10-301/601: Introduction to Machine Learning Lecture 32 – Generative Models for Vision

Henry Chai

6/16/25

Front Matter

- Announcements
  - HW8 released 6/13, due 6/17 (tomorrow) at 11:59 PM
  - Final on 6/20 (next Friday) at **8:30 AM** in BH A36 (here!)
    - **We will not use the full 3-hour window**
    - All topics from Lectures 17 to 30 are in-scope
    - **The final is *not* cumulative:** pre-midterm content may be referenced but will not be the *primary* focus of any question
    - You are allowed to bring one letter-size sheet of notes; you may put *whatever* you want on *both sides*

Image Generation

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

Image Generation

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

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

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

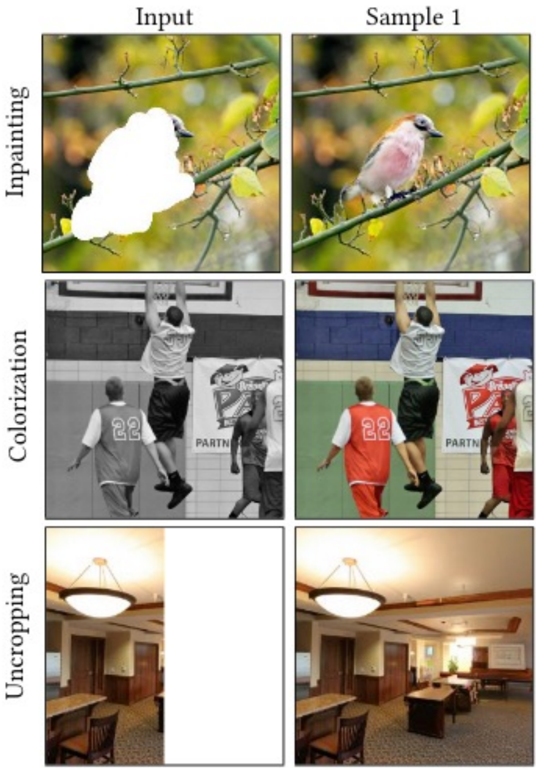
Image Generation

- 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.

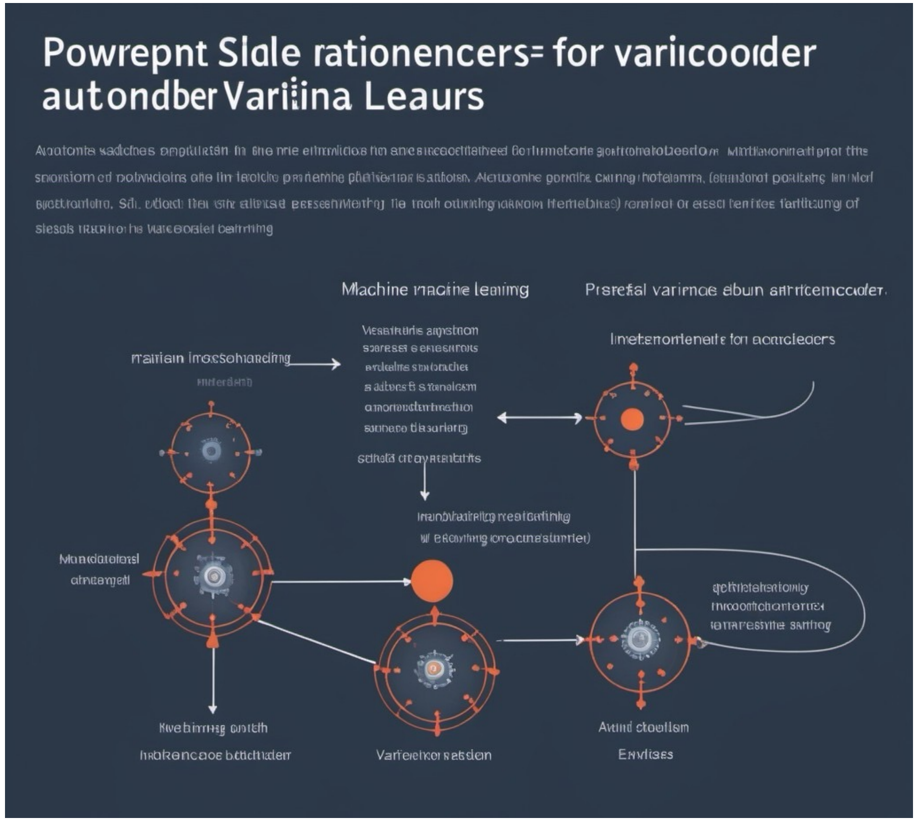
- Given a text description, sample an image that depicts the prompt

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

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

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

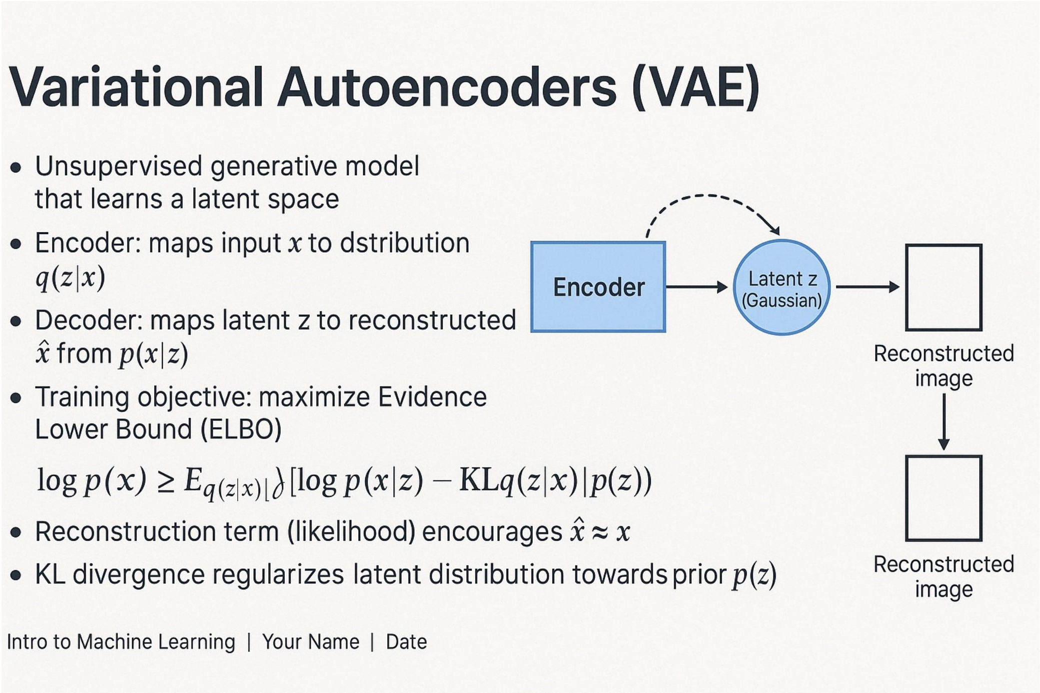


- Class-conditional generation
- Super resolution
- Image Editing
- Style transfer
- **Text-to-image (TTI) generation**

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

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



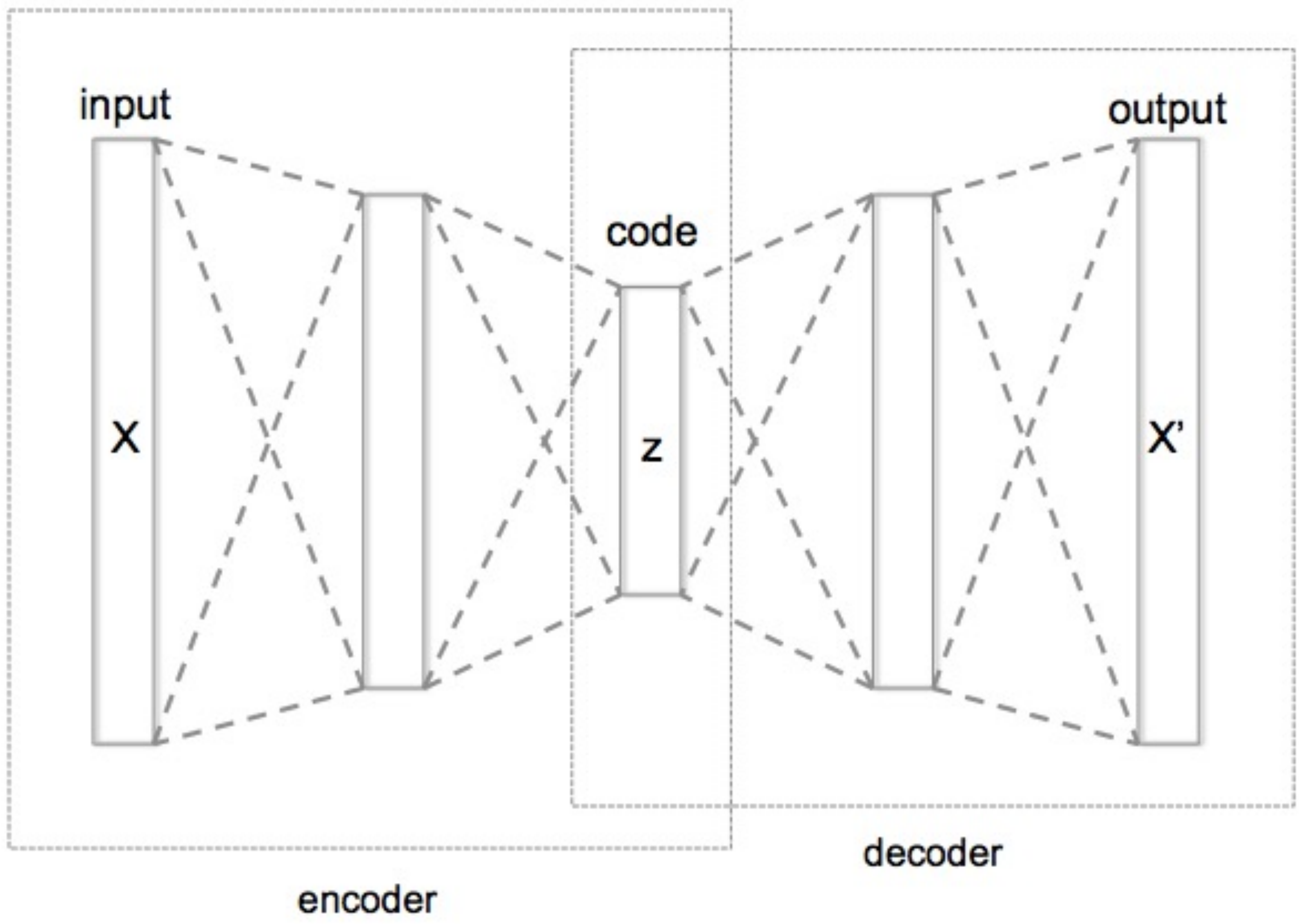
- Class-conditional generation
- Super resolution
- Image Editing
- Style transfer
- **Text-to-image (TTI) generation**

Source: https://chatgpt.com/ (4/18/25)

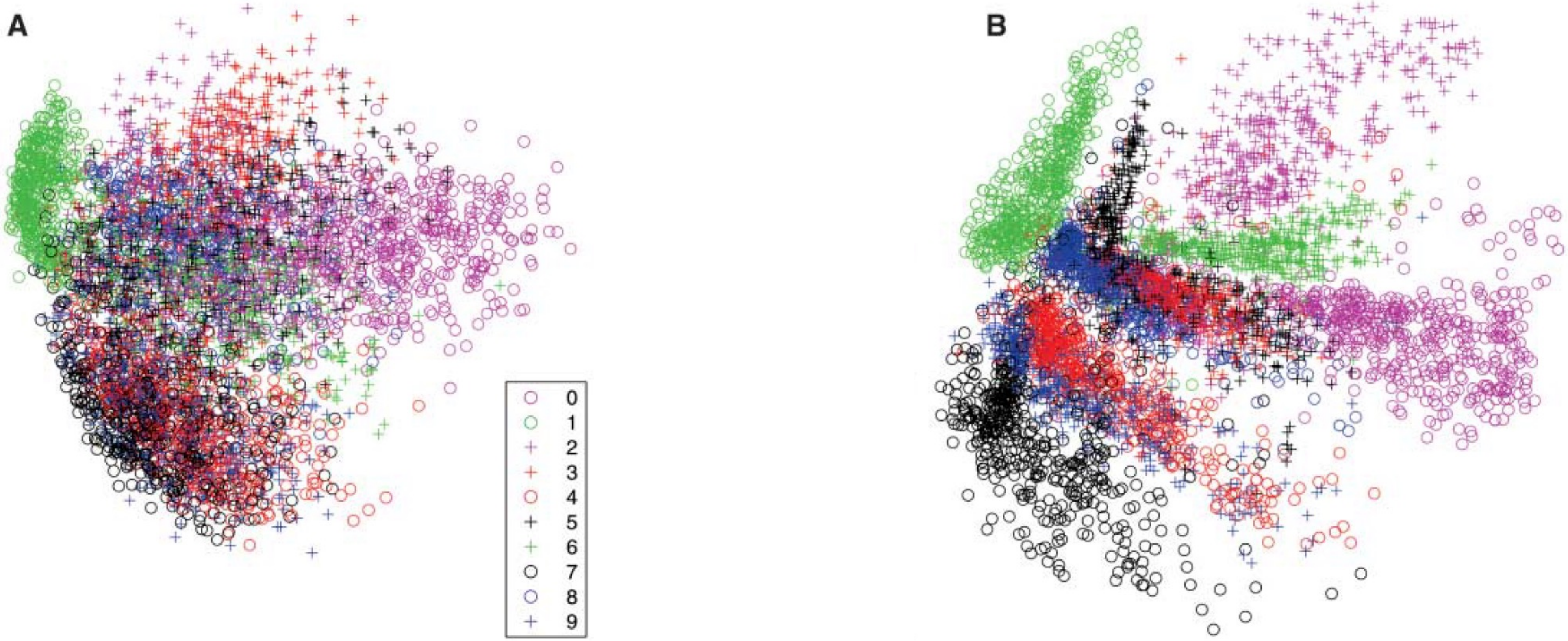
Image Generation

- 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: Deep Autoencoders

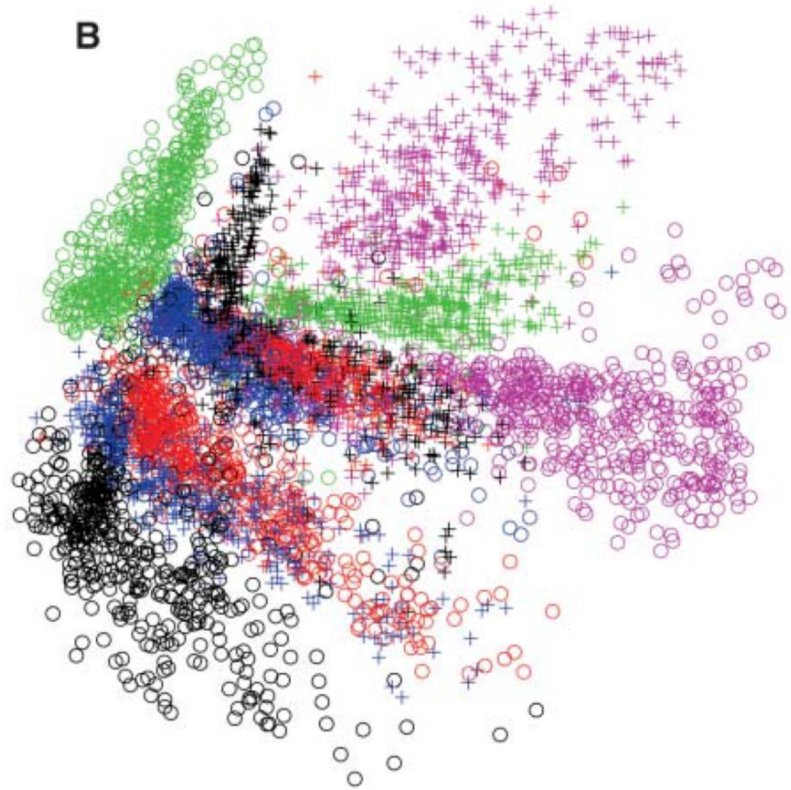


Source: https://en.wikipedia.org/wiki/Autoencoder#/media/File:Autoencoder_structure.png



PCA (A) vs. Autoencoders (B) (Hinton and Salakhutdinov, 2006)

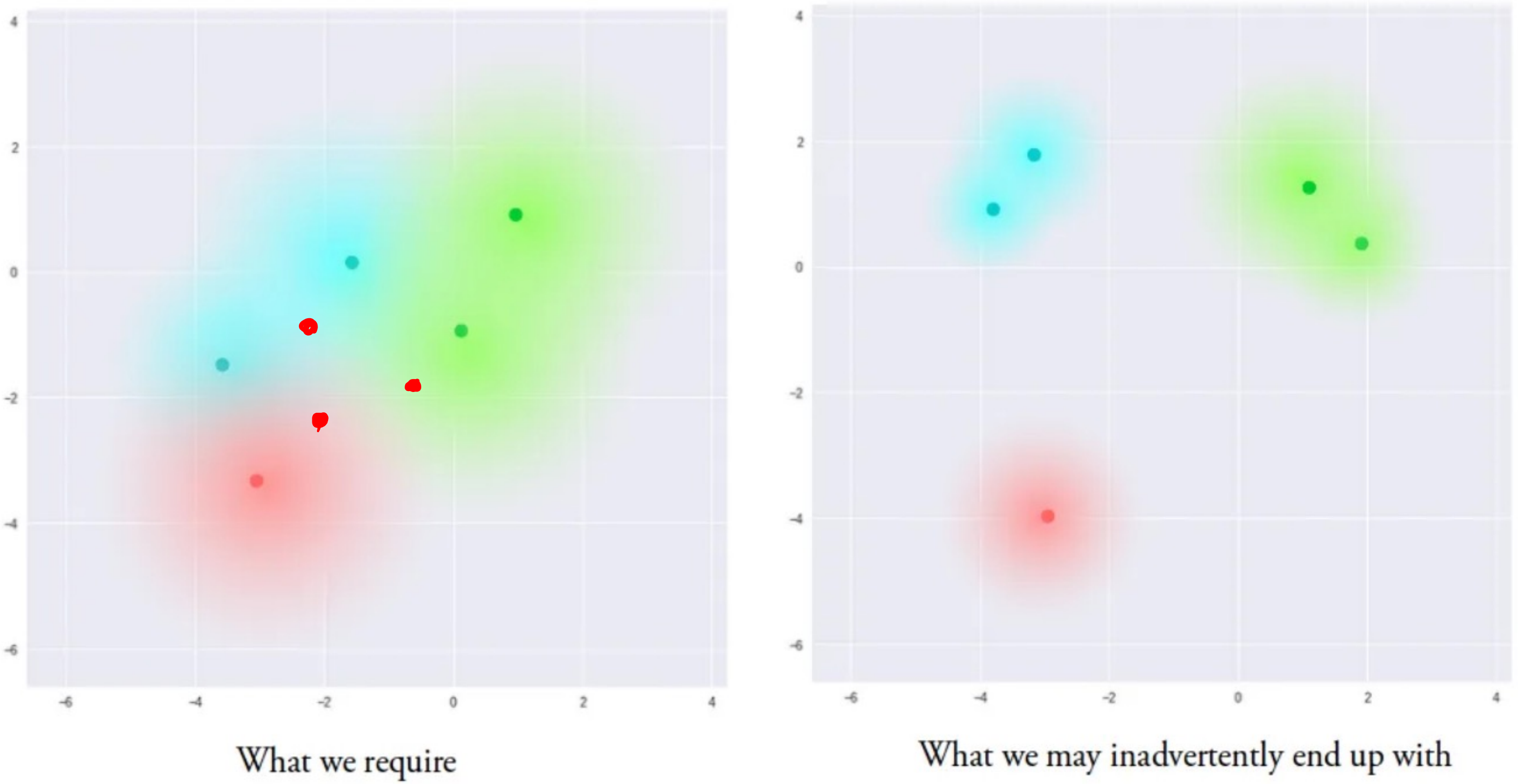
Source: https://www.science.org/doi/10.1126/science.1127647



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

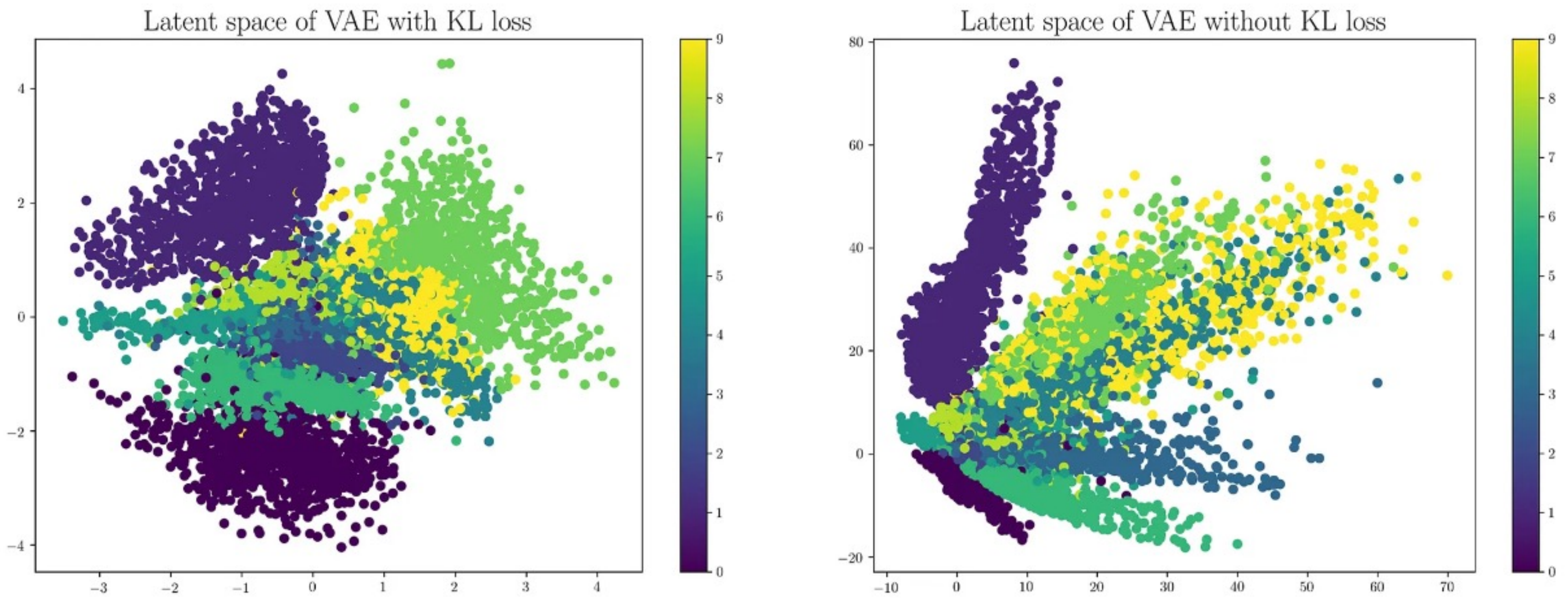
Autoencoder Latent Space

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



Autoencoder Latent Space

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



Variational Autoencoder Latent Space

Source: https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2

Variational Autoencoder: Network Perspective

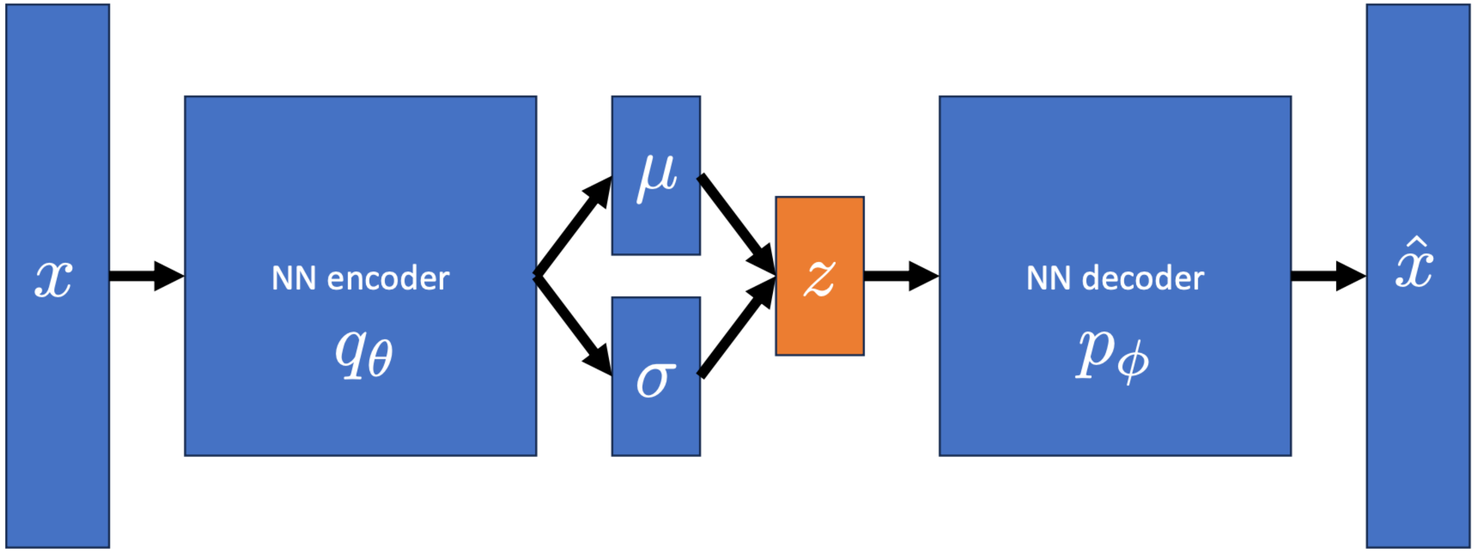
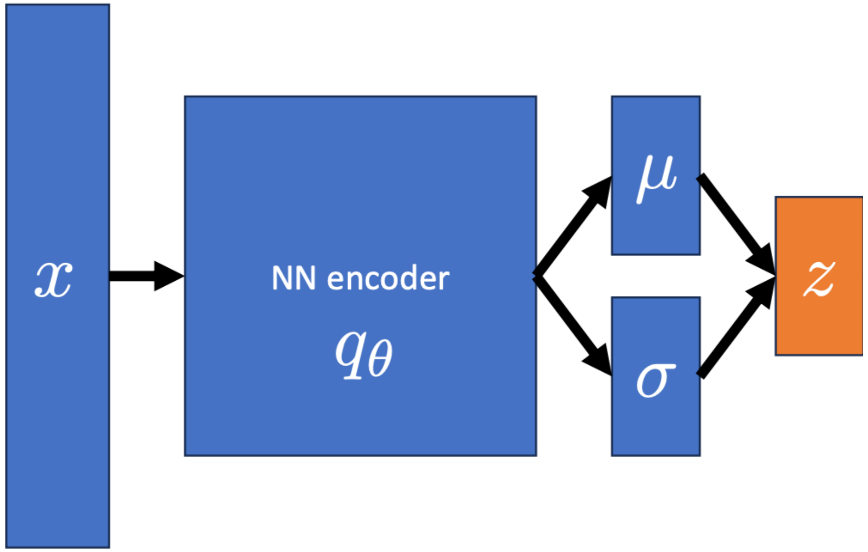


Figure courtesy of Zack Lipton

Variational Autoencoder: Network Perspective

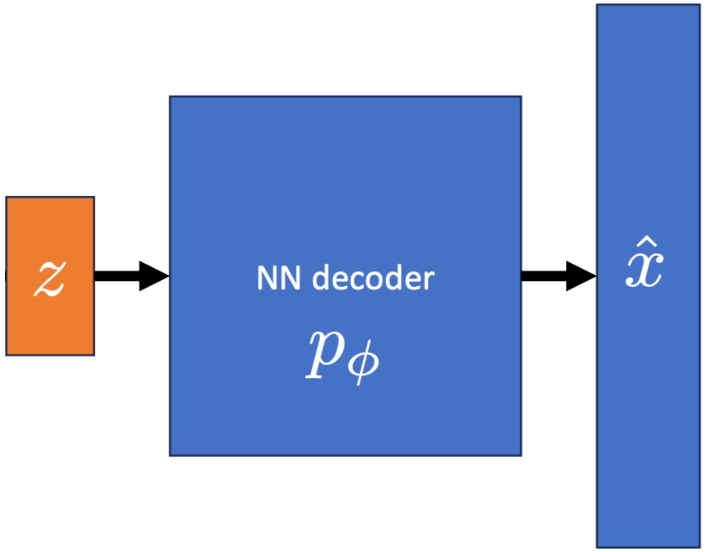


- Encoder learns a mean vector and a (diagonal) covariance matrix for each input
- These are used to *sample* a latent representation e.g.,

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

Figure courtesy of Zack Lipton

Variational Autoencoder: Network Perspective

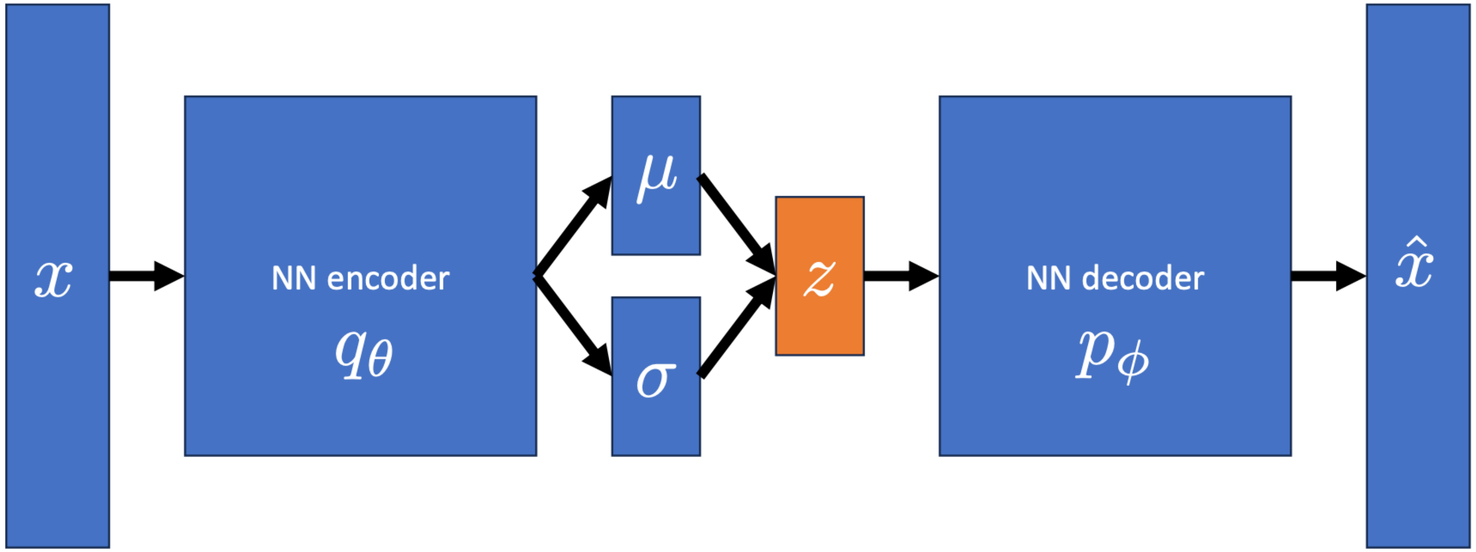


- Decoder tries to minimize the reconstruction error *in expectation* between $\boldsymbol{x}^{(i)}$ and a sample from another (conditional) distribution e.g.,

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

Figure courtesy of Zack Lipton

Variational Autoencoder: Network Perspective



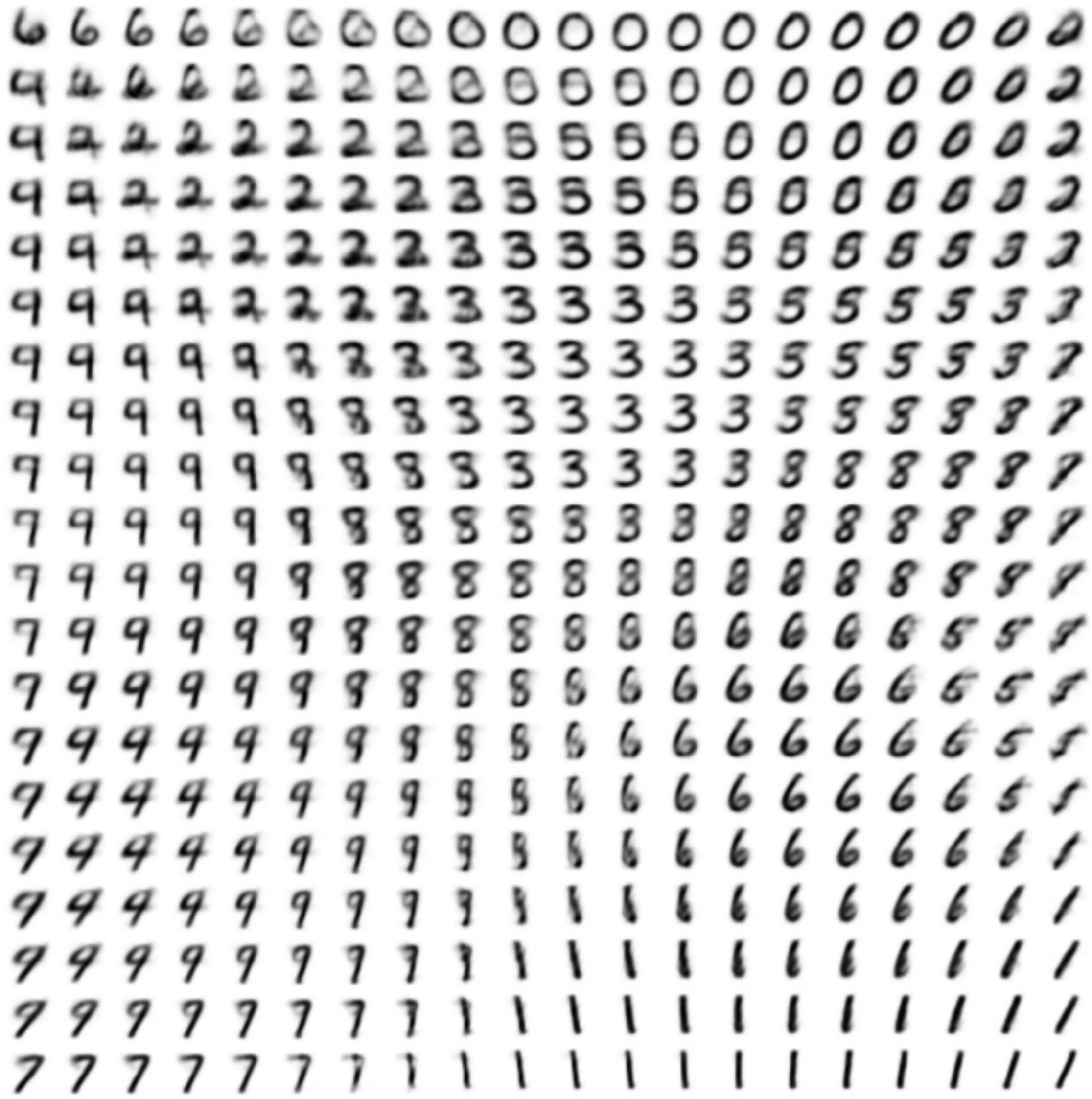
- Objective: minimize the expected reconstruction error plus a *regularizer* that encourages a dense latent space

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

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

Figure courtesy of Zack Lipton

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

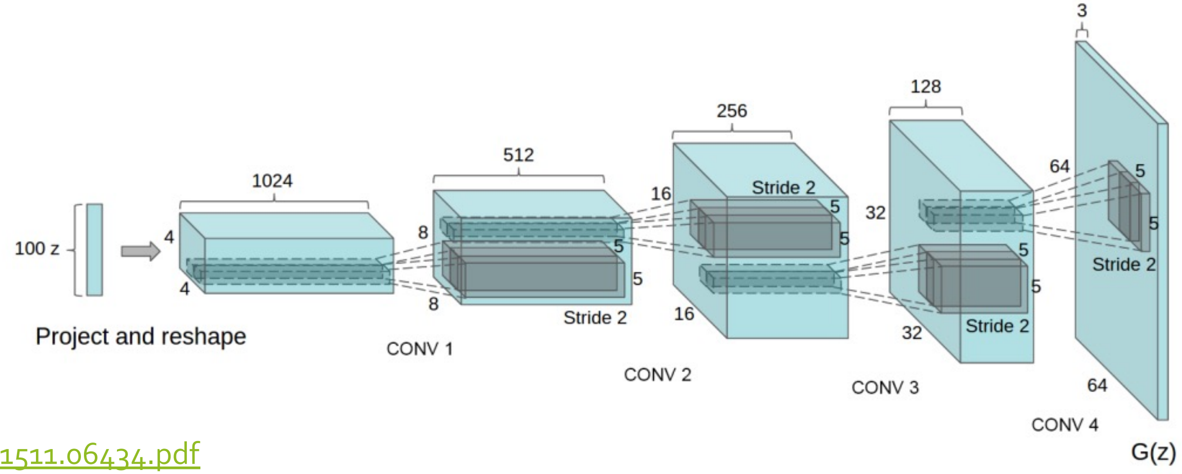
Source: MNIST

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)

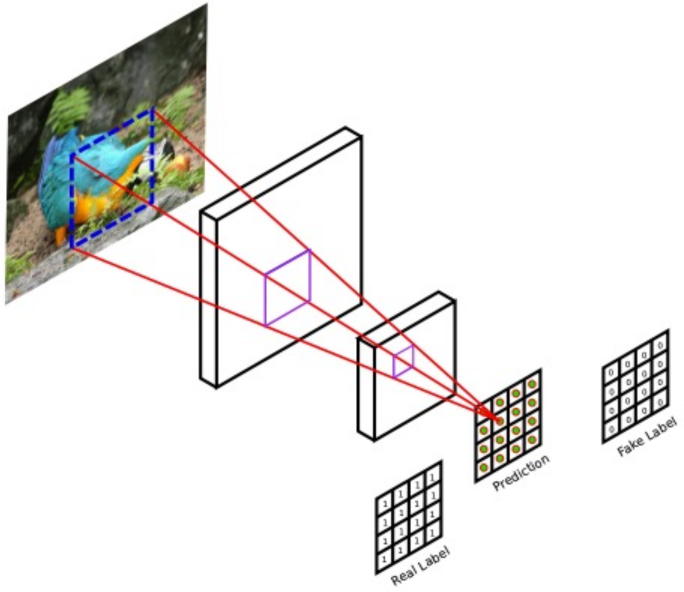
- A GAN consists of two (deterministic) models:
  - a **generator** that takes a vector of random noise as input, and generates an image
- Example generator: DCGAN
  - An inverted CNN with four *fractionally-strided* convolution layers that grow the size of the image from layer to layer; final layer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



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

Generative Adversarial Networks (GANs)

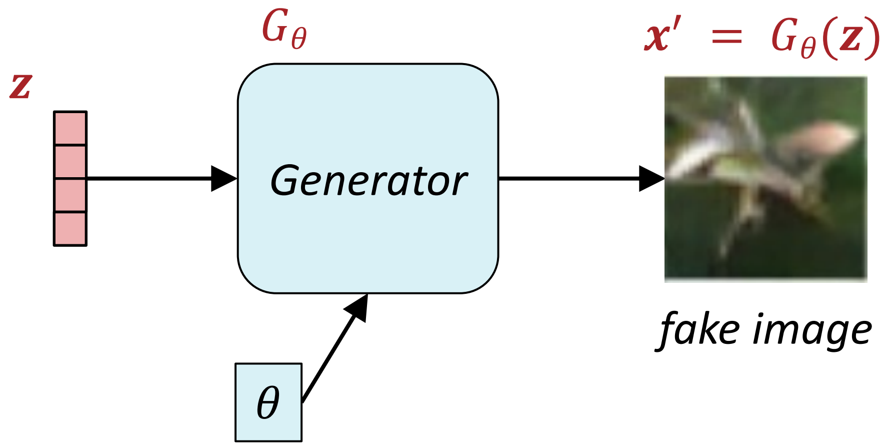
- A GAN consists of two (deterministic) models:
  - a **generator** that takes a vector of random noise as input, and generates an image
  - a **discriminator** that takes in an image classifies whether it is real (label = 1) or fake (label = 0)
- Example discriminator: PatchGAN
  - Traditional CNN that 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

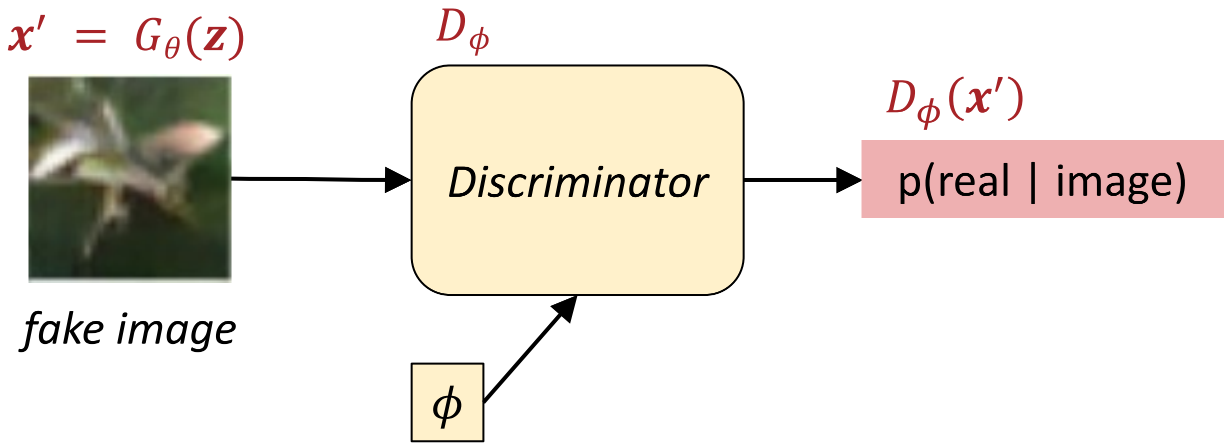
Generative Adversarial Networks (GANs): Training

- 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



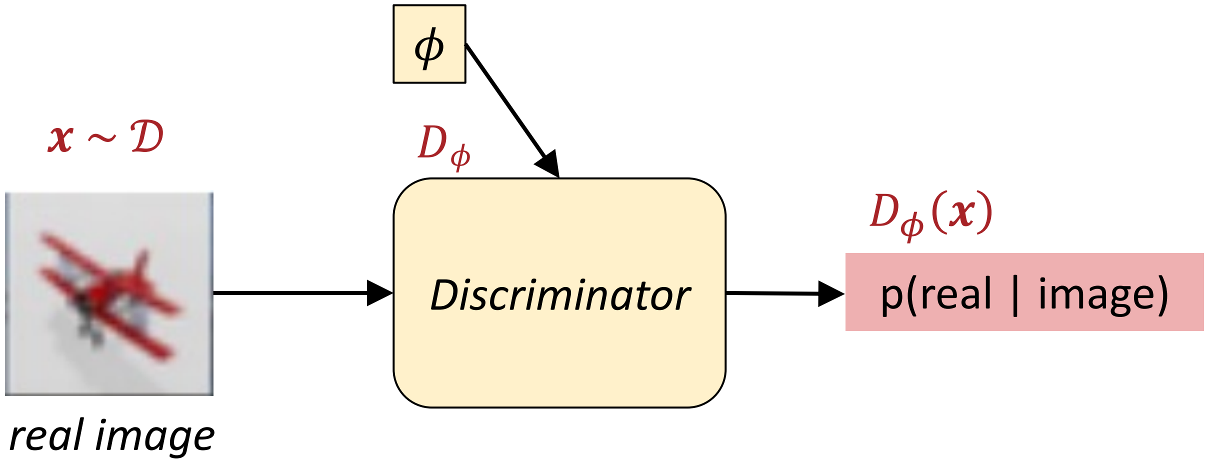
GANs: Architecture

Figure courtesy of Matt Gormley



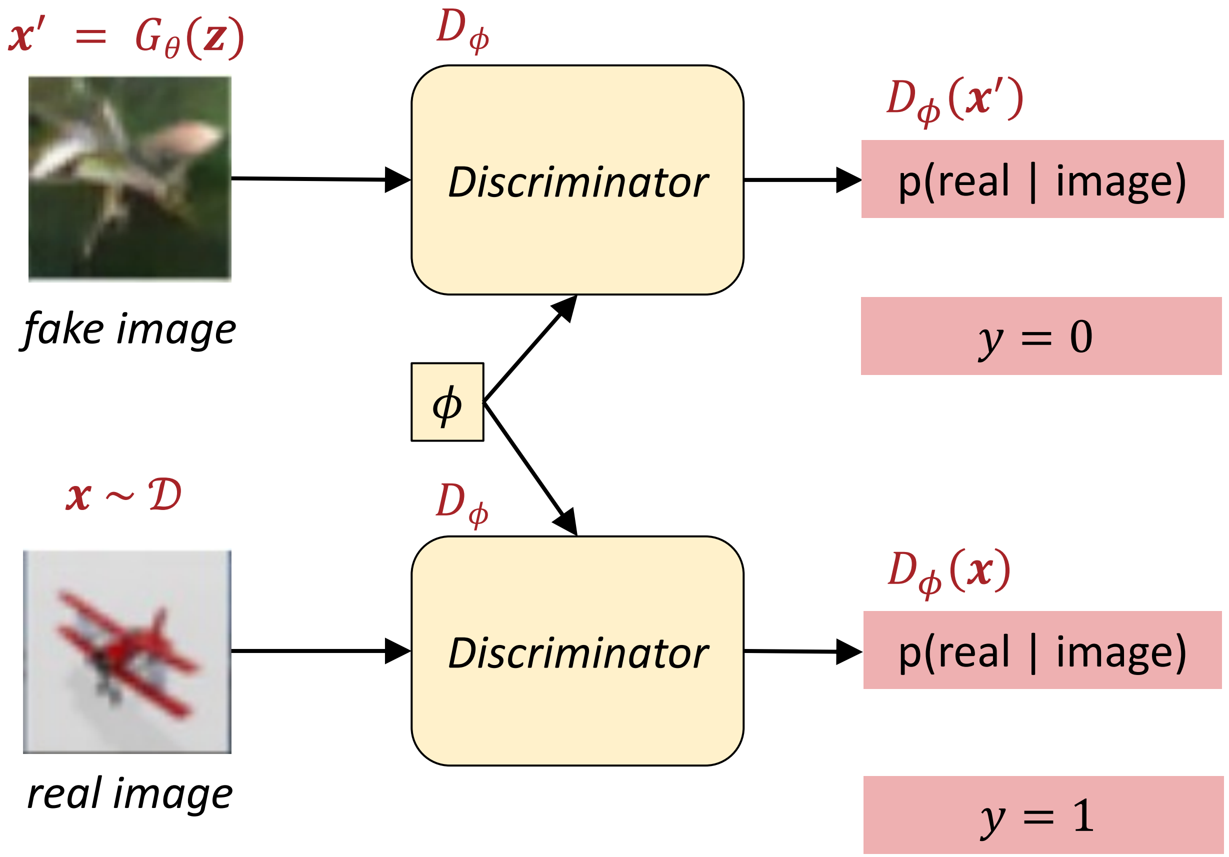
GANs: Architecture

Figure courtesy of Matt Gormley



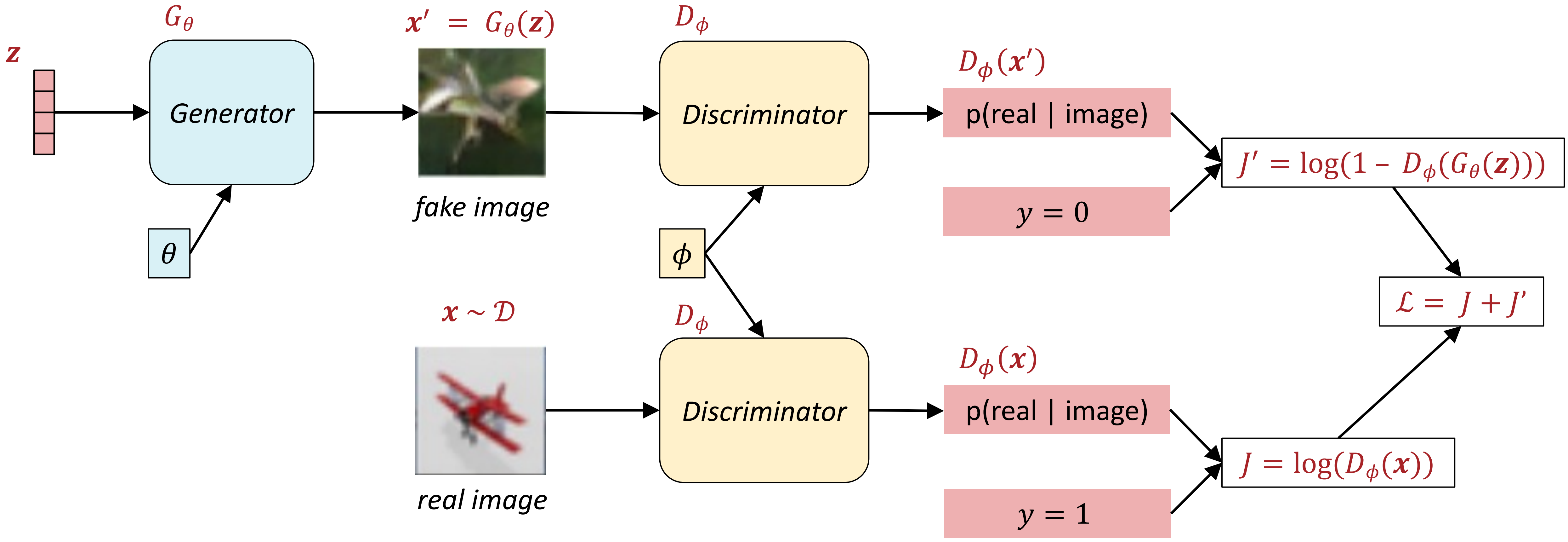
GANs: Architecture

Figure courtesy of Matt Gormley



GANs: Architecture

Figure courtesy of Matt Gormley



GANs: Architecture

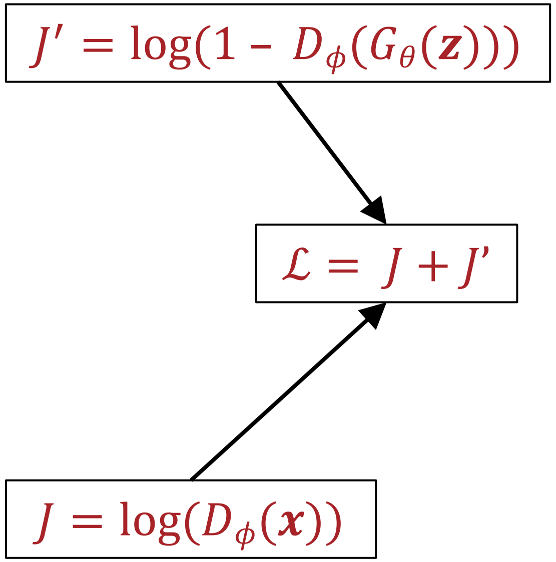
Figure courtesy of Matt Gormley

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

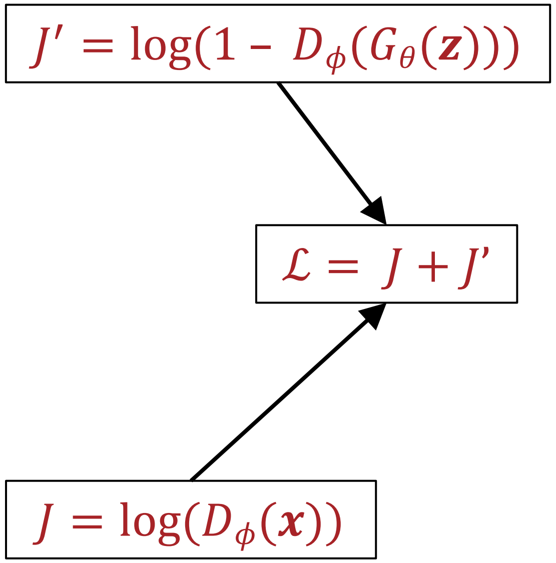


GANs: Architecture

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

GANs: Training

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**
  for k steps **do**
  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  • Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

  end for
  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Update the generator by descending its stochastic gradient:

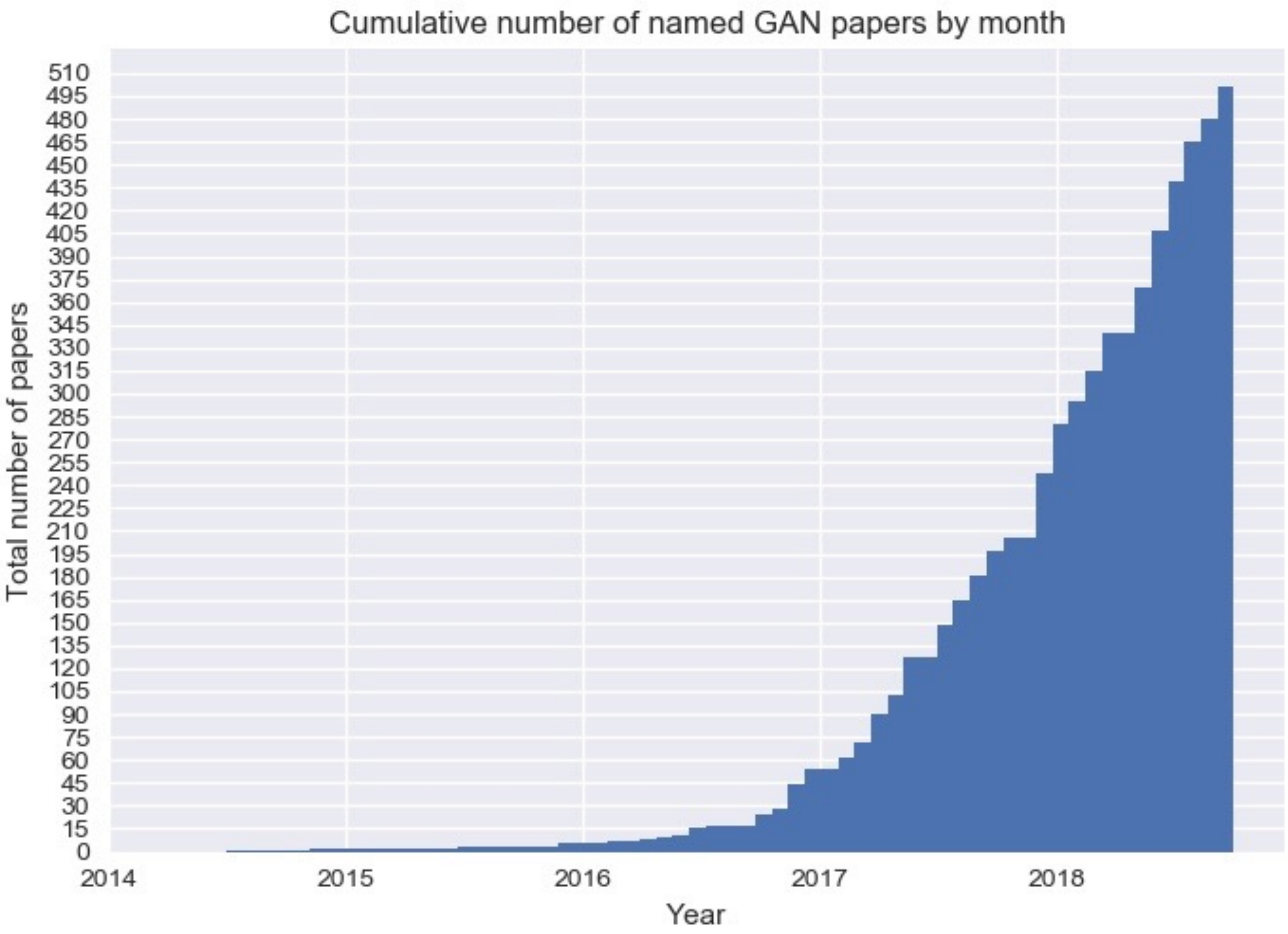
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

- Optimization is like block coordinate descent but instead of exact optimization, we take a step of mini-batch SGD

Source: https://arxiv.org/pdf/1406.2661.pdf

GANs Everywhere!



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

The rise of vision transformers and diffusion models

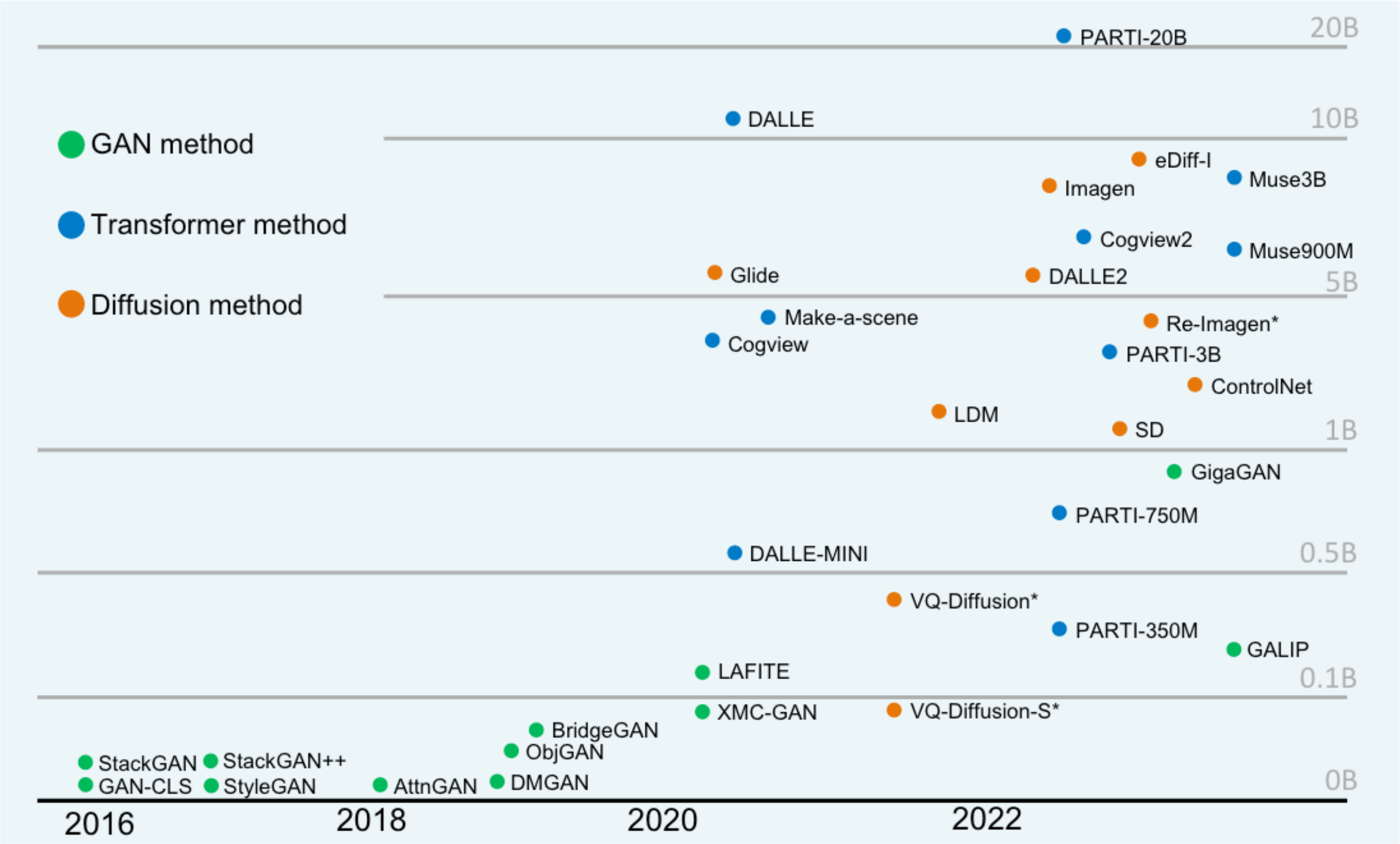


Fig. 5. Timeline of TTI model development, where green dots are GAN TTI models, blue dots are autoregressive Transformers and orange dots are Diffusion TTI models. Models are separated by their parameter, which are in general counted for all their components. Models with asterisk are calculated without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

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

But wait, what the heck are "vision transformers" and "diffusion"?

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

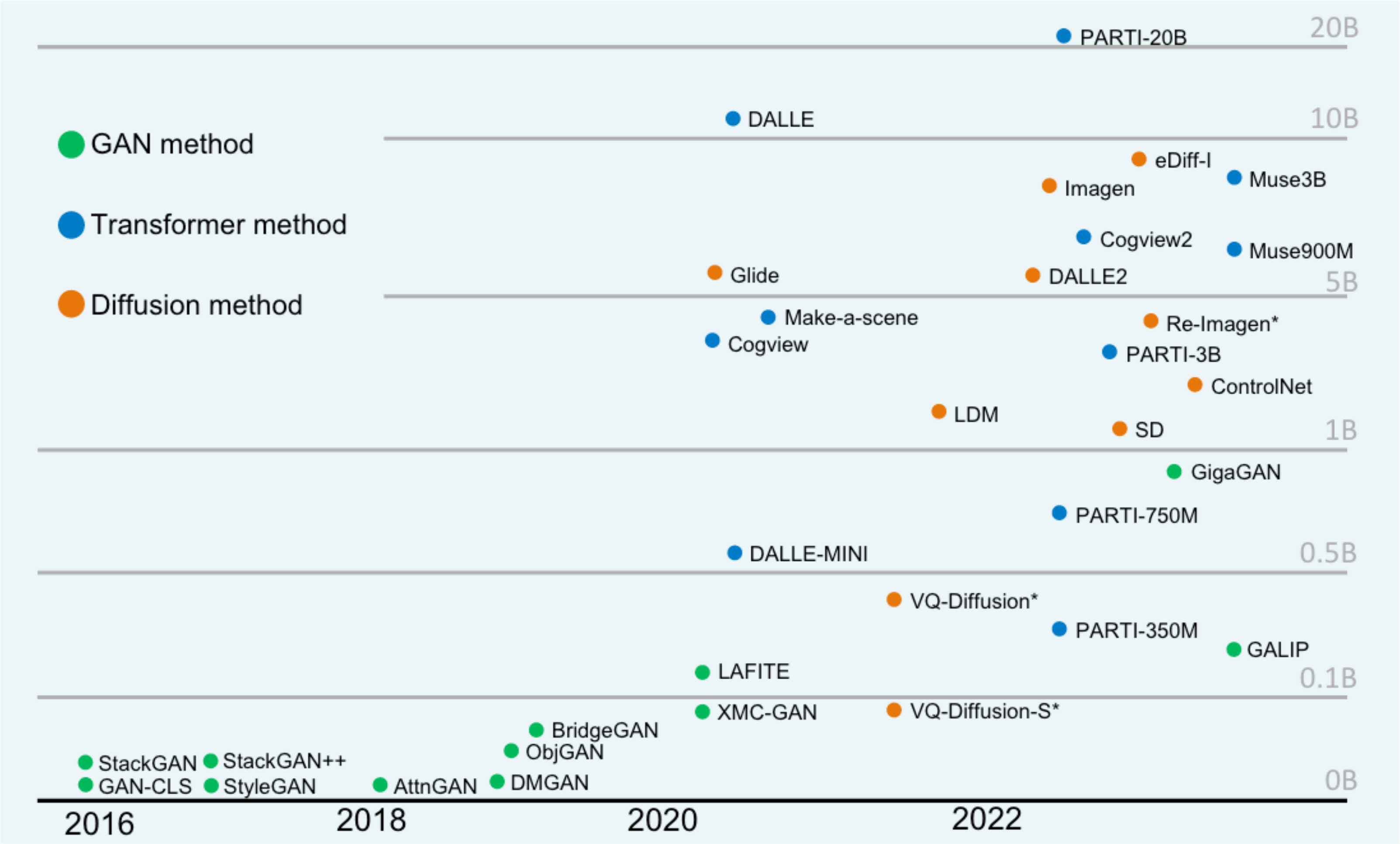


Fig. 5. Timeline of TTI model development, where green dots are GAN TTI models, blue dots are autoregressive Transformers and orange dots are Diffusion TTI models. Models are separated by their parameter, which are in general counted for all their components. Models with asterisk are calculated without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

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