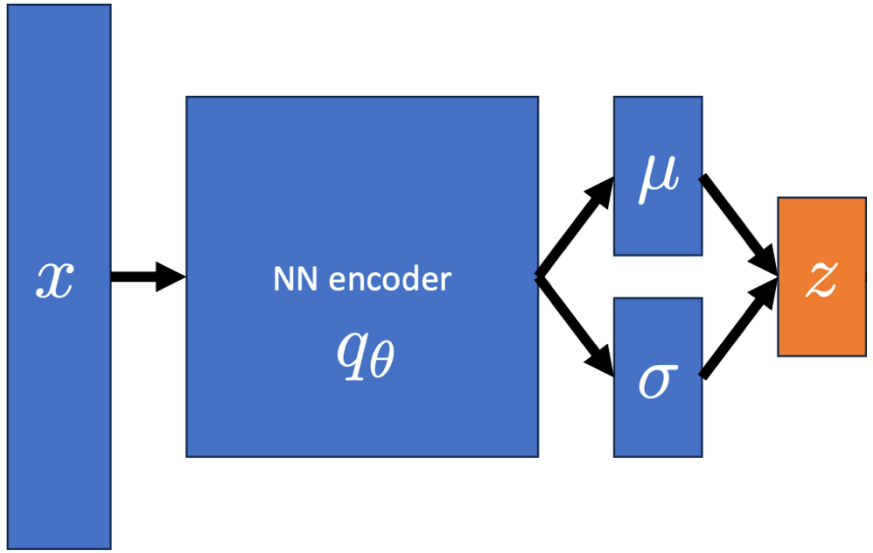


Figure courtesy of Zack Lipton

Variational Autoencoder: Network Perspective

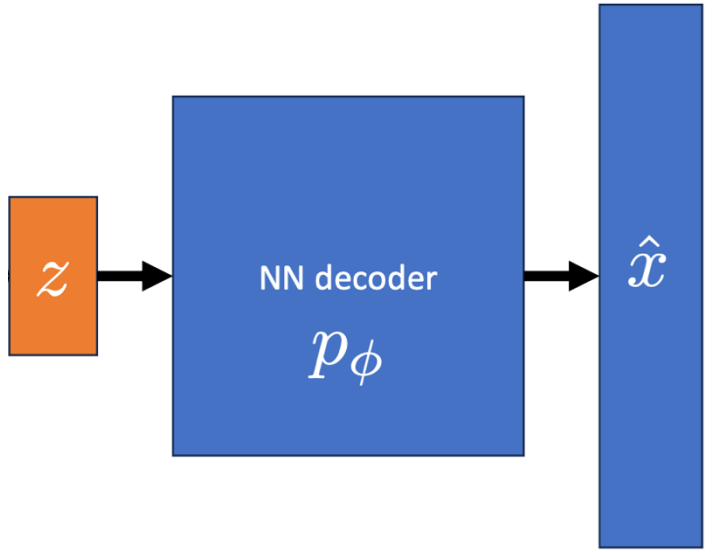


- Encoder learns a mean vector and a (diagonal) covariance matrix for each input
- These are used to *sample* a latent representation e.g.,

$$\boldsymbol{z}^{(i)} \mid \boldsymbol{x}^{(i)} \sim \mathcal{N}\left(\mu_\theta\left(\boldsymbol{x}^{(i)}\right), \sigma_\theta^2\left(\boldsymbol{x}^{(i)}\right)\right)$$

Figure courtesy of Zack Lipton

Variational Autoencoder: Network Perspective

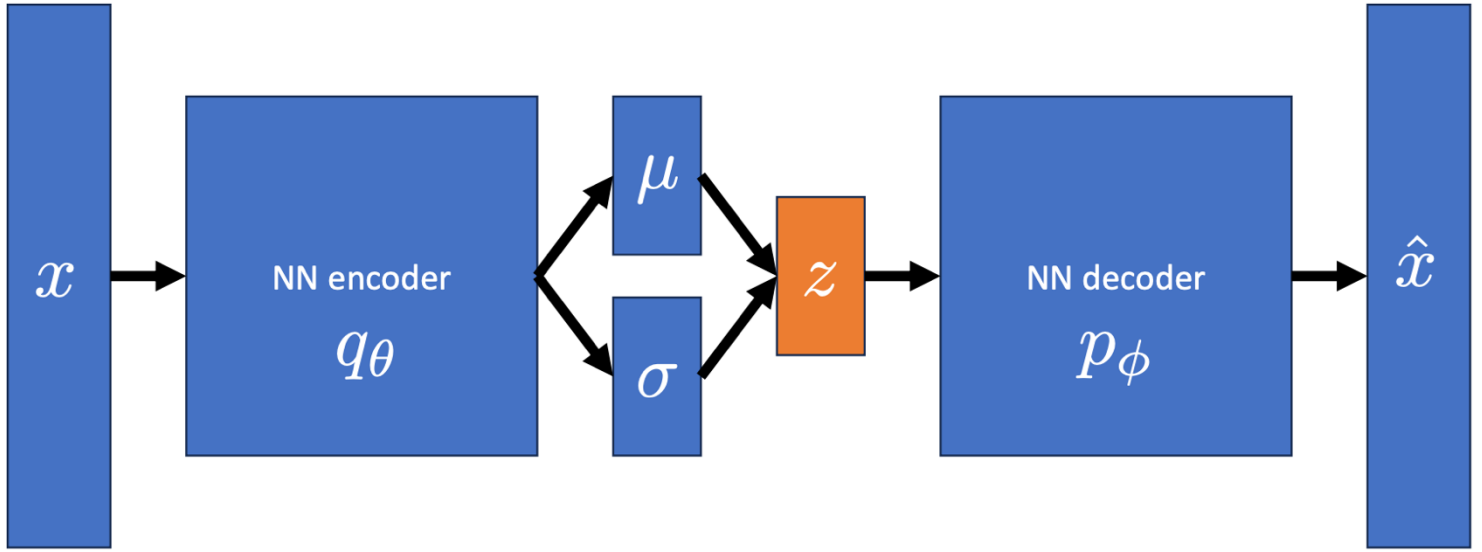


- Decoder tries to minimize the reconstruction error *in expectation* between $\boldsymbol{x}^{(i)}$ and a sample from another (conditional) distribution e.g.,

$$\hat{\boldsymbol{x}}^{(i)} \mid \boldsymbol{z}^{(i)} \sim \mathcal{N}\left(\mu_{\boldsymbol{\phi}}\left(\boldsymbol{z}^{(i)}\right), \sigma^2_{\boldsymbol{\phi}}\left(\boldsymbol{z}^{(i)}\right)\right)$$

Figure courtesy of Zack Lipton

Variational Autoencoder: Network Perspective



- Objective: minimize the expected reconstruction error plus a *regularizer* that encourages a dense latent space

$$\mathcal{L}(\theta, \phi) = \sum_{i=1}^{N} \left(-\mathbb{E}_{q_\theta(\boldsymbol{z}|\boldsymbol{x}^{(i)})} \left[\log p_\phi \left(\boldsymbol{x}^{(i)} | \boldsymbol{z} \right) \right] \right)$$

$$+ KL \left(q_\theta \left(\boldsymbol{z} | \boldsymbol{x}^{(i)} \right) \parallel p(\boldsymbol{z}) \right)$$

Figure courtesy of Zack Lipton

Variational Autoencoder: Probabilistic Perspective

- Model: assume data is generated by

1. First, drawing a sample from some latent space $p(\boldsymbol{z})$
2. Then, drawing a sample from a conditional distribution $p_\phi(\boldsymbol{x}|\boldsymbol{z})$

- Idea: maximize the *evidence* of our data

$$\mathcal{L}^{(i)}(\phi) = p\left(\boldsymbol{x}^{(i)}\right) = \int p_\phi\left(\boldsymbol{x}^{(i)}|\boldsymbol{z}\right)p(\boldsymbol{z})d\boldsymbol{z}$$

- Issue: for most interesting distributions, this integral is going to be intractable…

Evidence Lower Bound (ELBO)

$$\ell^{(i)}(\phi) = \log p\left(\boldsymbol{x}^{(i)}\right) = \mathbb{E}_{q_\theta(\boldsymbol{z}|\boldsymbol{x}^{(i)})}\left[\log p\left(\boldsymbol{x}^{(i)}\right)\right]$$

$$= \mathbb{E}_{q_\theta}\left[\log \frac{p_\phi(\boldsymbol{x}^{(i)}|\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x}^{(i)})}\right]$$

$$= \mathbb{E}_{q_\theta}\left[\log\left(\frac{p_\phi(\boldsymbol{x}^{(i)}|\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x}^{(i)})}\frac{q_\theta(\boldsymbol{z}|\boldsymbol{x}^{(i)})}{q_\theta(\boldsymbol{z}|\boldsymbol{x}^{(i)})}\right)\right]$$

$$= \mathbb{E}_{q_\theta}\left[\log p_\phi\left(\boldsymbol{x}^{(i)}|\boldsymbol{z}\right) - \log\left(\frac{q_\theta(\boldsymbol{z}|\boldsymbol{x}^{(i)})}{p(\boldsymbol{z})}\right) + \log\left(\frac{q_\theta(\boldsymbol{z}|\boldsymbol{x}^{(i)})}{p(\boldsymbol{z}|\boldsymbol{x}^{(i)})}\right)\right]$$

$$= \mathbb{E}_{q_\theta}\left[\log p_\phi\left(\boldsymbol{x}^{(i)}|\boldsymbol{z}\right)\right] - KL\left(q_\theta\left(\boldsymbol{z}|\boldsymbol{x}^{(i)}\right) \parallel p(\boldsymbol{z})\right)$$

$$+ KL\left(q_\theta\left(\boldsymbol{z}|\boldsymbol{x}^{(i)}\right) \parallel p\left(\boldsymbol{z} \mid \boldsymbol{x}^{(i)}\right)\right)$$

$$\geq \mathbb{E}_{q_\theta}\left[\log p_\phi\left(\boldsymbol{x}^{(i)}|\boldsymbol{z}\right)\right] - KL\left(q_\theta\left(\boldsymbol{z}|\boldsymbol{x}^{(i)}\right) \parallel p(\boldsymbol{z})\right)$$

Fun fact: VAE

The VAE objective has a built-in tug-of-war

The lecture's ELBO picture can be summarized as a tradeoff: reconstruct the input well, but also keep the approximate posterior close to the prior.

ELBO = reconstruction term − KL(q(z|x) || p(z))

Reconstruction term

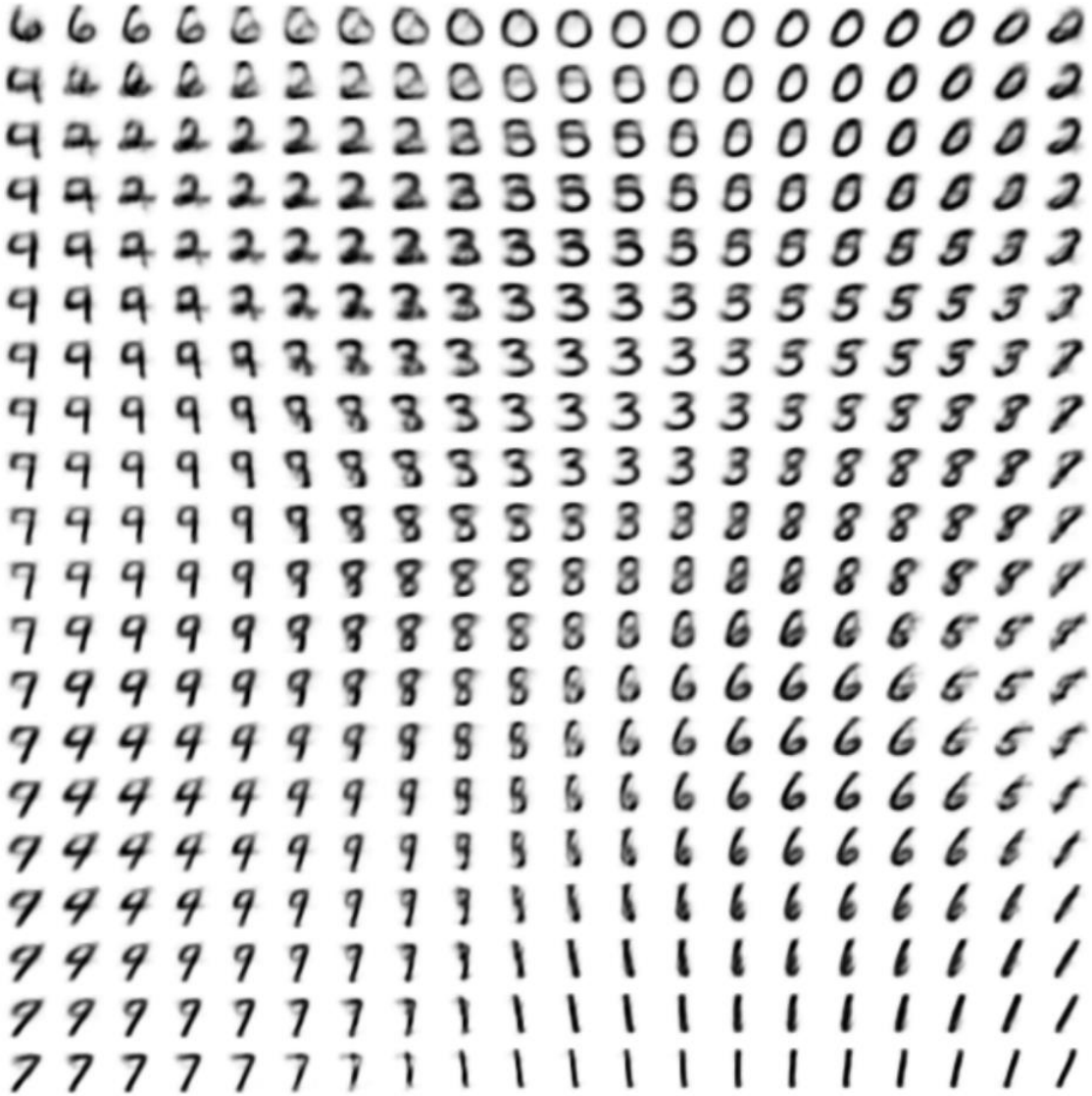
- make decoded $\hat{x}$ look like x
- preserve input information
- too strong → blurry / average-like outputs

KL regularization

- push q(z|x) toward prior p(z)
- make latent space smoother and sampleable
- too weak → latent space becomes messy

what happens if you weight the KL term too much or too little?

Variational Autoencoder: Latent Space Visualization



Source: https://arxiv.org/pdf/1312.6114.pdf

Variational Autoencoder: Generated Samples

Source: https://arxiv.org/pdf/1312.6114.pdf

Variational Autoencoder: Generated Samples?

Source: https://arxiv.org/pdf/1312.6114.pdf?

Can we encode the idea that samples should be indistinguishable from real observations into the objective function?

Source: https://arxiv.org/pdf/1312.6114.pdf

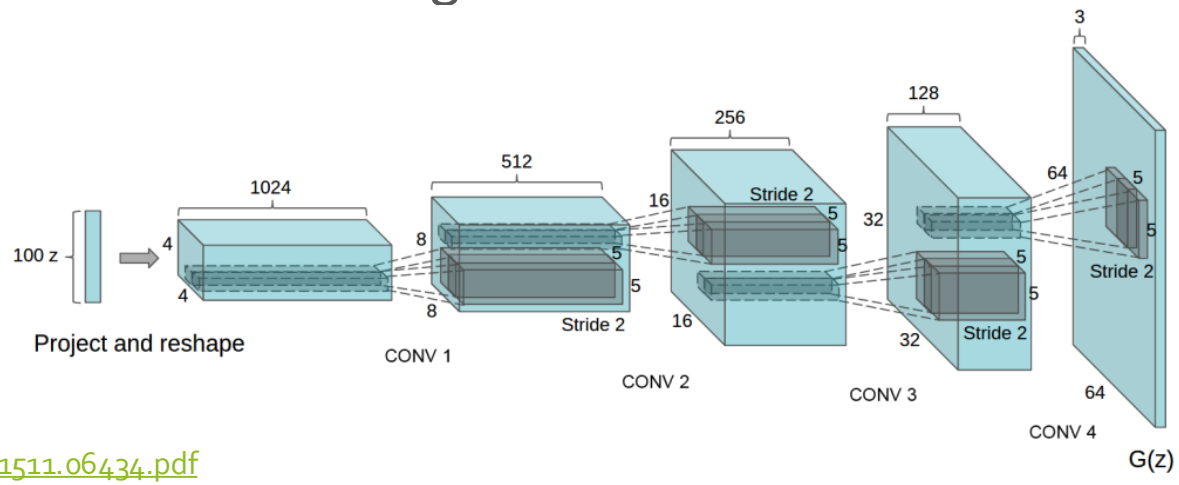
Source: MNIST

Generative Adversarial Networks (GANs)

- A GAN consists of two (deterministic) models:
  - a **generator** that takes a vector of random noise as input, and generates an image
  - a **discriminator** that takes in an image classifies whether it is real (label = 1) or fake (label = 0)
  - Both models are typically (but not necessarily) neural networks
- During training, the GAN plays a two-player minimax game: the generator tries to create realistic images to fool the discriminator and the discriminator tries to identify the real images from the fake ones

Generative Adversarial Networks (GANs)

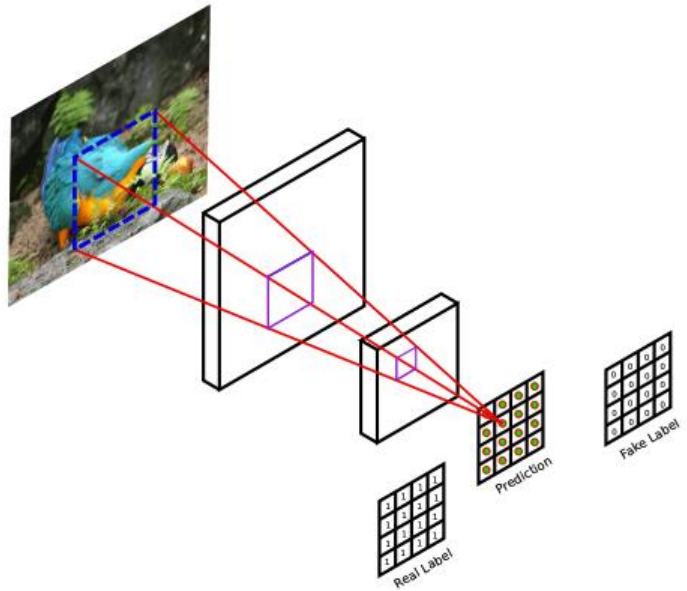
- A GAN consists of two (deterministic) models:
  - a **generator** that takes a vector of random noise as input, and generates an image
- Example generator: DCGAN
  - An inverted CNN with four *fractionally-strided* convolution layers that grow the size of the image from layer to layer; final layer has three channels to generate color images



Source: https://arxiv.org/pdf/1511.06434.pdf

Generative Adversarial Networks (GANs)

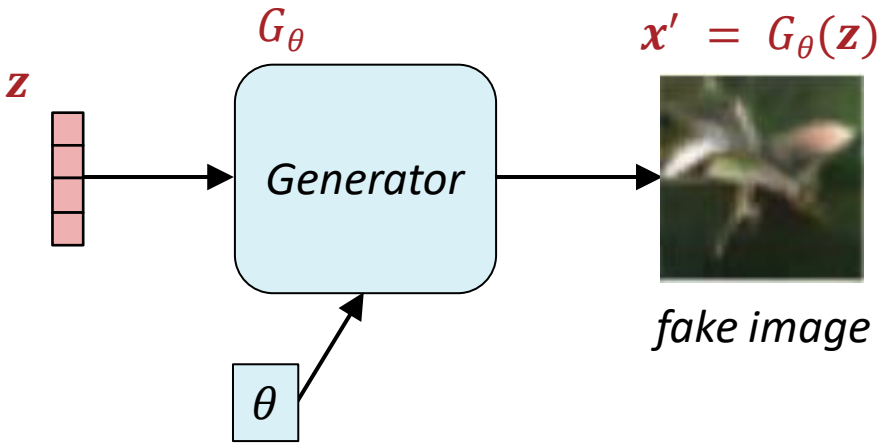
- A GAN consists of two (deterministic) models:
  - a **generator** that takes a vector of random noise as input, and generates an image
  - a **discriminator** that takes in an image classifies whether it is real (label = 1) or fake (label = 0)
- Example discriminator: PatchGAN
  - Traditional CNN that looks at each patch of the image and tries to predict whether it is real or fake; can help encourage to generator to avoid creating blurry images



Source: https://arxiv.org/pdf/1803.07422.pdf

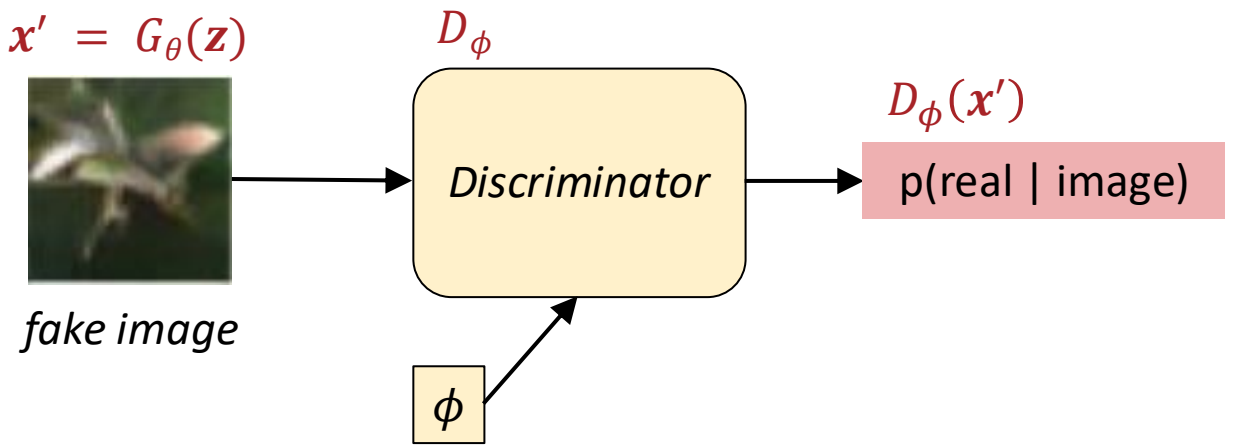
Generative Adversarial Networks (GANs): Training

- A GAN consists of two (deterministic) models:
  - a **generator** that takes a vector of random noise as input, and generates an image
  - a **discriminator** that takes in an image classifies whether it is real (label = 1) or fake (label = 0)
  - Both models are typically (but not necessarily) neural networks
- During training, the GAN plays a two-player minimax game: the generator tries to create realistic images to fool the discriminator and the discriminator tries to identify the real images from the fake ones



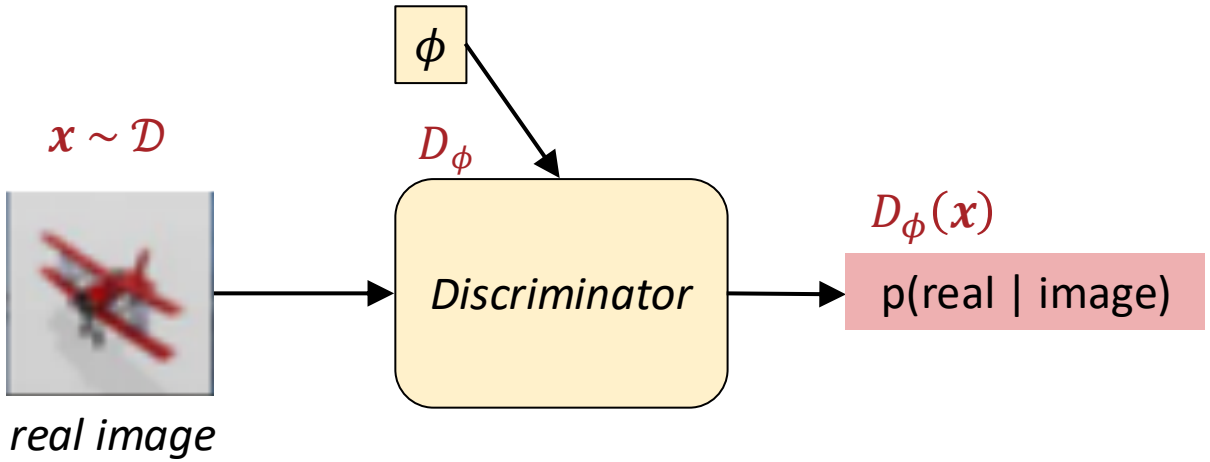
GANs: Architecture

Figure courtesy of Matt Gormley



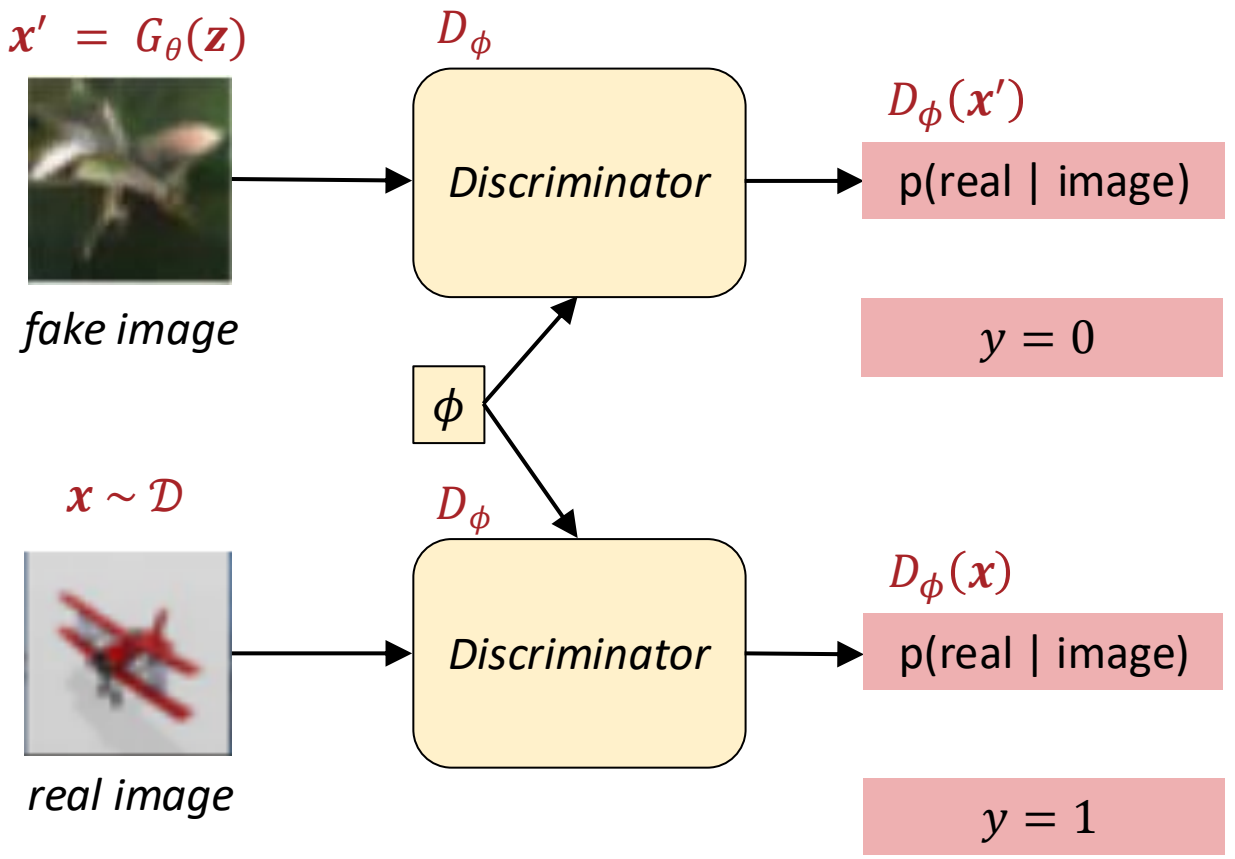
GANs: Architecture

Figure courtesy of Matt Gormley



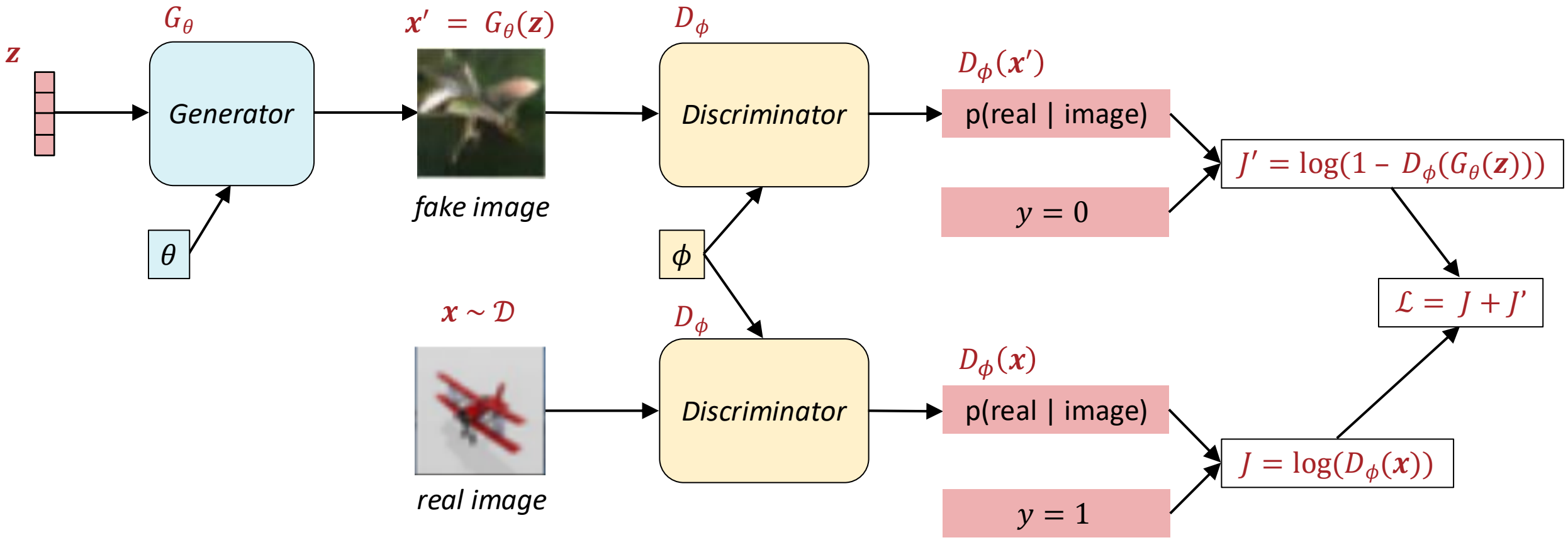
GANs: Architecture

Figure courtesy of Matt Gormley



GANs: Architecture

Figure courtesy of Matt Gormley



GANs: Architecture

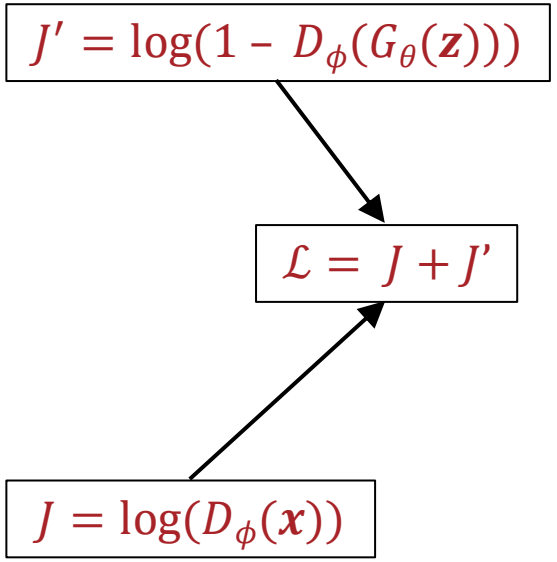
Figure courtesy of Matt Gormley

The discriminator is trying to maximize the usual cross-entropy loss for binary classification with labels {real = 1, fake = 0}

$$\min_{\phi} \log\left(D_\phi(\mathbf{x}^{(i)})\right) + \log\left(1 - D_\phi(G_\theta(\mathbf{z}^{(i)}))\right)$$

$$\max_{\theta} \log\left(1 - D_\phi(G_\theta(\mathbf{z}^{(i)}))\right)$$

The generator is trying to maximize the likelihood of its generated (fake) image being classified as real, according to a fixed discriminator



GANs: Architecture

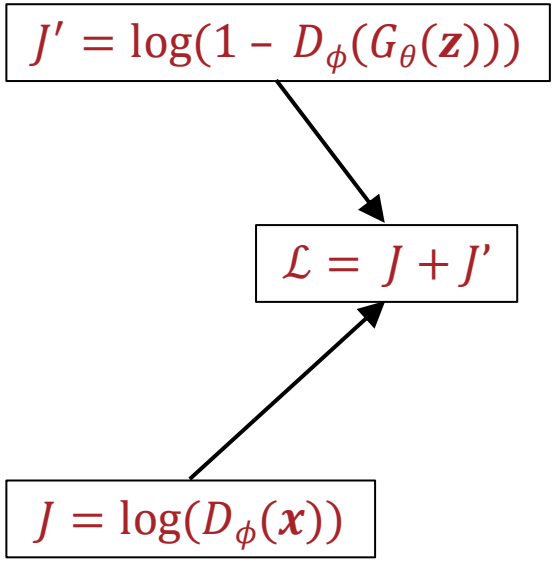
Both objectives (and hence, their sum) are differentiable

$$\min_{\phi} \log\left(D_\phi(\mathbf{x}^{(i)})\right) + \log\left(1 - D_\phi(G_\theta(\mathbf{z}^{(i)}))\right)$$

$$\max_{\theta} \log\left(1 - D_\phi(G_\theta(\mathbf{z}^{(i)}))\right)$$

Training alternates between:

1. Keeping θ fixed and backpropagating through D_ϕ
2. Keeping ϕ fixed and backpropagating through G_θ



GANs: Architecture

Myth Busters

MythBusters #2

“The discriminator should win.”

Myth: if the discriminator becomes nearly perfect, GAN training is going great.

Reality: GANs are a game. If the discriminator becomes too strong too early, the generator may get poor learning signals. Training works best when both models keep each other improving.

The lecture’s story is adversarial training, not “one model crushes the other.”

Quick Game

Quick game: which model sounds most appropriate?

Match each goal to AE, VAE, or GAN.

Goals

- **A** compress / reconstruct inputs
- **B** smooth latent space + generation by sampling z
- **C** make samples hard to distinguish from real images

Quick Game

Quick game: which model sounds most appropriate?

Match each goal to AE, VAE, or GAN.

Goals

A compress / reconstruct inputs

B smooth latent space + generation by sampling z

C make samples hard to distinguish from real images

Reveal

AE
reconstruction / compression

A → AE

VAE
sampleable latent space

B → VAE

GAN
realistic-looking samples

C → GAN

Quick Game

Two truths and a lie (generative models)

Vote for the lie: A, B, or C.

A. A VAE adds a KL term to encourage a smoother, sampleable latent space.

B. In a GAN, the discriminator tries to distinguish real images from generated ones.

C. A plain autoencoder guarantees that randomly sampled latent vectors decode to realistic images.

Quick Game

Two truths and a lie (generative models)

Vote for the lie: A, B, or C.

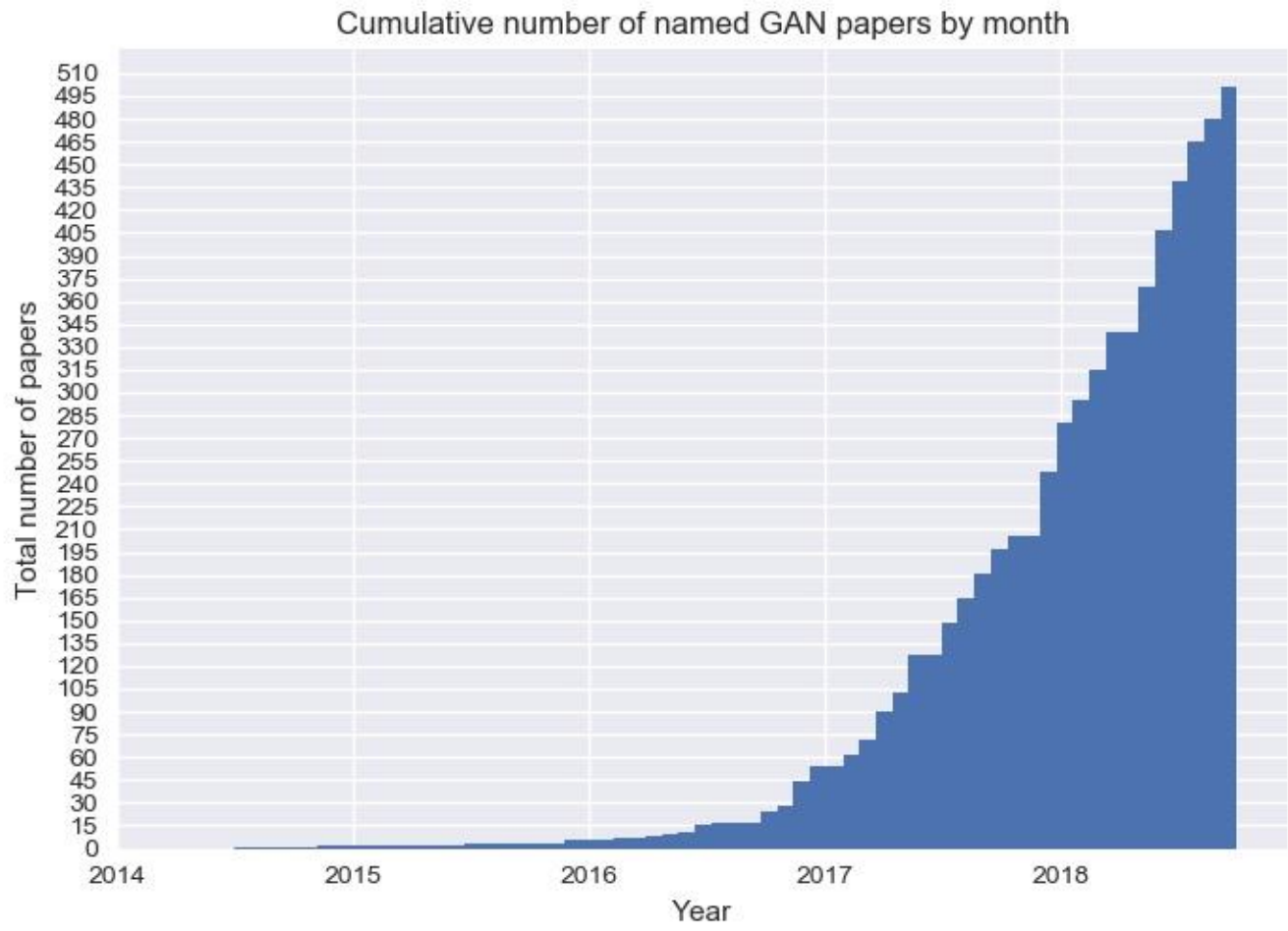
A. A VAE adds a KL term to encourage a smoother, sampleable latent space.

B. In a GAN, the discriminator tries to distinguish real images from generated ones.

C. A plain autoencoder guarantees that randomly sampled latent vectors decode to realistic images.

Reveal: C is the lie.

GANs Everywhere!



Source: https://github.com/hindupuravinash/the-gan-zoo/tree/master

The rise of vision transformers and diffusion models

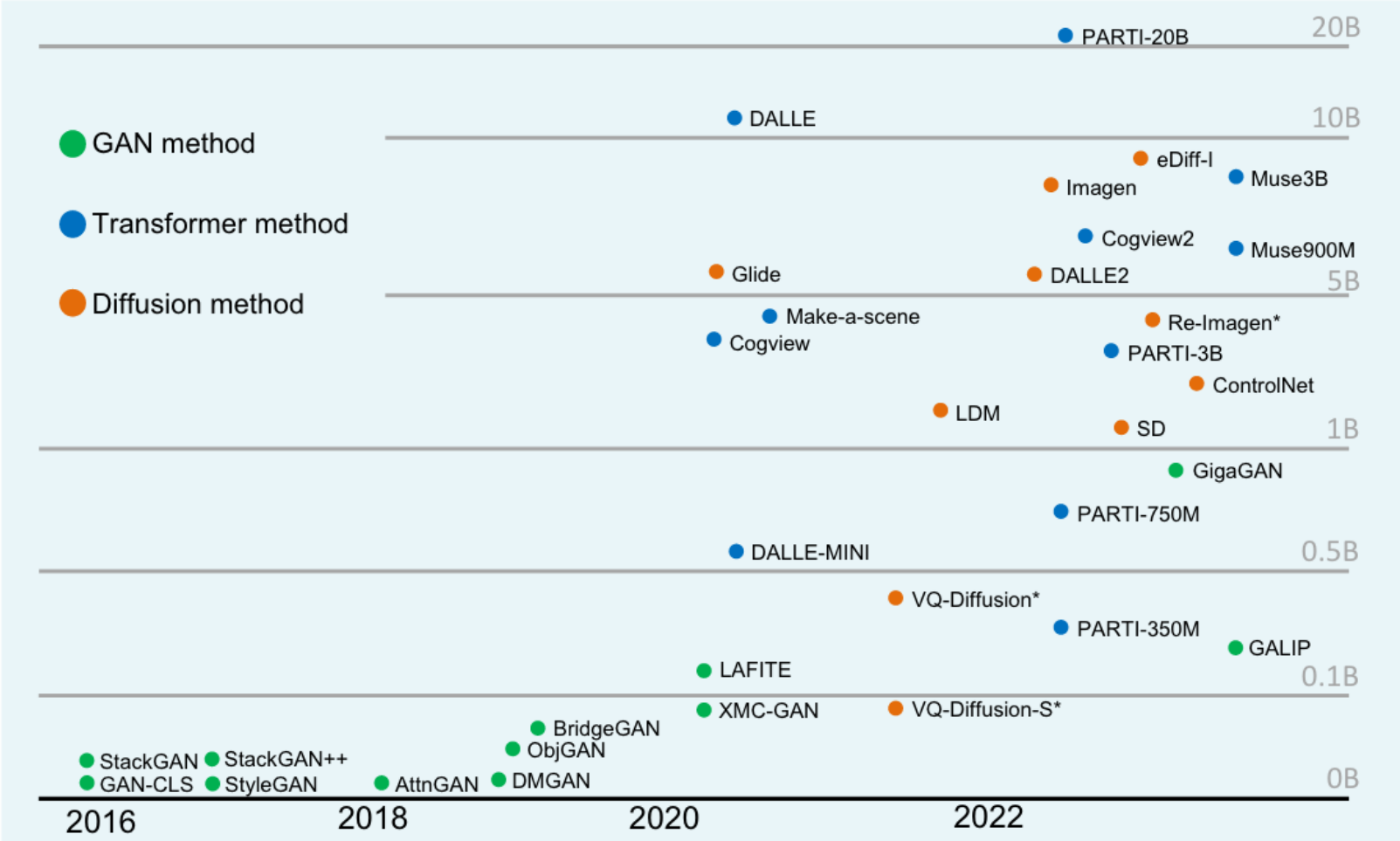


Fig. 5. Timeline of TTI model development, where green dots are GAN TTI models, blue dots are autoregressive Transformers and orange dots are Diffusion TTI models. Models are separated by their parameter, which are in general counted for all their components. Models with asterisk are calculated without the involvement of their text encoders.

Source: https://arxiv.org/pdf/2309.00810.pdf

Vision Transformer (ViT)

Model:

- model is almost identical to BERT
- instead of words as input the inputs are PxP pixel image patches, P ∈ {14, 16, 32} (no overlap)
- each patch is embedded linearly into a vector of size 1024
- 1D positional embeddings

Training:

- for pre-training, optimize for image classification on large supervised dataset (e.g. ImageNet 21K, JFT-300M)—same setup as a CNN
- for fine-tuning, learn a new classification head on a small dataset (e.g. CIFAR-100)

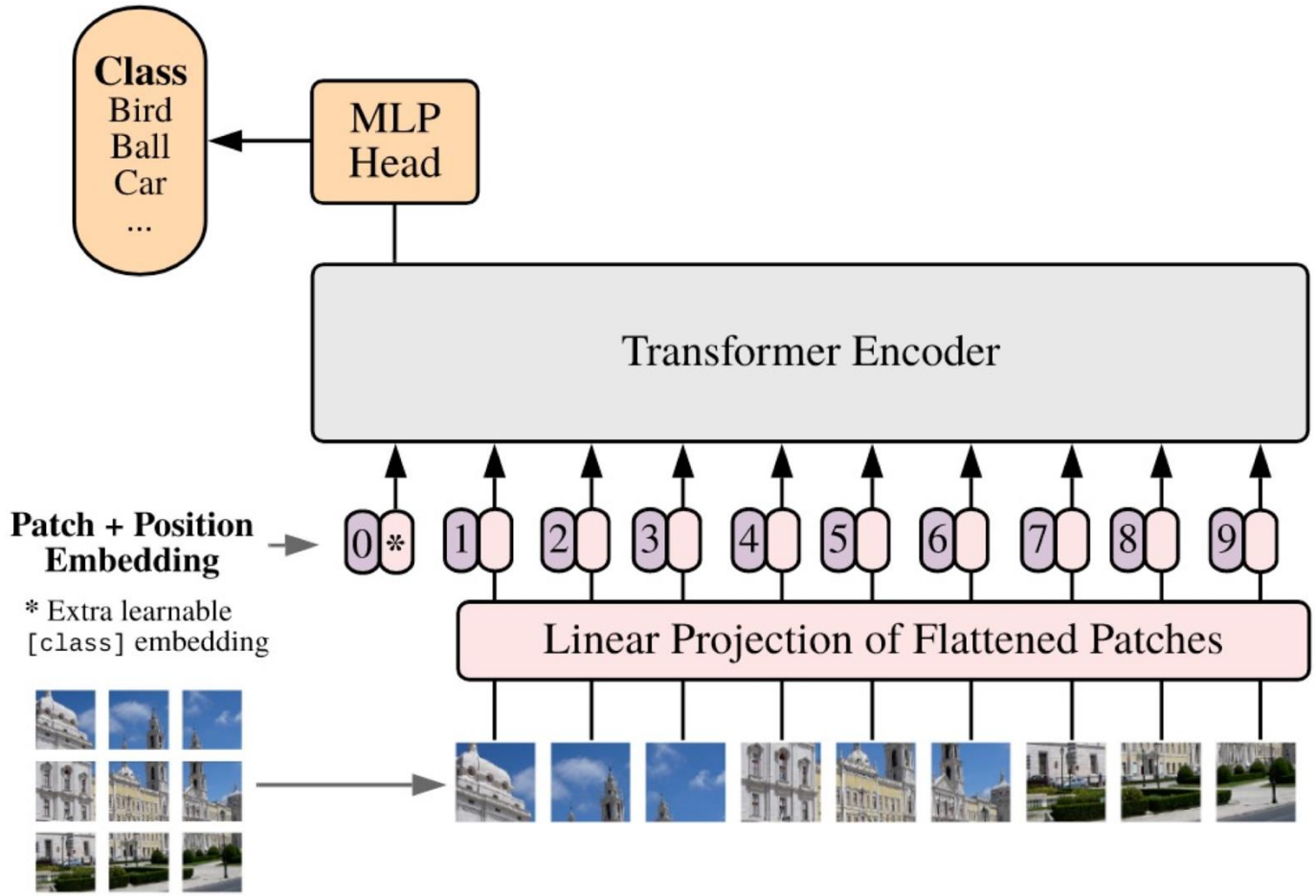


Figure from https://arxiv.org/abs/2010.11929

Vision Transformer (ViT)

Model:

- model is almost identical to BERT
- instead of words as input the inputs are PxP pixel image patches, P ∈ {14, 16, 32} (no overlap)
- each patch is embedded linearly into a vector of size 1024
- 1D positional embeddings

Training:

- for pre-training, optimize for image classification on large supervised dataset (e.g. ImageNet 21K, JFT-300M)—same setup as a CNN
- for fine-tuning, learn a new classification head on a small dataset (e.g. CIFAR-100)

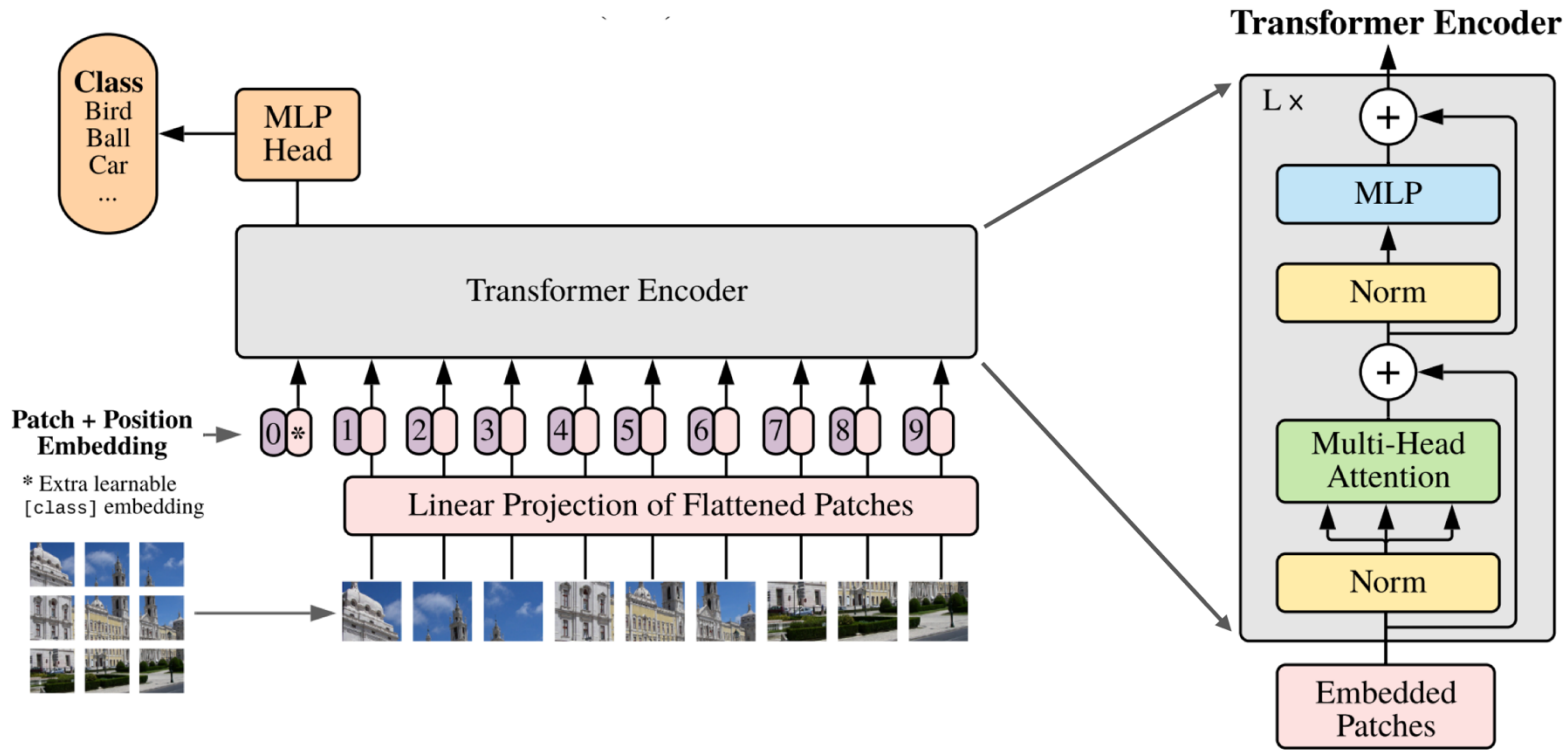


Figure from https://arxiv.org/abs/2010.11929

Vision Transformer (ViT)

Model:

- model is almost identical to BERT
- instead of words as input the inputs are PxP pixel image patches, P ∈ {14, 16, 32} (no overlap)
- each patch is embedded linearly into a vector of size 1024
- 1D positional embeddings

Training:

- for pre-training, optimize for image classification on large supervised dataset (e.g. ImageNet 21K, JFT-300M)—same setup as a CNN
- **for fine-tuning, learn a new classification head on a small dataset (e.g. CIFAR-100)**

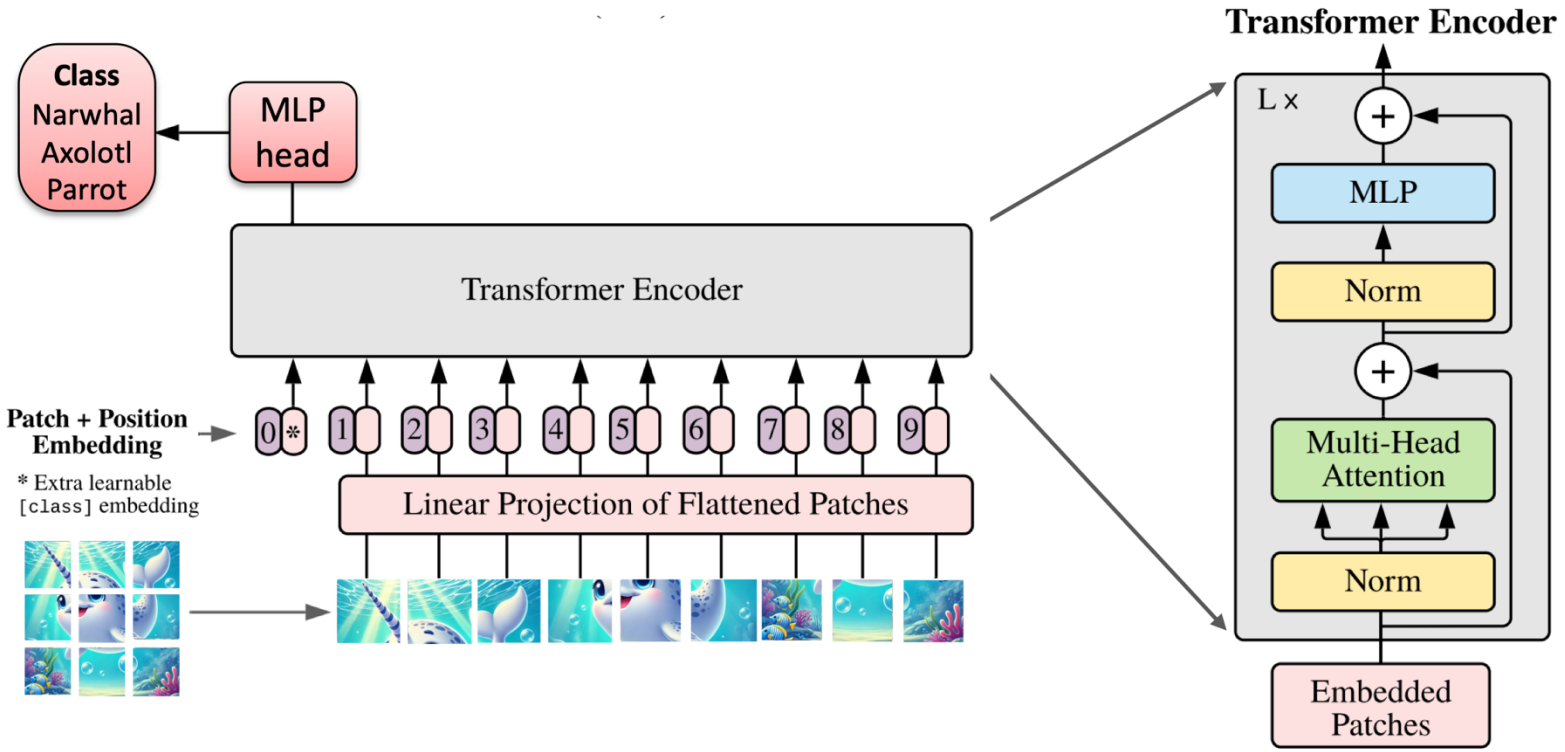
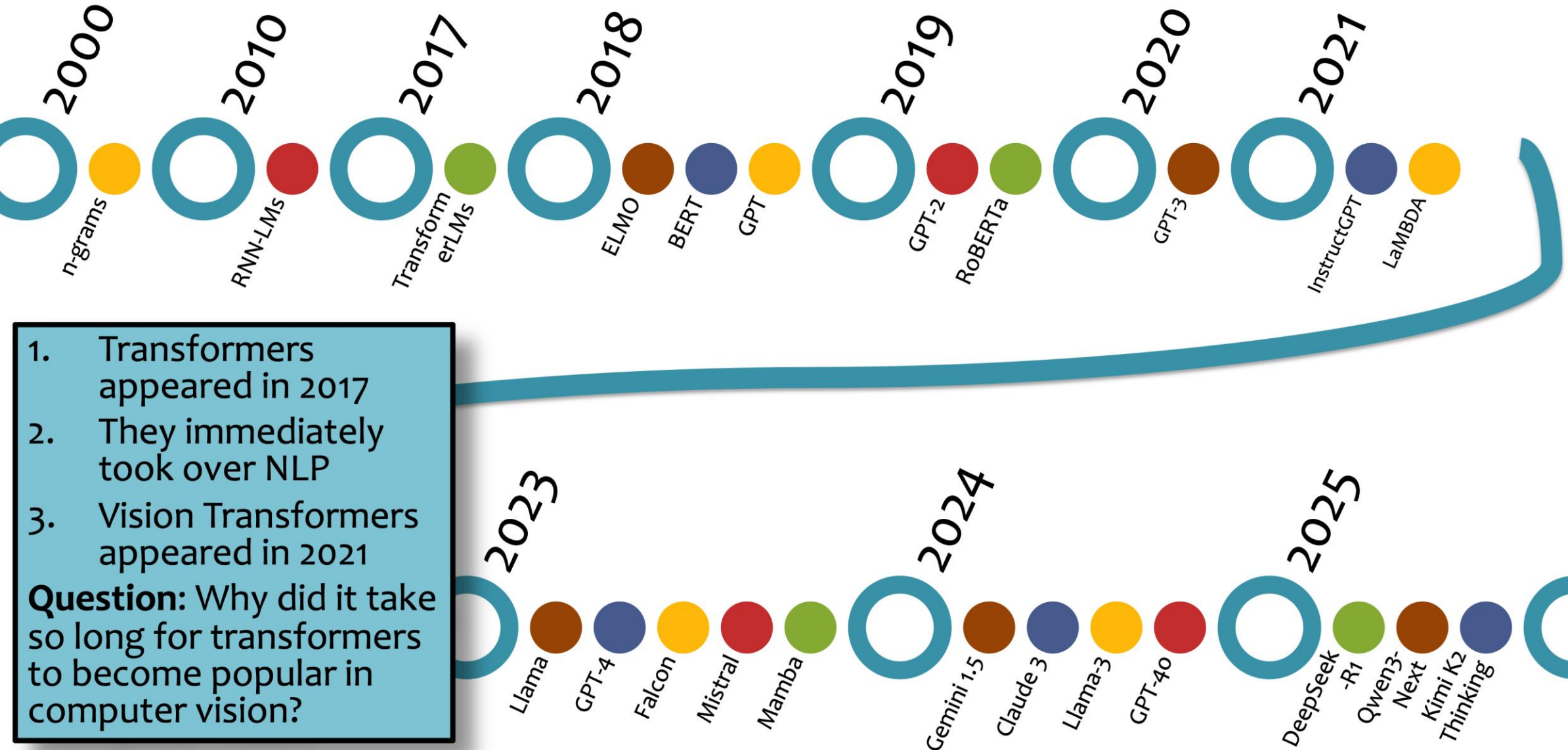


Figure from Henry Chai and https://arxiv.org/abs/2010.11929

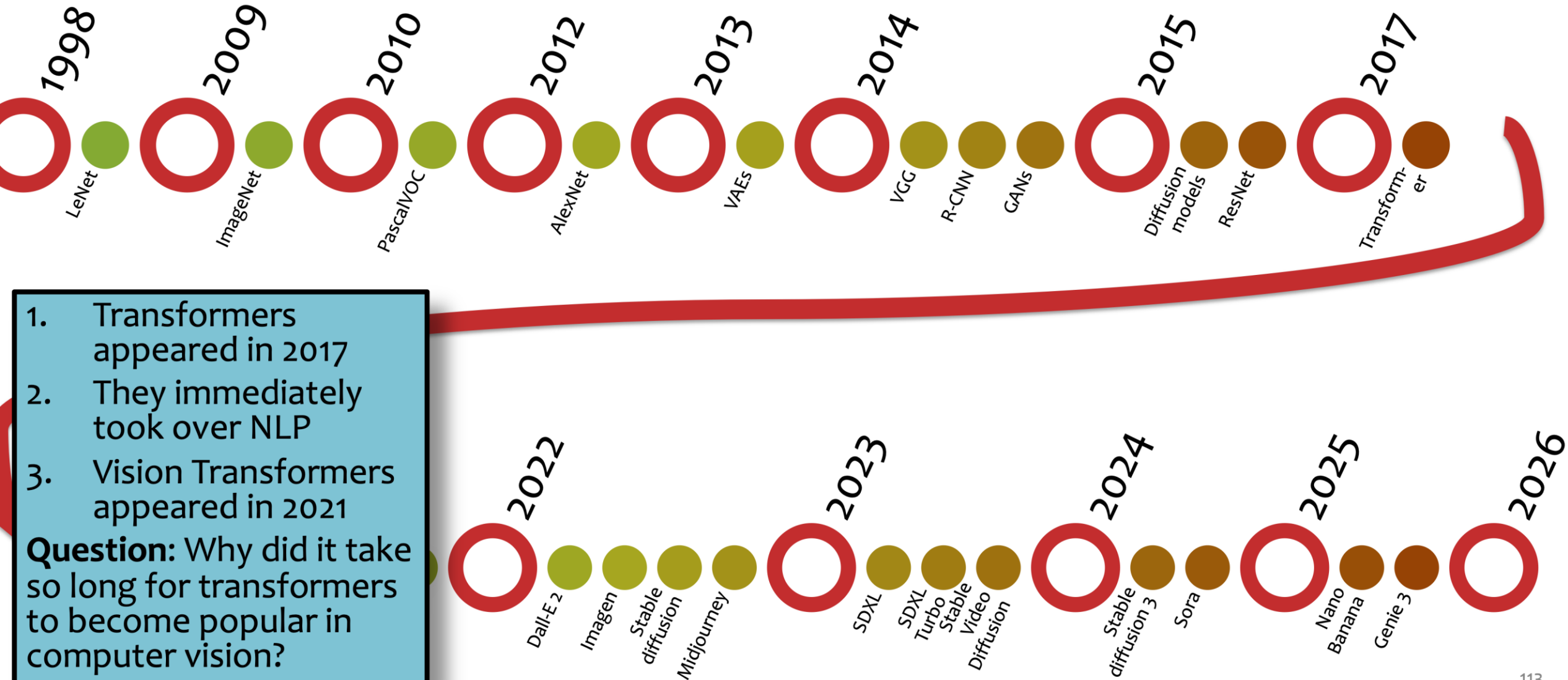
Timeline: Language Modeling



1. Transformers appeared in 2017
2. They immediately took over NLP
3. Vision Transformers appeared in 2021

Question: Why did it take so long for transformers to become popular in computer vision?

Timeline: Image Generation



1. Transformers appeared in 2017
2. They immediately took over NLP
3. Vision Transformers appeared in 2021

Question: Why did it take so long for transformers to become popular in computer vision?

Timeline: Image Generation

1. Transformers appeared in 2017
2. They immediately took over NLP
3. Vision Transformers appeared in 2021

Question: Why did it take so long for transformers to become popular in computer vision?

AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Alexey Dosovitskiy[*,†], **Lucas Beyer**[*], **Alexander Kolesnikov**[*], **Dirk Weissenborn**[*], **Xiaohua Zhai**[*], **Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby**[*,†]

[*]equal technical contribution, [†]equal advising

Google Research, Brain Team

{adosovitskiy, neilhoulsby}@google.com

When trained on mid-sized datasets such as ImageNet without strong regularization, these models yield modest accuracies of a few percentage points below ResNets of comparable size. This seemingly discouraging outcome may be expected: Transformers lack some of the inductive biases inherent to CNNs, such as translation equivariance and locality, and therefore do not generalize well when trained on insufficient amounts of data.

However, the picture changes if the models are trained on larger datasets (14M-300M images). We find that large scale training trumps inductive bias. Our Vision Transformer (ViT) attains excellent results when pre-trained at sufficient scale and transferred to tasks with fewer datapoints. When pre-trained on the public ImageNet-21k dataset or the in-house JFT-300M dataset, ViT approaches or beats state of the art on multiple image recognition benchmarks. In particular, the best model reaches the accuracy of 88.55% on ImageNet, 90.72% on ImageNet-ReaL, 94.55% on CIFAR-100, and 77.63% on the VTAB suite of 19 tasks.

Timeline: Image Generation

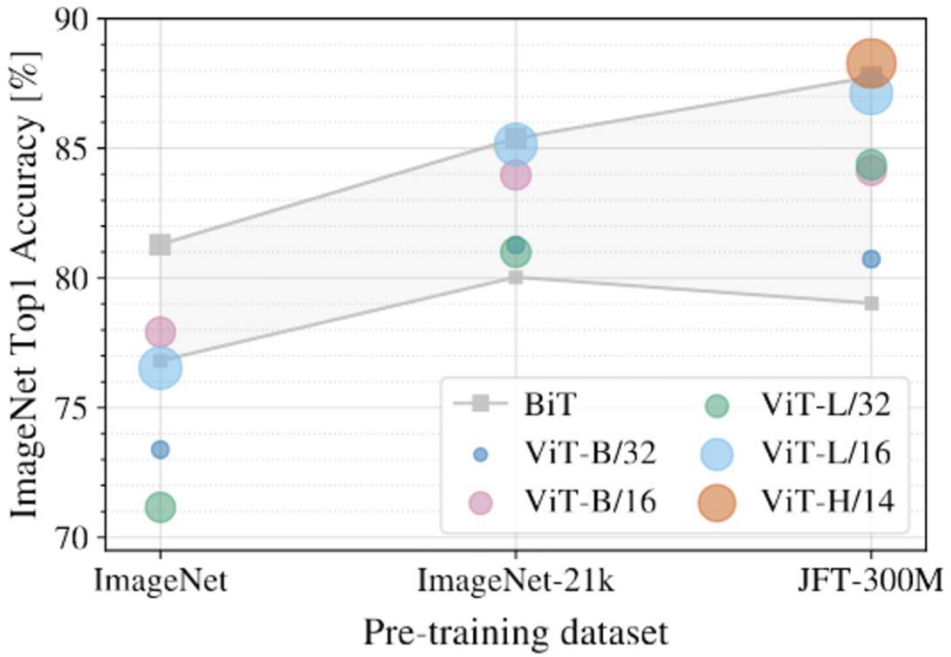


Figure 3: Transfer to ImageNet. While large ViT models perform worse than BiT ResNets (shaded area) when pre-trained on small datasets, they shine when pre-trained on larger datasets. Similarly, larger ViT variants overtake smaller ones as the dataset grows.

1. Transformers appeared in 2017
2. They immediately took over NLP
3. Vision Transformers appeared in 2021

Question: Why did it take so long for transformers to become popular in computer vision?

Comparison of two model families:

1. BiT – large CNNs based on ResNet
2. ViT – vision transformers of various sizes

Vision Transformer (ViT)

- The original Vision Transformer models were quite small compared to the Large Language Models (LLMs) of the time
- By 2023, ViT had been scaled to 22 billion parameters with good success

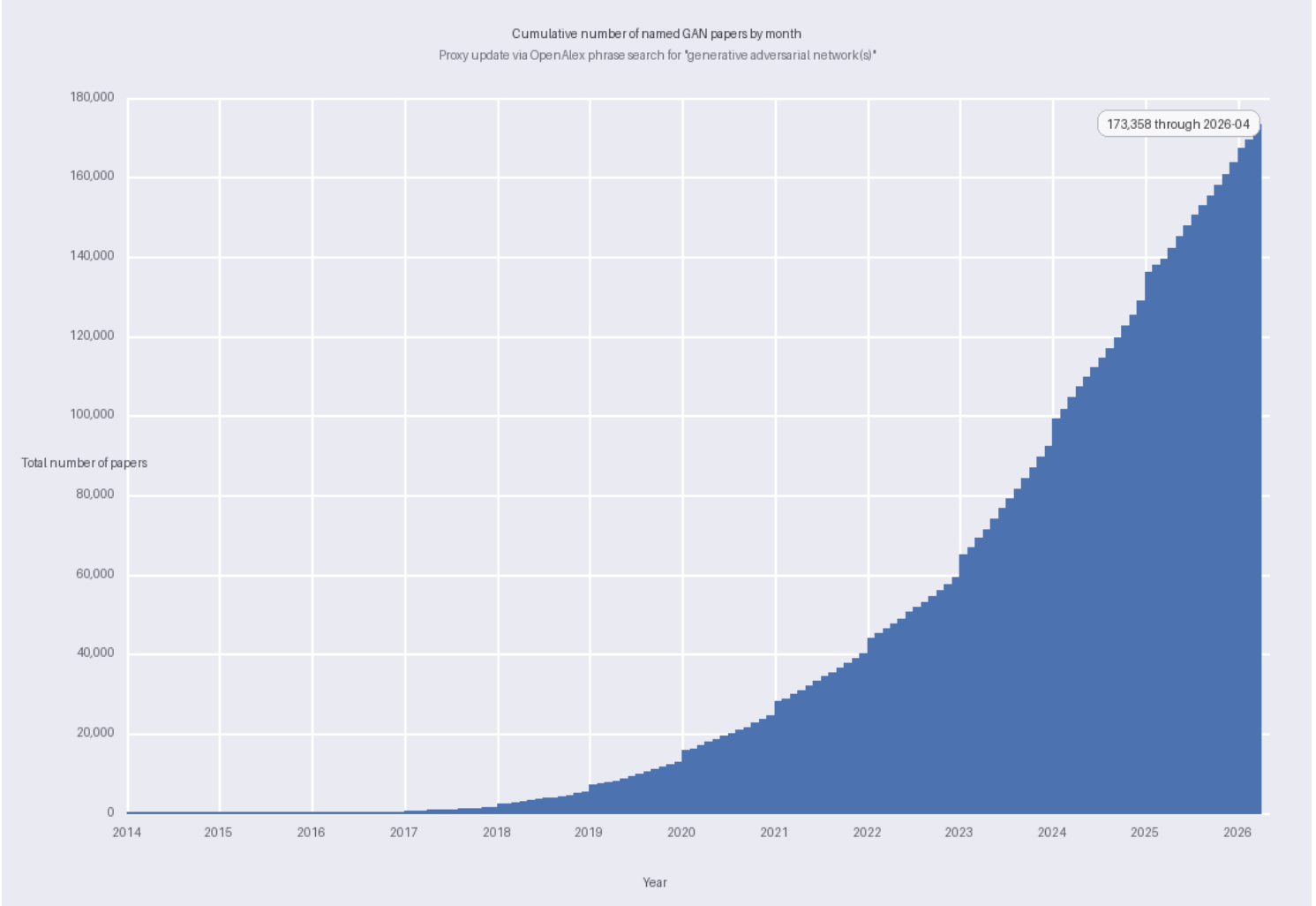
Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Table 1: Details of Vision Transformer model variants.

Table 1: ViT-22B model architecture details.

Name	**Width**	**Depth**	**MLP**	**Heads**	**Params [M]**
ViT-G	1664	48	8192	16	1843
ViT-e	1792	56	15360	16	3926
ViT-22B	6144	48	24576	48	21743

GANs Everywhere!



Latent Variable Models

Recall…

- For GANs and VAEs, we assume that there are (unknown) **latent variables** which give rise to our observations
- The **vector z** are those latent variables
- After learning a GAN or VAE, we can **interpolate** between images in latent **z** space

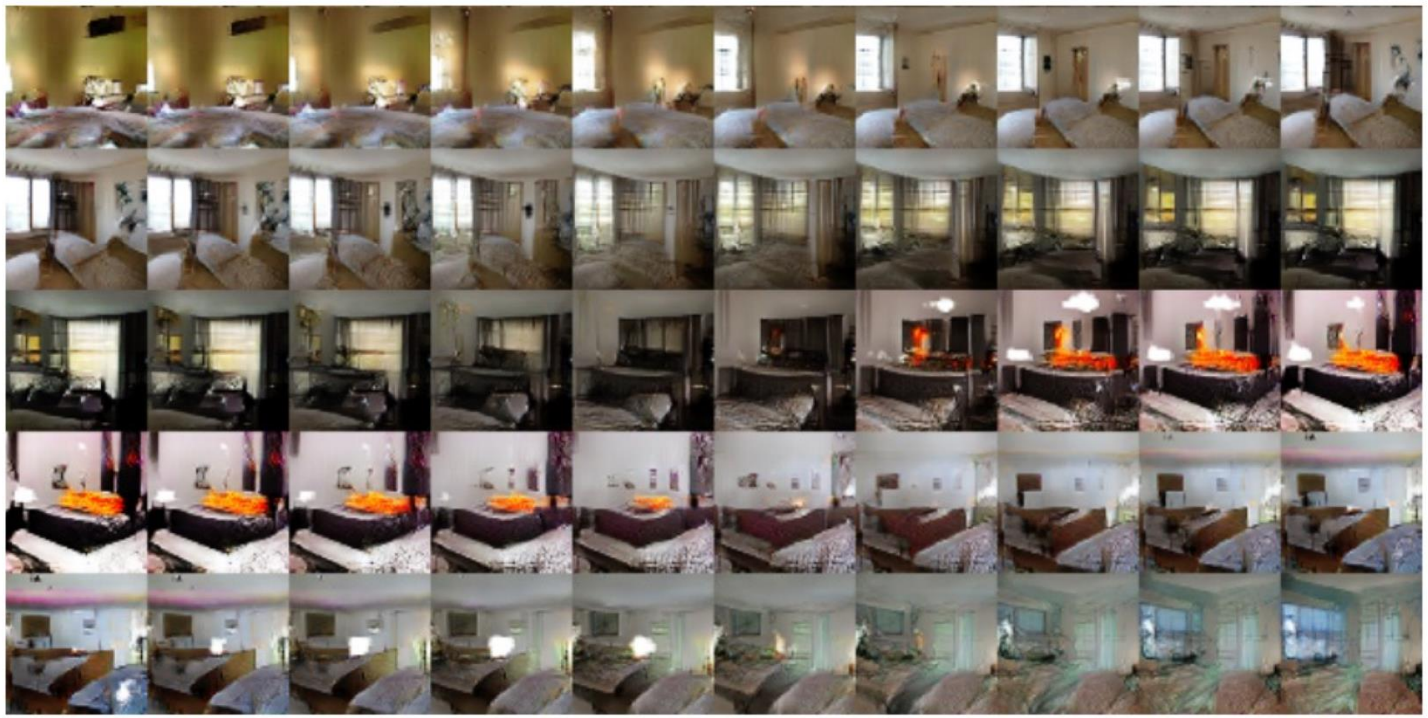


Figure 4: Top rows: Interpolation between a series of 9 random points in Z show that the space learned has smooth transitions, with every image in the space plausibly looking like a bedroom. In the 6th row, you see a room without a window slowly transforming into a room with a giant window. In the 10th row, you see what appears to be a TV slowly being transformed into a window.

Figure from Radford et al. (2016)

GAN → VAE → Diffusion

GAN → VAE → Diffusion

GANs

$p_\theta(\mathbf{z}) \sim \mathcal{N}(0, \mathbf{I})$ $p_\theta(\mathbf{x} \mid \mathbf{z}) =$ point mass on $G_\theta(\mathbf{z})$

Model:

$\mathbf{z}$ $\mathbf{x}$

$$p_\theta(\mathbf{x} \mid \mathbf{z}) = \begin{cases} 1 & \text{if } \mathbf{x} = G_\theta(\mathbf{z}) \\ 0 & \text{otherwise} \end{cases}$$

MLE?:

$$\hat{\theta} = \arg\max_\theta \sum_{i=1}^{N} \log p_\theta(\mathbf{x}^{(i)})$$ NOPE!

$$p_\theta(\mathbf{x}) = \int_{\mathbf{z}} p_\theta(\mathbf{x} \mid \mathbf{z})\, p_\theta(\mathbf{z})\, d\mathbf{z}$$

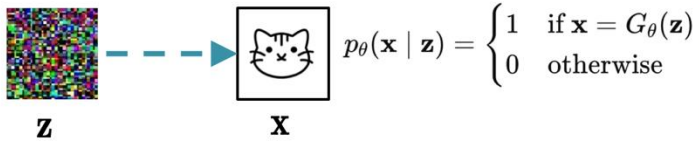
Learn: minimax game

GAN → VAE → Diffusion

GANs

$p_\theta(\mathbf{z}) \sim \mathcal{N}(0, \mathbf{I})$ $\quad p_\theta(\mathbf{x} \mid \mathbf{z}) =$ point mass on $G_\theta(\mathbf{z})$

Model:



$$p_\theta(\mathbf{x} \mid \mathbf{z}) = \begin{cases} 1 & \text{if } \mathbf{x} = G_\theta(\mathbf{z}) \\ 0 & \text{otherwise} \end{cases}$$

MLE?: $\hat{\theta} = \arg\max_\theta \sum_{i=1}^{N} \log p_\theta(\mathbf{x}^{(i)})$ NOPE!

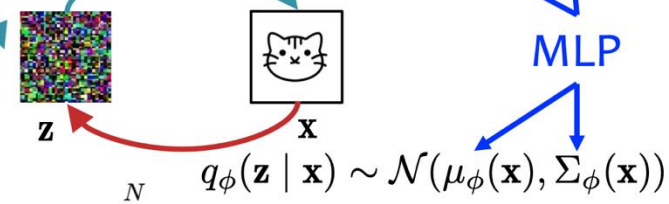
$$p_\theta(\mathbf{x}) = \int_{\mathbf{z}} p_\theta(\mathbf{x} \mid \mathbf{z})\, p_\theta(\mathbf{z})\, d\mathbf{z}$$

Learn: minimax game

VAEs

$p_\theta(\mathbf{z}) \sim \mathcal{N}(0, \mathbf{I})$ $\quad p_\theta(\mathbf{x} \mid \mathbf{z}) \sim \mathcal{N}(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z}))$

Model:



$q_\phi(\mathbf{z} \mid \mathbf{x}) \sim \mathcal{N}(\mu_\phi(\mathbf{x}), \Sigma_\phi(\mathbf{x}))$

$$h = \tanh(\mathbf{W}_1 \mathbf{z} + \mathbf{b}_1)$$
$$\mu_\theta(\mathbf{z}) = \mathbf{W}_2 h + \mathbf{b}_2$$
$$\Sigma_\theta(\mathbf{z}) = (\mathbf{W}_3 h + \mathbf{b}_3)\mathbf{I}$$

MLE?: $\hat{\theta} = \arg\max_\theta \sum_{i=1}^{N} \log p_\theta(\mathbf{x}^{(i)})$ NOPE!

$$p_\theta(\mathbf{x}) = \int_{\mathbf{z}} p_\theta(\mathbf{x} \mid \mathbf{z})\, p_\theta(\mathbf{z})\, d\mathbf{z}$$

Learn: $\theta, \phi = \arg\min_{\theta,\phi} \mathrm{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, p_\theta(\mathbf{z} \mid \mathbf{x}))$

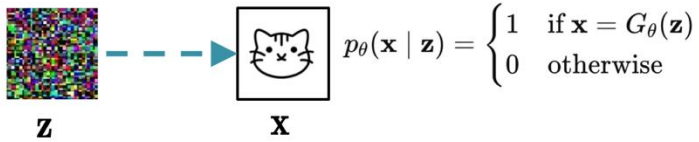
where $p_\theta(\mathbf{z} \mid \mathbf{x})$ is true posterior of $p_\theta(\mathbf{x} \mid \mathbf{z})\, p_\theta(\mathbf{z})$

GAN → VAE → Diffusion

GANs

$p_\theta(\mathbf{z}) \sim \mathcal{N}(0, \mathbf{I})$ $\quad p_\theta(\mathbf{x} \mid \mathbf{z}) =$ point mass on $G_\theta(\mathbf{z})$

Model:



$$p_\theta(\mathbf{x} \mid \mathbf{z}) = \begin{cases} 1 & \text{if } \mathbf{x} = G_\theta(\mathbf{z}) \\ 0 & \text{otherwise} \end{cases}$$

MLE?: $\hat{\theta} = \arg\max_\theta \sum_{i=1}^{N} \log p_\theta(\mathbf{x}^{(i)})$ NOPE!

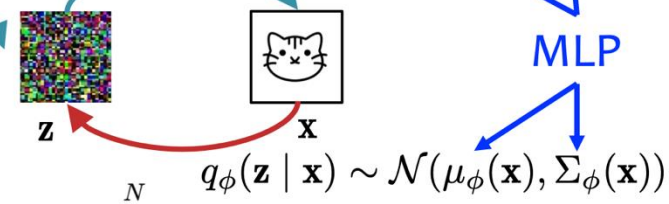
$$p_\theta(\mathbf{x}) = \int_{\mathbf{z}} p_\theta(\mathbf{x} \mid \mathbf{z})\, p_\theta(\mathbf{z})\, d\mathbf{z}$$

Learn: minimax game

VAEs

$p_\theta(\mathbf{z}) \sim \mathcal{N}(0, \mathbf{I})$ $\quad p_\theta(\mathbf{x} \mid \mathbf{z}) \sim \mathcal{N}(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z}))$

Model:



$$h = \tanh(\mathbf{W}_1\mathbf{z} + \mathbf{b}_1)$$
$$\mu_\theta(\mathbf{z}) = \mathbf{W}_2 h + \mathbf{b}_2$$
$$\Sigma_\theta(\mathbf{z}) = (\mathbf{W}_3 h + \mathbf{b}_3)\mathbf{I}$$

$q_\phi(\mathbf{z} \mid \mathbf{x}) \sim \mathcal{N}(\mu_\phi(\mathbf{x}), \Sigma_\phi(\mathbf{x}))$

MLE?: $\hat{\theta} = \arg\max_\theta \sum_{i=1}^{N} \log p_\theta(\mathbf{x}^{(i)})$ NOPE!

$$p_\theta(\mathbf{x}) = \int_{\mathbf{z}} p_\theta(\mathbf{x} \mid \mathbf{z})\, p_\theta(\mathbf{z})\, d\mathbf{z}$$

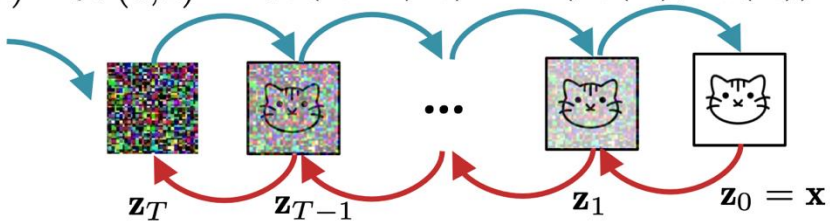
Learn: $\theta, \phi = \arg\min_{\theta,\phi} \mathrm{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, p_\theta(\mathbf{z} \mid \mathbf{x}))$

where $p_\theta(\mathbf{z} \mid \mathbf{x})$ is true posterior of $p_\theta(\mathbf{x} \mid \mathbf{z})\, p_\theta(\mathbf{z})$

Diffusion

$p_\theta(\mathbf{z}) \sim \mathcal{N}(0, \mathbf{I})$ $\quad p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t) \sim \mathcal{N}(\mu_\theta(\mathbf{z}_t), \Sigma_\theta(\mathbf{z}_t))$

Model:



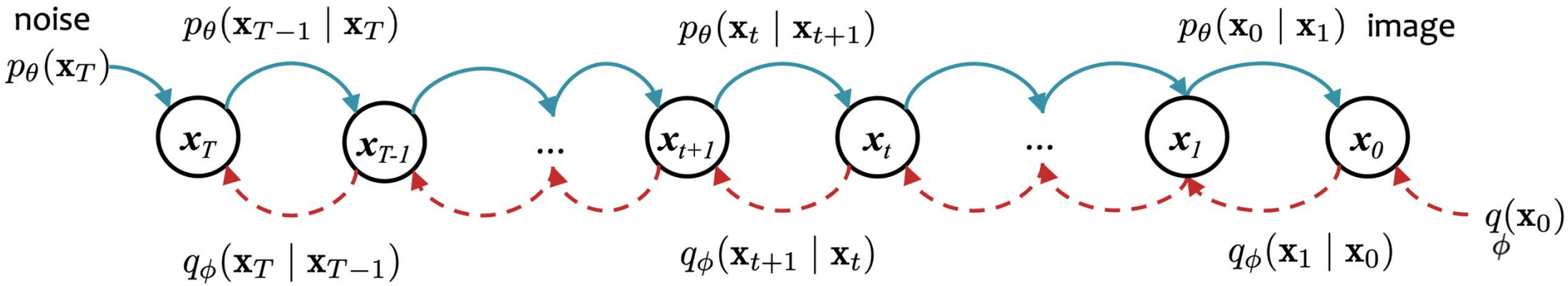
$q_\phi(\mathbf{z}_t \mid \mathbf{z}_{t-1}) \sim \mathcal{N}(\mu_\phi(\mathbf{z}_{t-1}), \Sigma_\phi(\mathbf{z}_{t-1}))$ $\quad q_\phi(\mathbf{z}_0) =$ data dist.

MLE?: NOPE!

Learn: $\theta = \arg\min_\theta \mathrm{KL}(q_\phi(\mathbf{z}_{1:T} \mid \mathbf{x}) \,\|\, p_\theta(\mathbf{z}_{1:T} \mid \mathbf{x}))$, $\quad \phi$ chosen ahead of time

where $p_\theta(\mathbf{z}_{1:T} \mid \mathbf{x})$ is true posterior of $p_\theta(\mathbf{x} \mid \mathbf{z}_1) \cdots p_\theta(\mathbf{z}_{T-1} \mid \mathbf{z}_T)\, p_\theta(\mathbf{z}_T)$

Diffusion Model



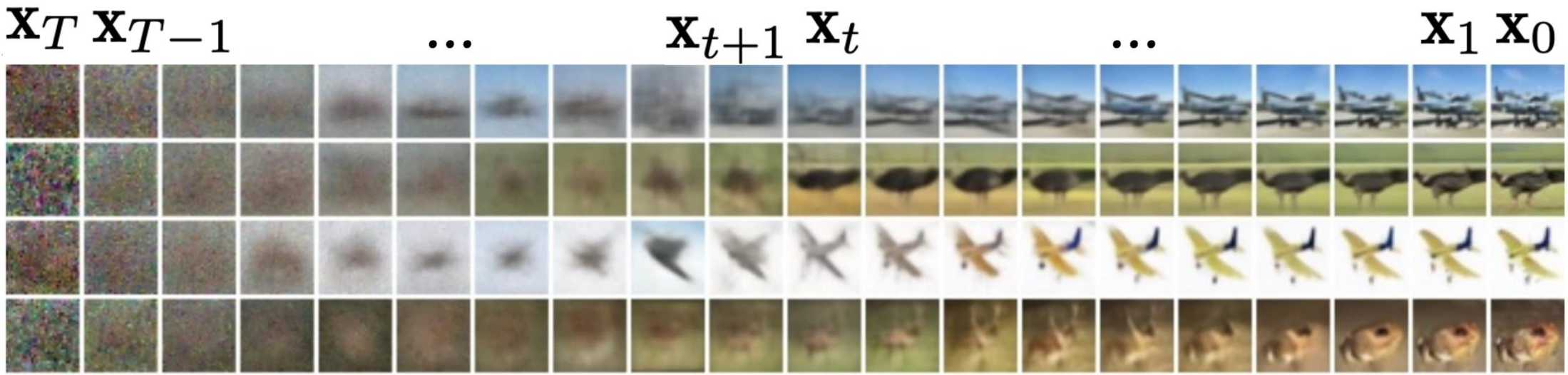
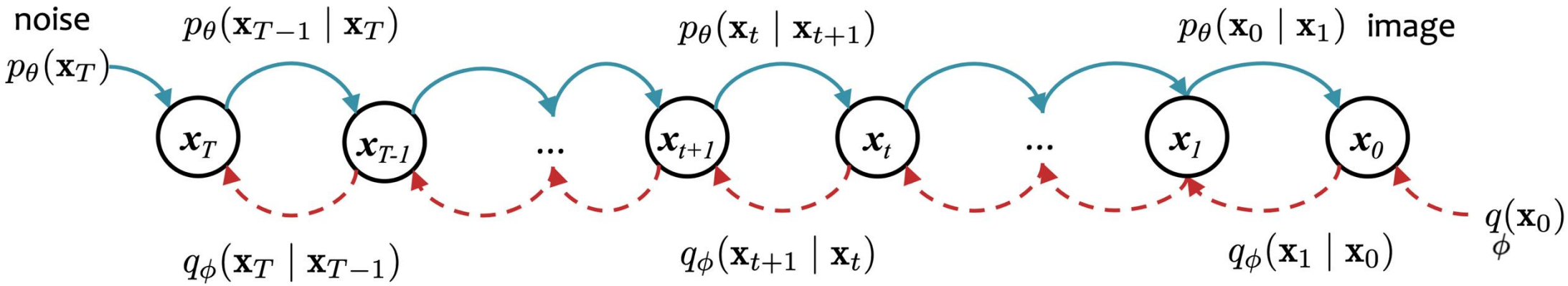
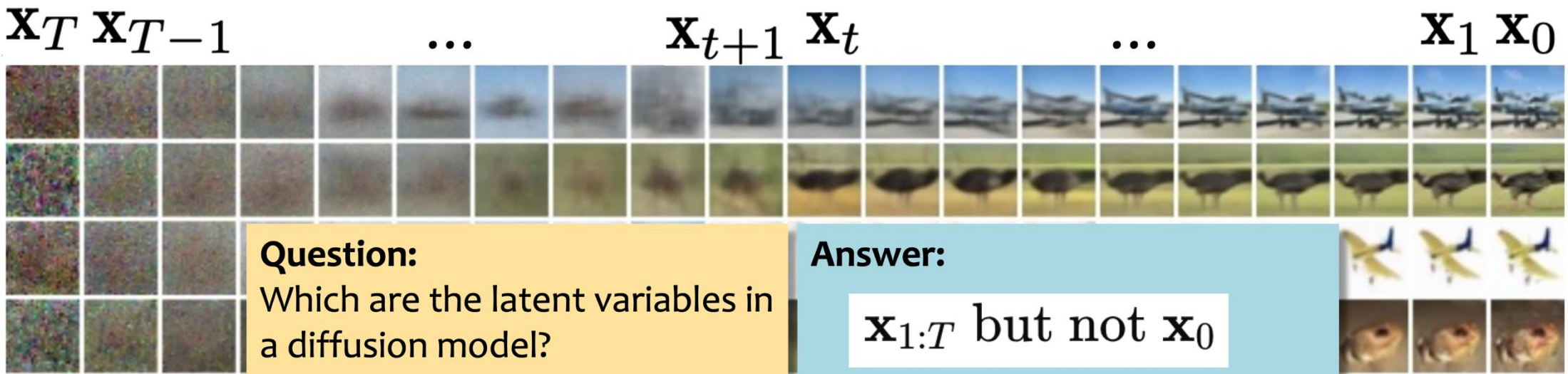


Figure from Ho et al. (2020)

Diffusion Model





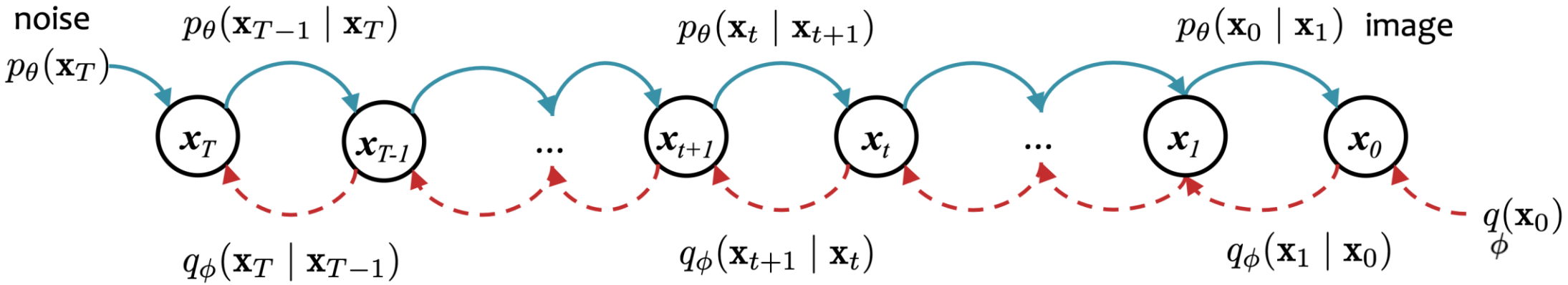
Question:
Which are the latent variables in a diffusion model?

Answer:

$\mathbf{x}_{1:T}$ but not $\mathbf{x}_0$

Figure from Ho et al. (2020)

Denoising Diffusion Probabilistic Model (DDPM)



Forward Process (*adds* noise to image):

$$q_\phi(\mathbf{x}_{0:T}) = q_\phi(\mathbf{x}_0) \prod_{t=1}^{T} q_\phi(\mathbf{x}_t \mid \mathbf{x}_{t-1})$$

$q_\phi(\mathbf{x}_0)$ = data distribution

$$q_\phi(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \sim \mathcal{N}(\sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I})$$

(Learned) Reverse Process (*removes* noise):

$$p_\theta(\mathbf{x}_{0:T}) = p_\theta(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$$

$$p_\theta(\mathbf{x}_T) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \sim \mathcal{N}(\mu_\theta(\mathbf{x}_t, t), \mathbf{\Sigma}_\theta(\mathbf{x}_t, t))$$

I want to know more about generative models!

- Take 10-423/623 next semester to find out!

Poll:

Final Poll

Final poll (pick one)

Which statement is FALSE?

A. A VAE adds a KL term to encourage a smoother, more sampleable latent space.

B. In a GAN, the generator learns by trying to fool the discriminator.

C. Diffusion models generate samples by starting from noise and iteratively denoising.

D. Vision transformers process images by splitting them into patches and treating patches like tokens.

E. In diffusion models, the forward process gradually removes noise from the data.