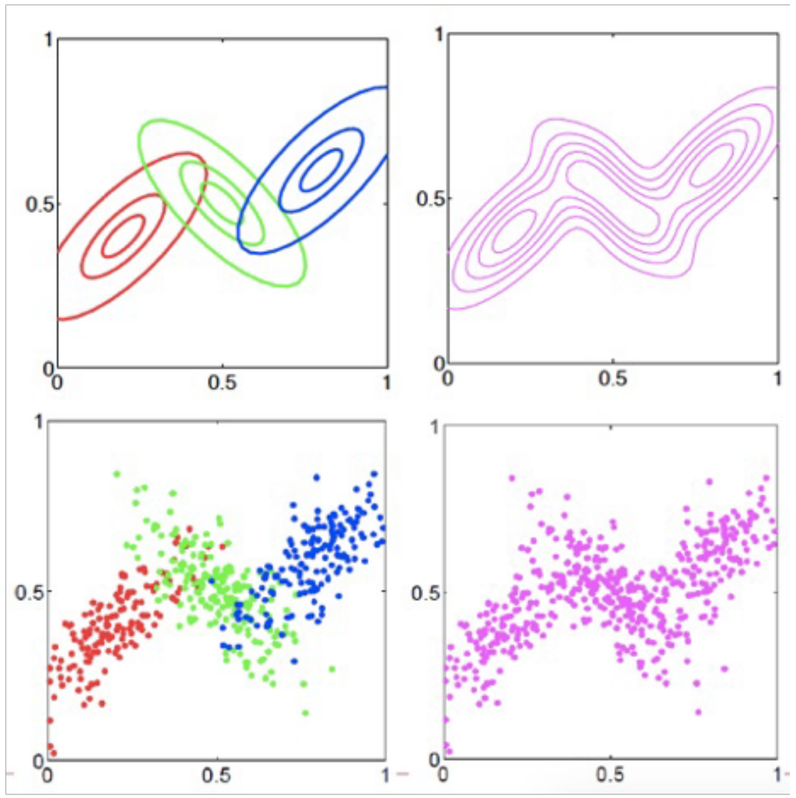
19. Generative Models

EECE454 Introduction to Machine Learning Systems

2023 Fall, Jaeho Lee

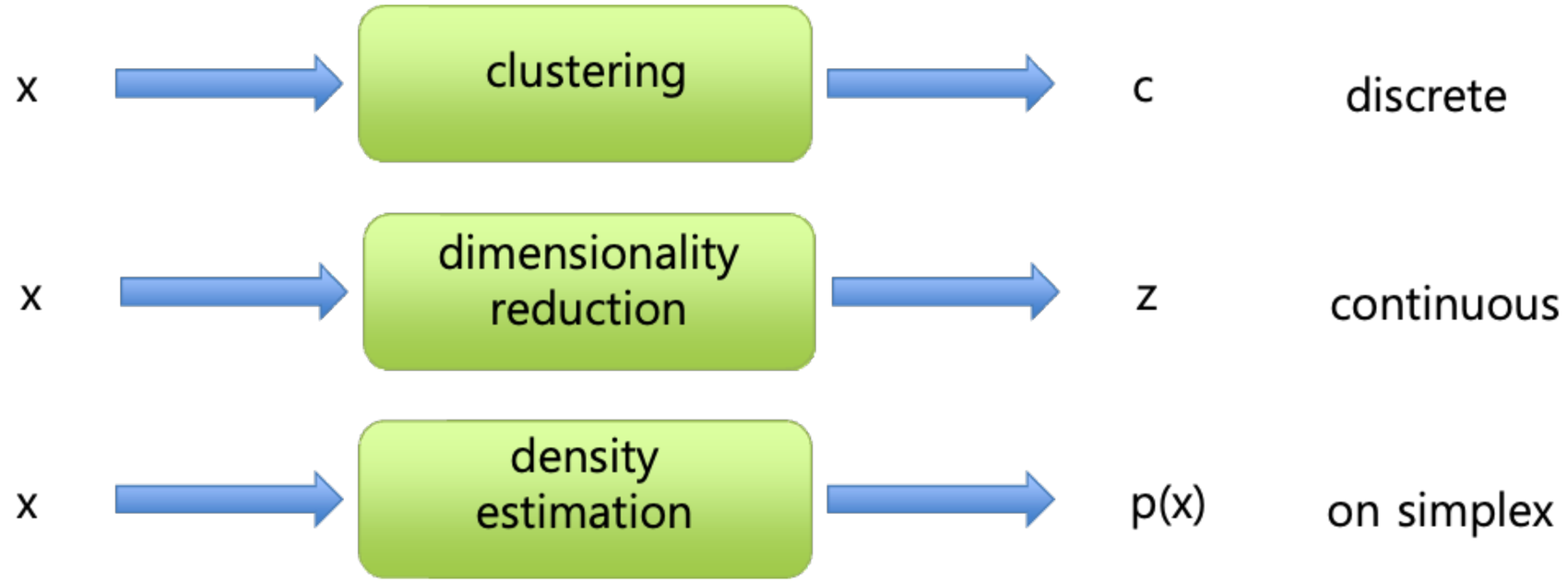
Recap

- **GMM.** We approximated a data-generating distribution from the data
  - An elementary ***generative modeling*** (or density estimation)

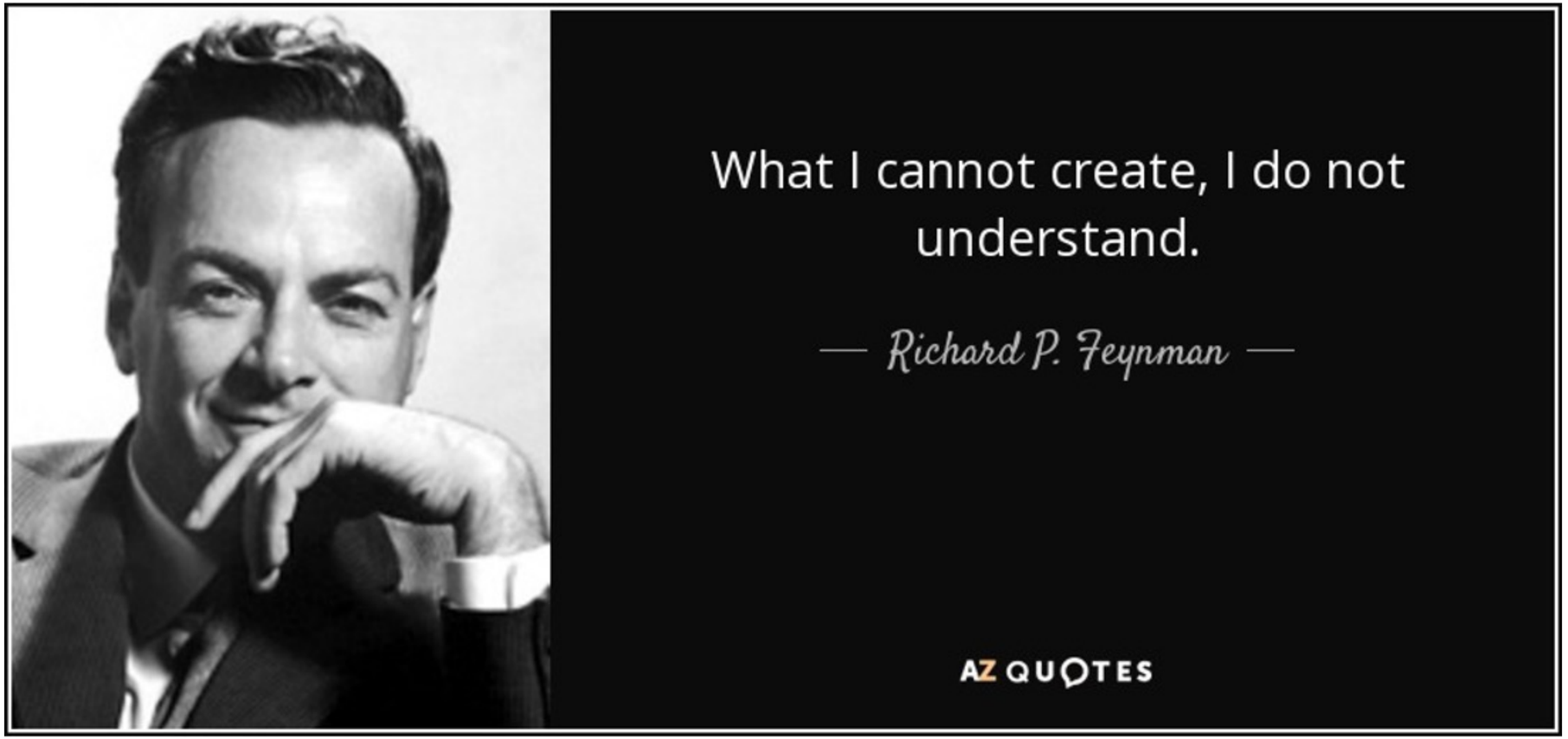


Unsupervised learning

- **Data.** Unlabeled data $\{\mathbf{x}_i\}_{i=1}^n \sim p(\mathbf{x})$
- **Goal.** Learn the underlying structure of the data



Generative modeling



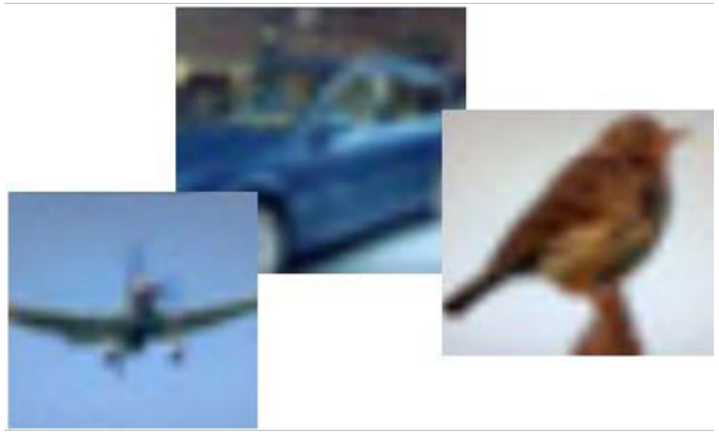
⇔ *“What I understand, I can create.”*

Generative modeling

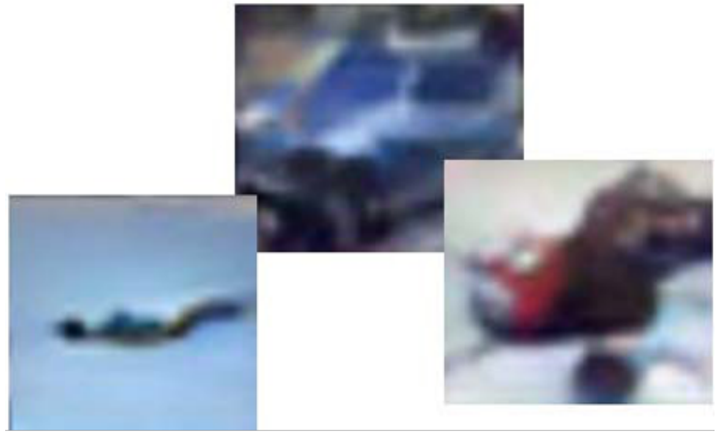
- We are given a dataset $D = \{\mathbf{x}_i\}_{i=1}^n$, where $\mathbf{x}_i \sim p_{\text{data}}(\mathbf{x})$
- **Goal.** Fit a nice model $p_\theta \approx p_{\text{data}}$

  so that we can generate new samples from $p_\theta(\mathbf{x})$

$$\min_\theta d(p_\theta, p_{\text{data}})$$



Training data $\sim p_{\text{data}}(\mathbf{x})$



New data $\sim p_\theta(\mathbf{x})$

Generative modeling

- We are given a dataset $D = \{\mathbf{x}_i\}_{i=1}^n$, where $\mathbf{x}_i \sim p_{\text{data}}(\mathbf{x})$
- **Goal.** Fit a nice model $p_\theta \approx p_{\text{data}}$

  so that we can generate new samples from $p_\theta(\mathbf{x})$
- **Flavors.**
  - **Explicit.** Explicitly define and solve for $p_\theta(\mathbf{x})$
  - **Implicit.** Learn a model that can sample from the model $p_\theta(\mathbf{x})$

What can good generative models do?

- Suppose that we have a good model on the *joint distribution*

$$p_\theta(\mathbf{x}, y) \approx p_{\text{data}}(\mathbf{x}, y)$$

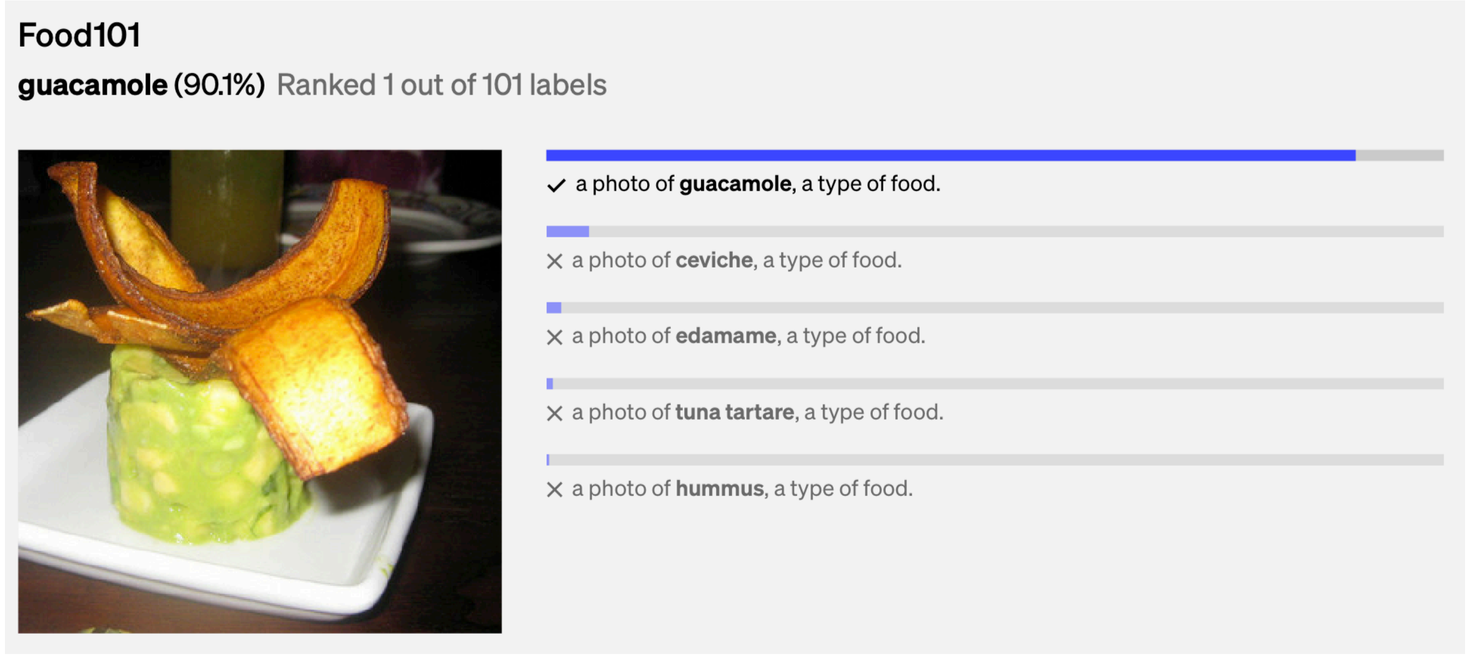
e.g., learned from image-text pairs crawled from web

What can good generative models do?

- Then we can easily build a *discriminative model* via Bayes rule—

$$p_\theta(y \mid \mathbf{x}) = \frac{p_\theta(\mathbf{x}, y)}{p_\theta(\mathbf{x})}$$

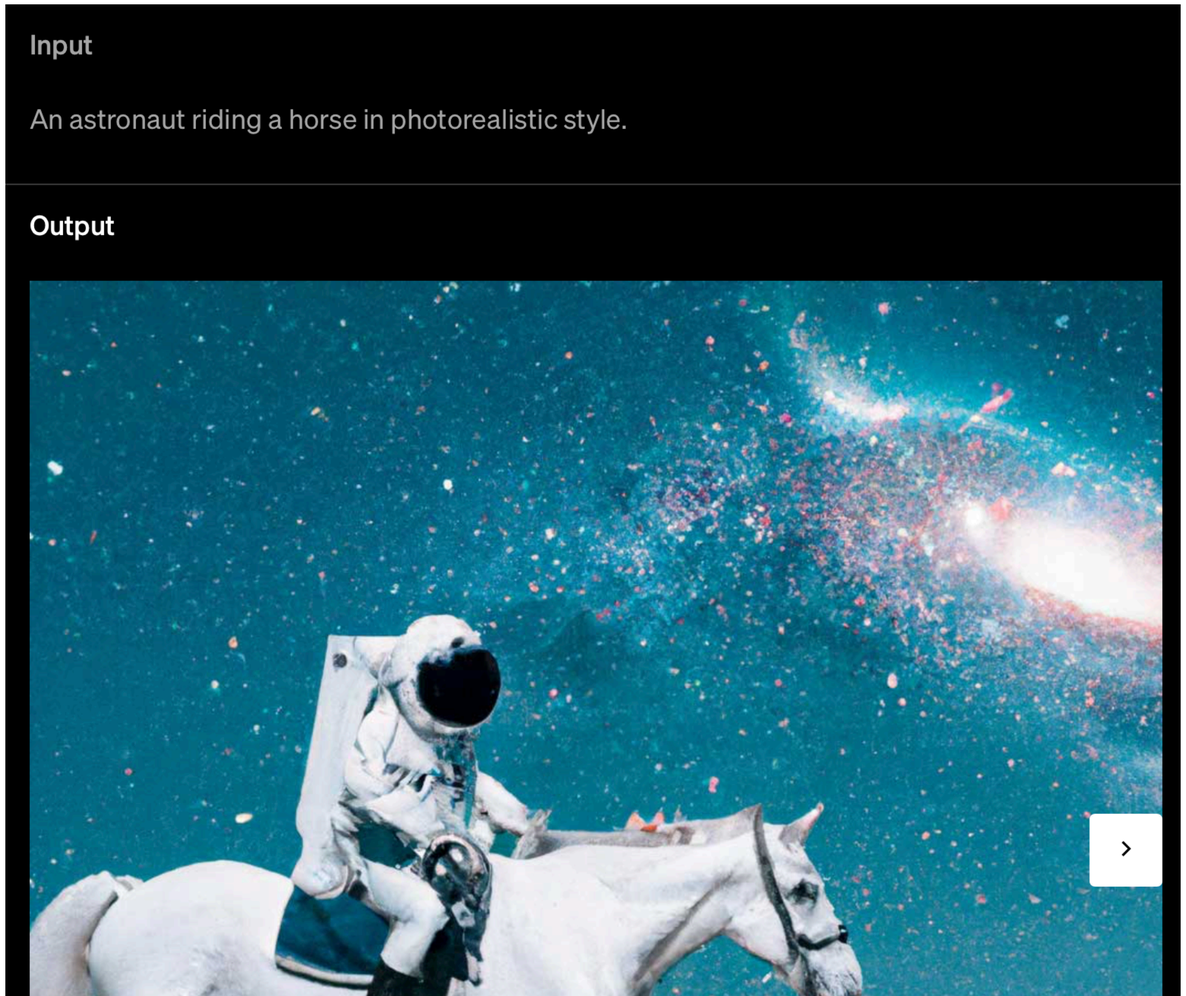
(often called generative classifiers)



OpenAI “Dall-E 2”

What can good generative models do?

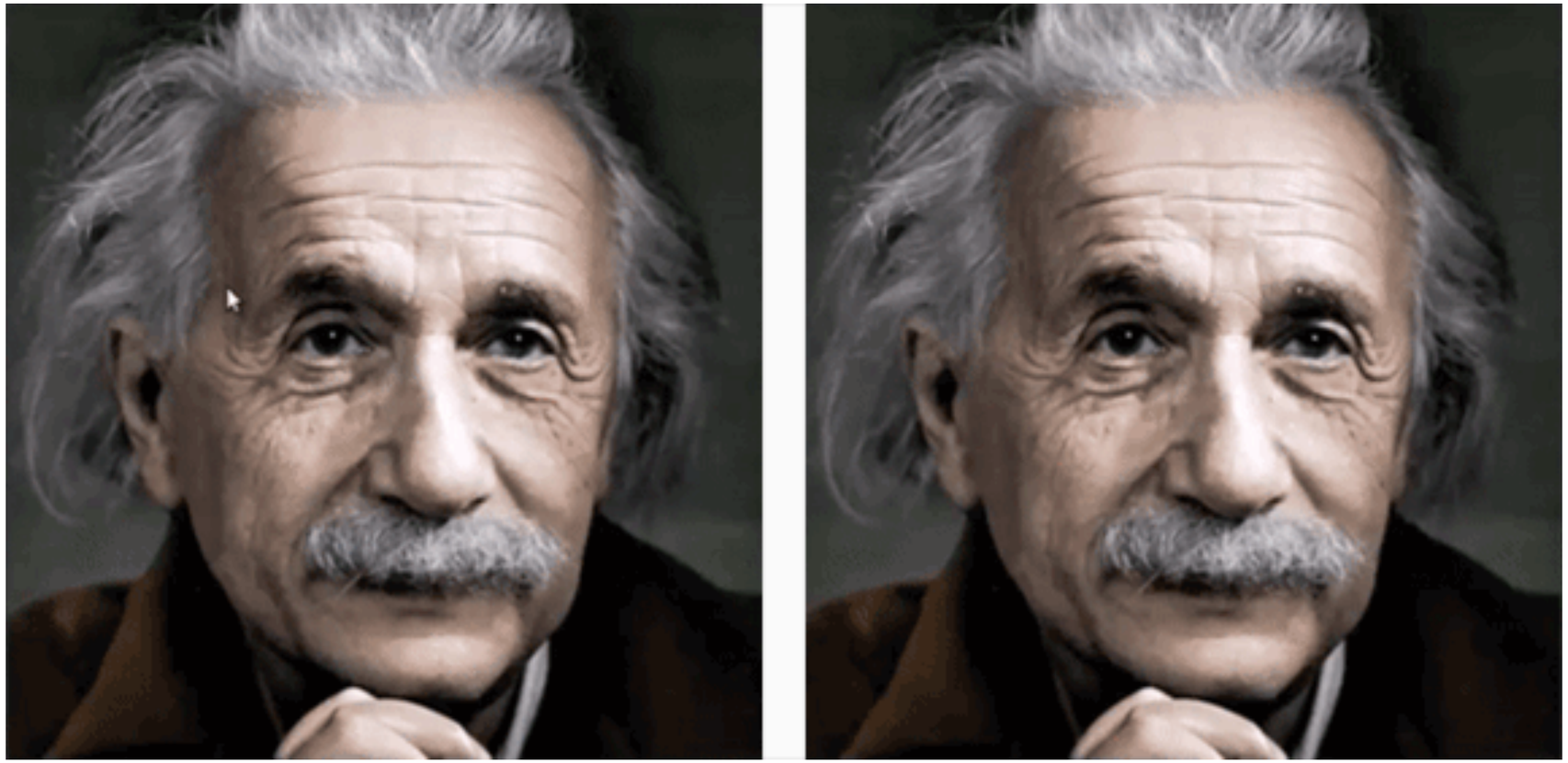
- We can also do *class-conditional generation* $p_\theta(\mathbf{x} \mid y)$



NVIDIA “Image Inpainting AI Playground”

What can good generative models do?

- We can also do *inpainting* $p_\theta(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d)$



OpenAI “ChatGPT”

What can good generative models do?

- We can also do *text generation* $p_\theta(y_{n+1} \mid y_1, \ldots, y_n)$

You

To study AI, where should I go? Please give a detailed answer.

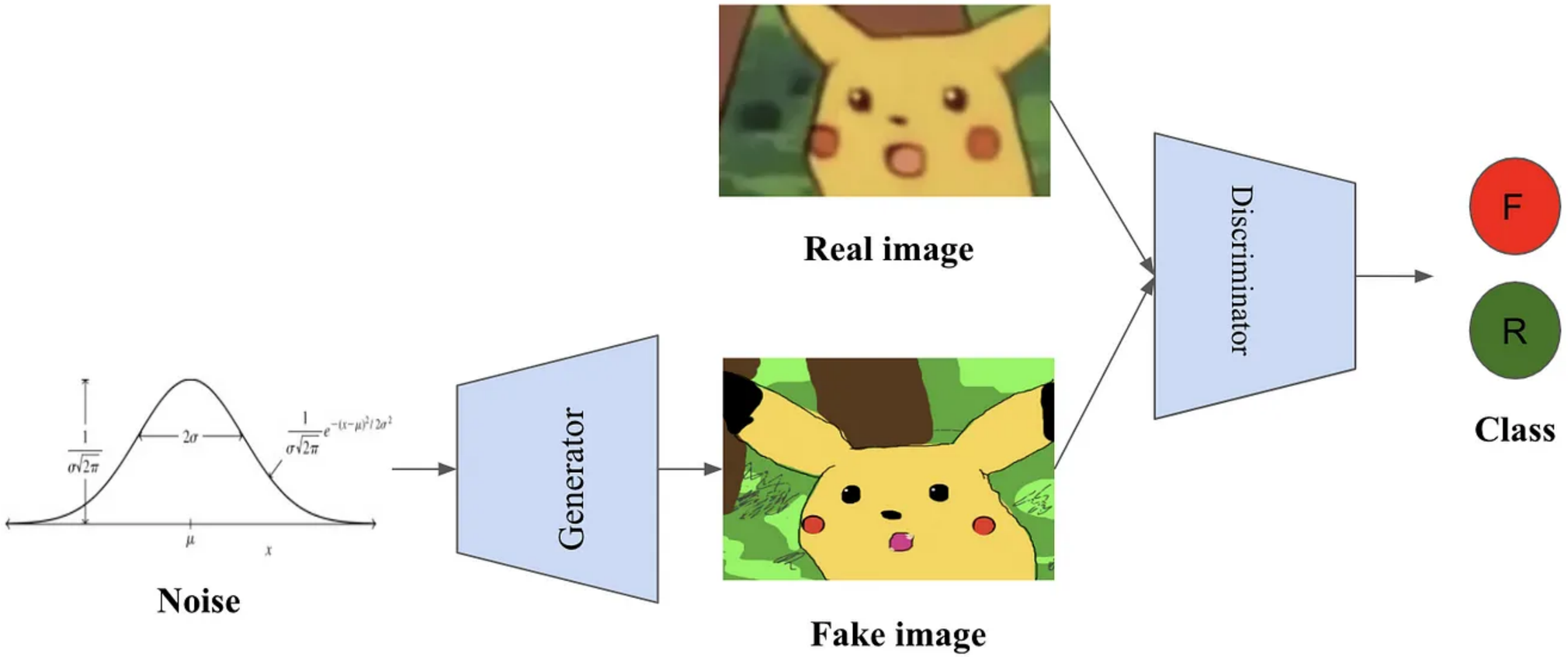
ChatGPT

To study AI, there are several excellent institutions and universities around the world that offer comprehensive programs and research opportunities. POSTECH is one of them, but there are many others as well. The choice of where to study AI depends on various factors such as your academic interests, location preferences, budget, and career goals.

Here are some top institutions known for their AI programs and research:

What can good generative models do?

- We can also perform *data augmentation* $\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{2n} \sim p_\theta(\mathbf{x})$



Classic Autoregressive Models

Preliminaries

- Chain rule

$$p(x_1, \ldots, x_n) = p(x_1)p(x_2 \mid x_1) \cdots p(x_n \mid x_1, \ldots, x_{n-1})$$

- Bayes rule

$$p(x \mid y) = \frac{p(x, y)}{p(y)} = \frac{p(y \mid x)p(x)}{p(y)}$$

- Conditional independence

$$\text{If } x \perp y \mid z, \text{ then } p(x \mid y, z) = p(x \mid z)$$

Fully Observable Model

- **Explicit Density Model**
  - Use chain rule to decompose the likelihood of an image $\mathbf{x}$ into products of 1D distributions.
  - Then maximize the likelihood of training data

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_1 \mid x_1, \ldots, x_{i-1})$$

Likelihood of image $\mathbf{x}$

Probability of i-th pixel, given all previous pixel values

Why use such re-expression?

- Suppose that we have a sample binary image $\mathbf{x} = (x_1, \ldots, x_n)$, with n binary pixels.
- Suppose a distribution $p(\mathbf{x}) = p(x_1, \ldots, x_n)$ that can sample an image.
- **Q.** How many possible states?
  - **A**. 2^n states.

Why use such re-expression?

- **Q.** How many parameters?
  - **A**. Consider a Bernoulli distribution $x \sim \text{Bern}(p)$. Then,

$$p(x_1, \ldots, x_n) = p(x_1)\,p(x_2 \mid x_1)\,p(x_3 \mid x_1, x_2) \cdots p(x_n \mid x_1, \ldots, x_{n-1})$$

1 param | 2 params | 4 params | 2^{n-1} params

Why use such re-expression?

- **Q.** How many parameters, under independence?
  - **A**. Consider a Bernoulli distribution $x \sim \text{Bern}(p)$. Then,

$$p(x_1, \ldots, x_n) = p(x_1)p(x_2)p(x_3)\cdots p(x_n)$$

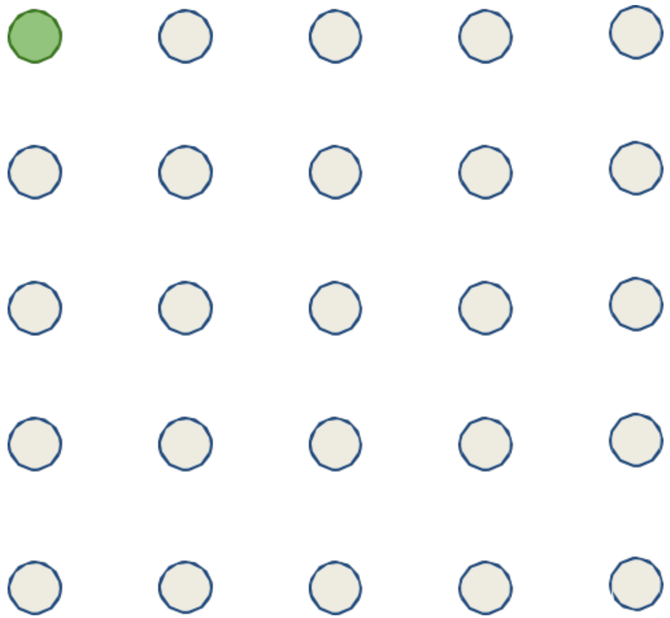
1 param

But this is too restrictive to model useful distribution...

🤔 Maybe have a neural net do this...

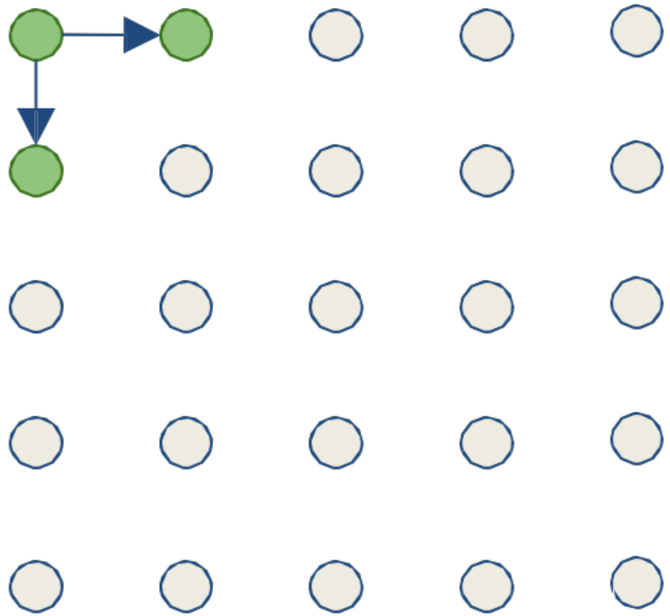
PixelRNN

- Generate image pixels starting from a corner.
- Dependency on previous pixels modeled using an **RNN (LSTM)**



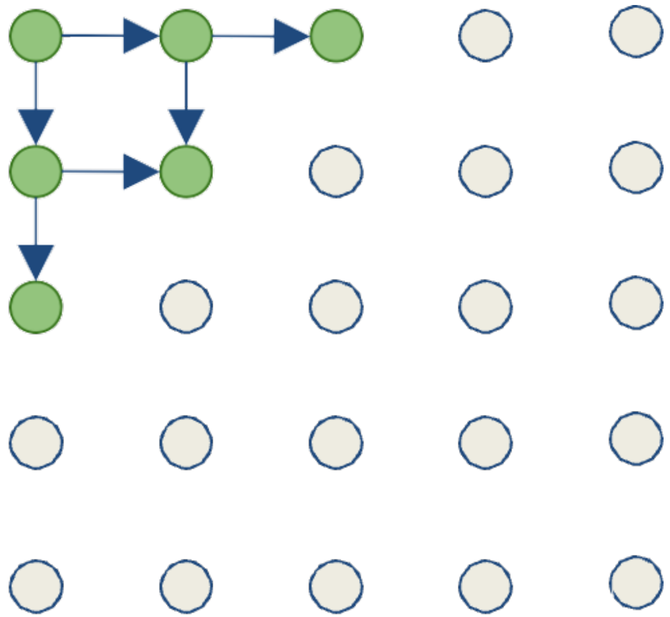
PixelRNN

- Generate image pixels starting from a corner.
- Dependency on previous pixels modeled using an **RNN (LSTM)**



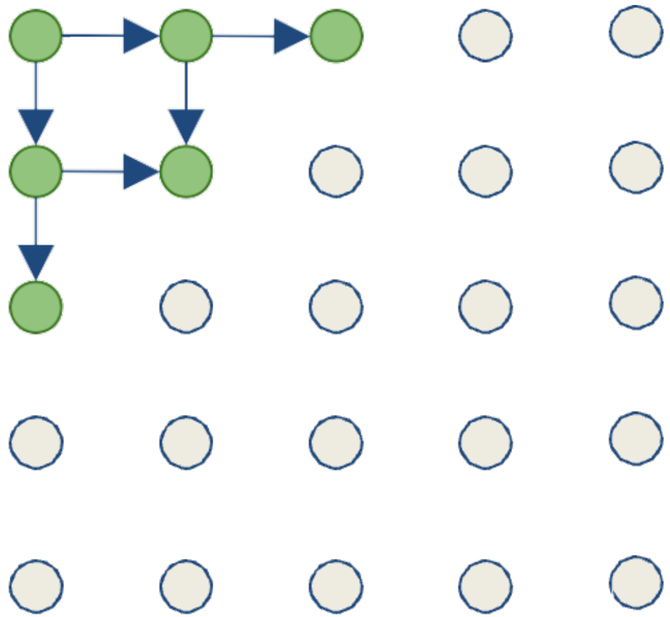
PixelRNN

- Generate image pixels starting from a corner.
- Dependency on previous pixels modeled using an **RNN (LSTM)**



PixelRNN

- Generate image pixels starting from a corner.
- Dependency on previous pixels modeled using an **RNN (LSTM)**
  - **Training.** Maximize likelihood over all training images



Why use such re-expression?

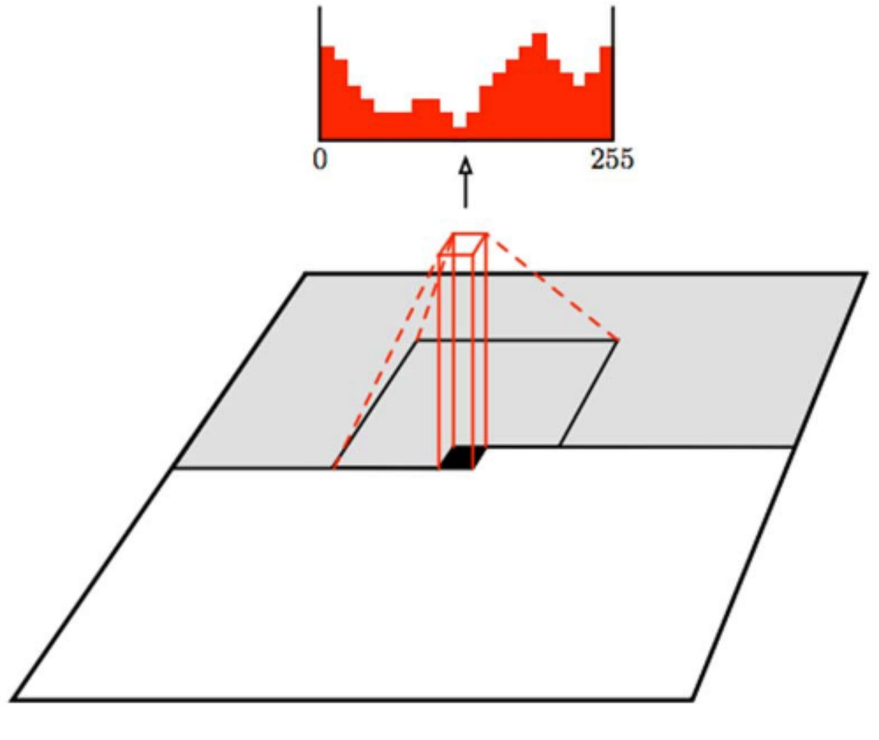
- Use conditional independence
  - Markov 1st order assumption: $p(x_{i+1} \mid x_1, \ldots, x_i) = p(x_{i+1} \mid x_i)$
- **Q.** Number of parameters?

$$p(x_1, \ldots, x_n) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2) \cdots p(x_n \mid x_{n-1})$$

  - **A.** Total $2n - 1$, which is an exponential reduction.

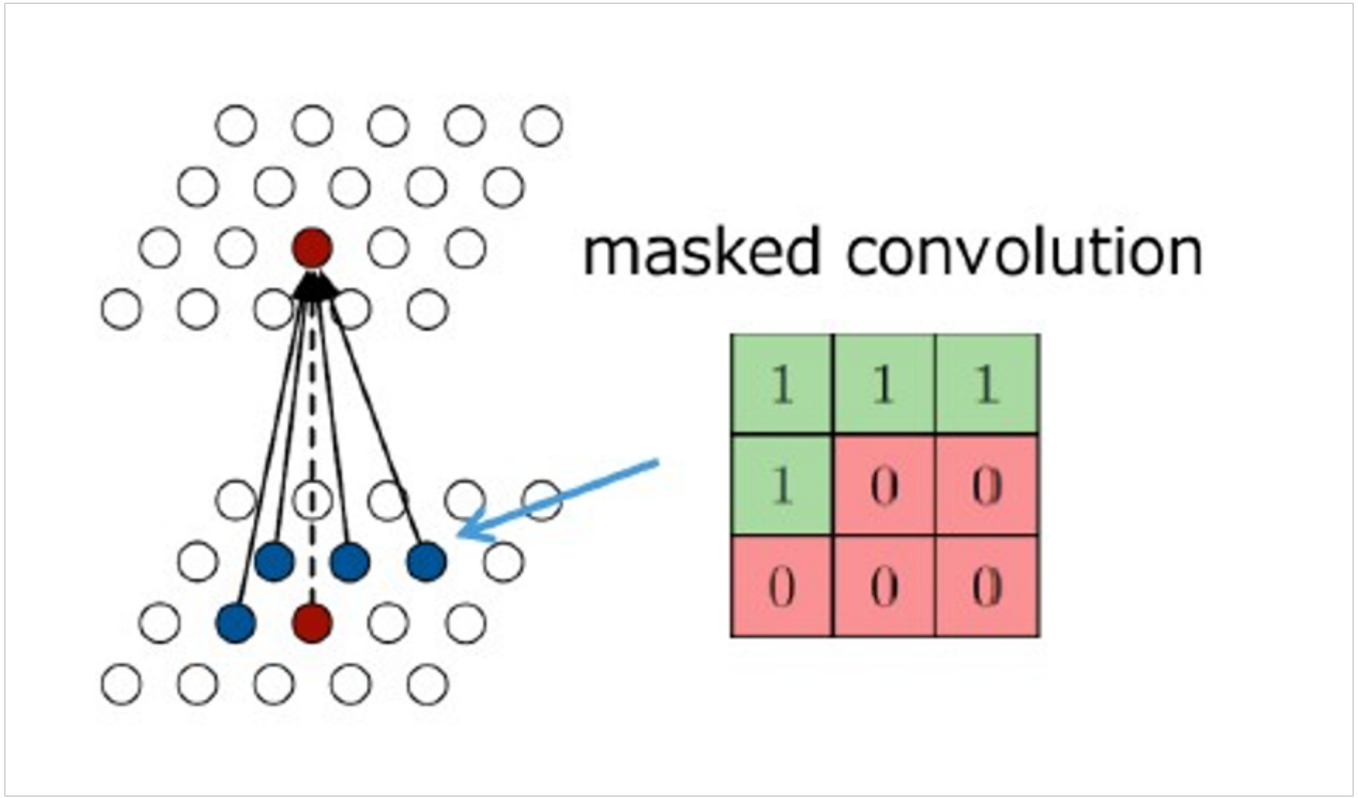
PixelCNN

- Again, generate pixels starting from corner.
- Dependency now modeled using CNN over context regions



Masked Convolutions

- Apply masks so that a pixel does not see “future” pixels



PixelCNN

- Again, generate pixels starting from corner.
- Dependency now modeled using CNN over context regions
- Faster training than PixelCNN
  - Can parallelize convolutions since context region values are known.
- Generation must be done sequentially...

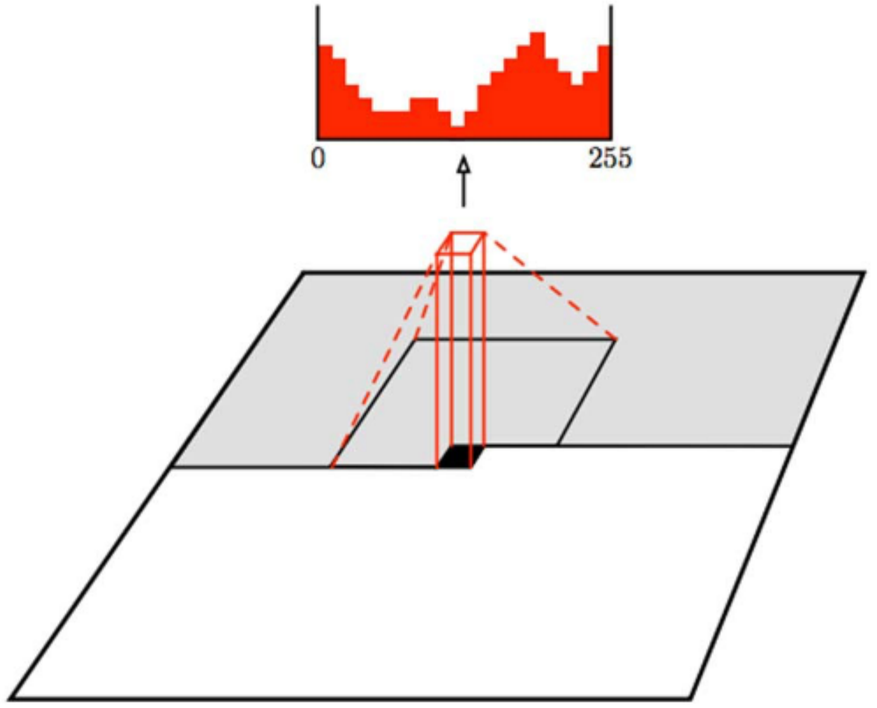
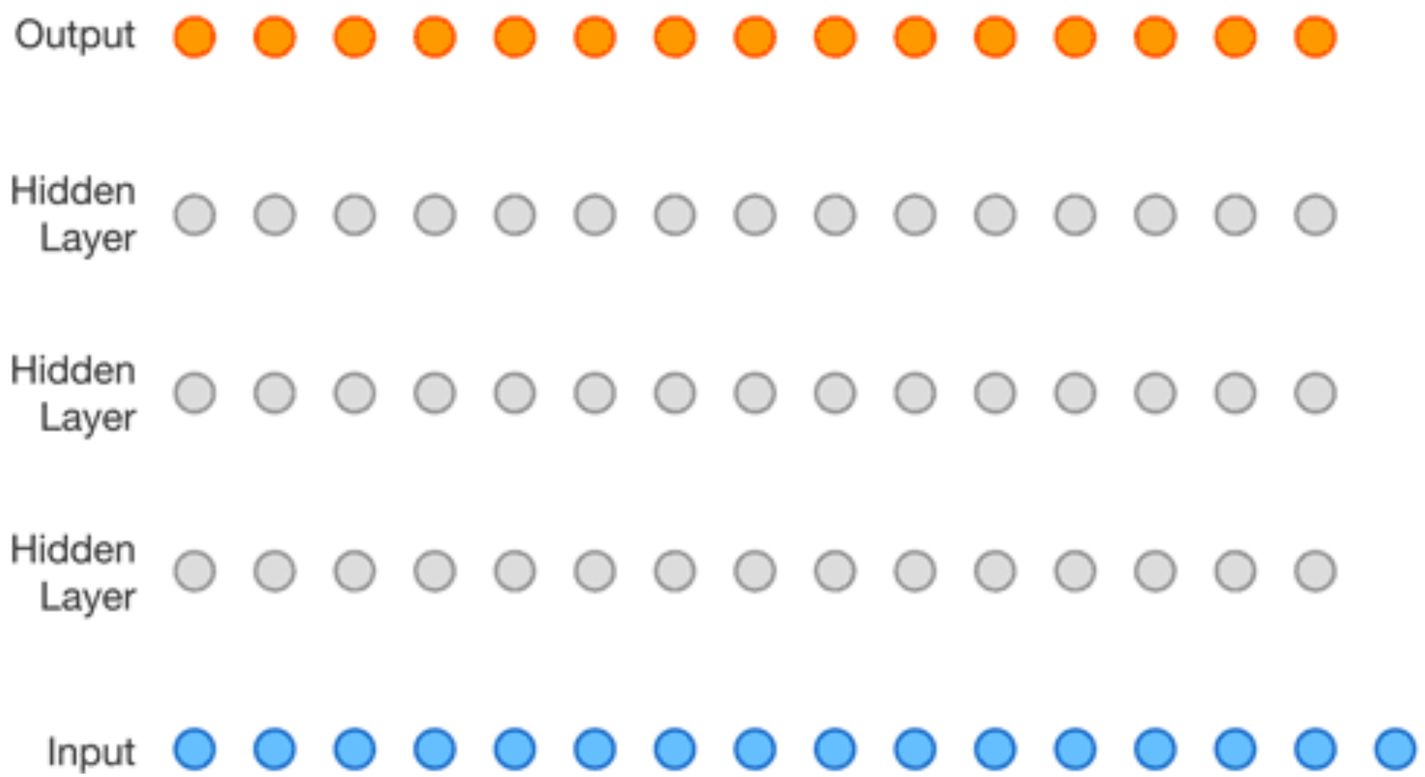


Figure 1. Image completions sampled from a PixelRNN.

Special case: WaveNet

- Basically a 1D pixelCNN for speech / music generation.



PixelRNN / PixelCNN

- **Pros.**
  - Can explicitly compute likelihood $p(\mathbf{x})$
  - Explicit likelihood of training data gives good evaluation metric
  - Good quality, in general.

- **Cons.**
  - Sequential generation; too slow!

(many follow-ups though; PixelCNN++)

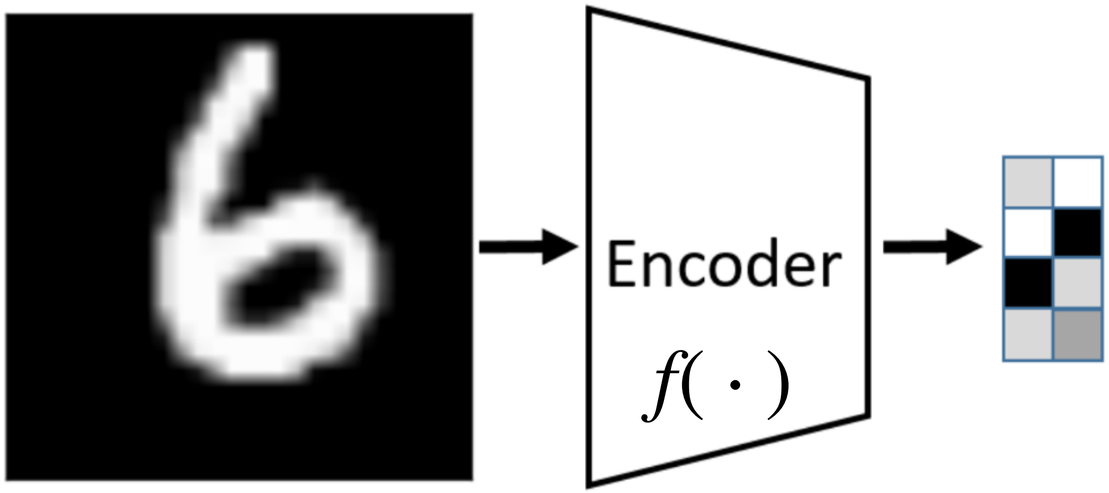
Autoencoders

Basic Autoencoder

Input $\mathbf{x}$

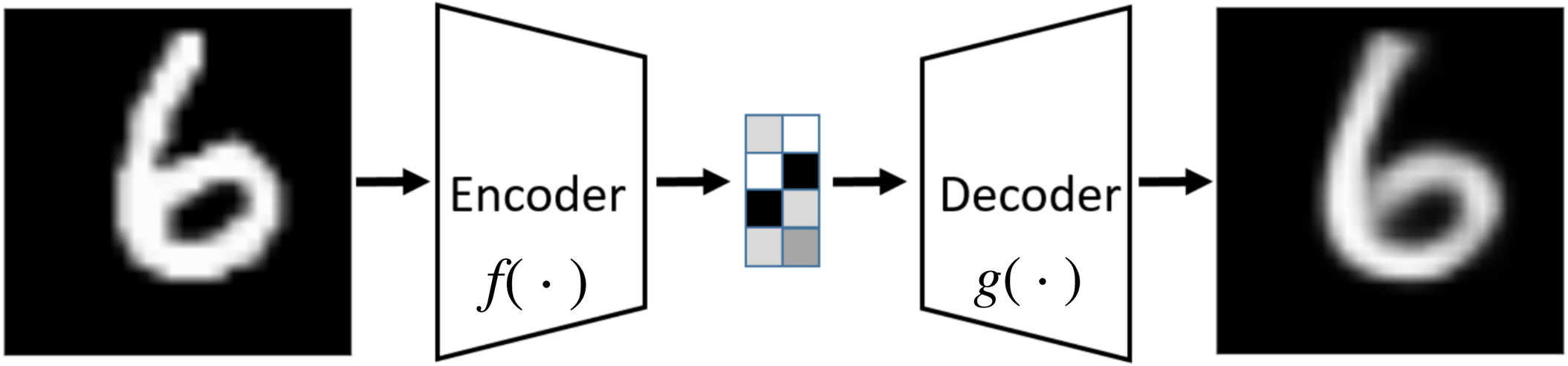
Basic Autoencoder

Input $\mathbf{x}$ Representation $f(\mathbf{x})$

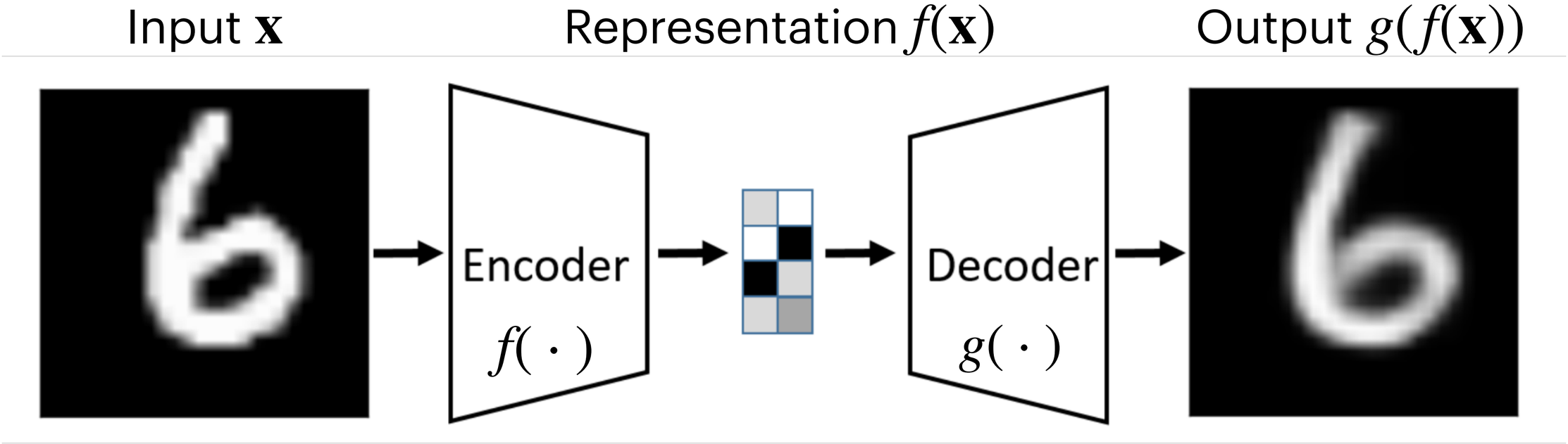


Basic Autoencoder

Input $\mathbf{x}$ Representation $f(\mathbf{x})$ Output $g(f(\mathbf{x}))$

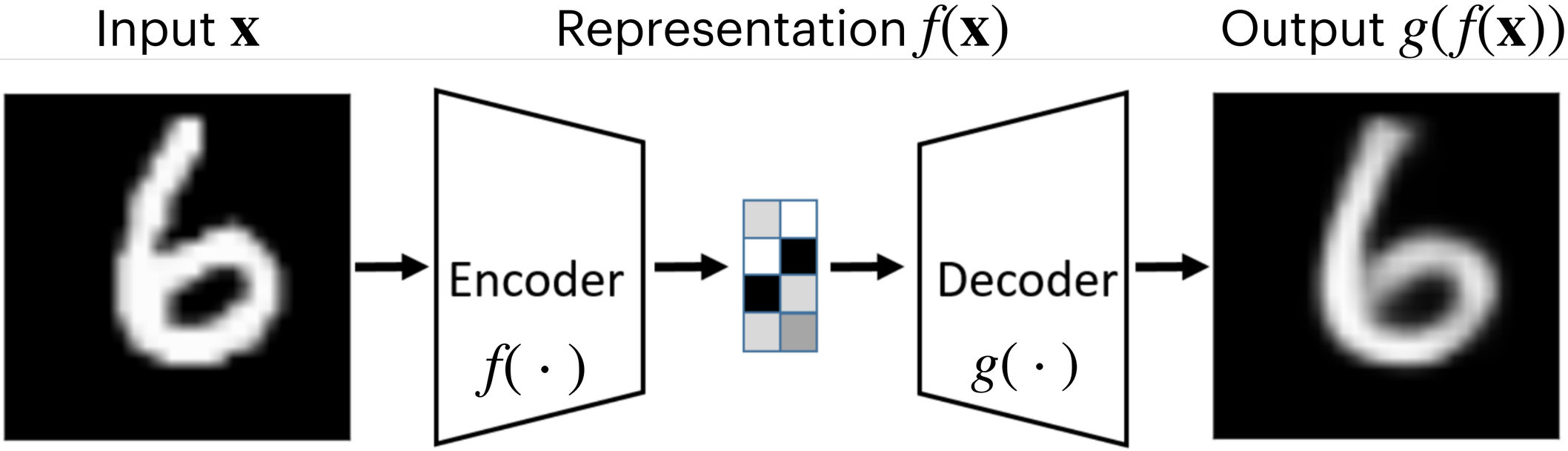


Basic Autoencoder



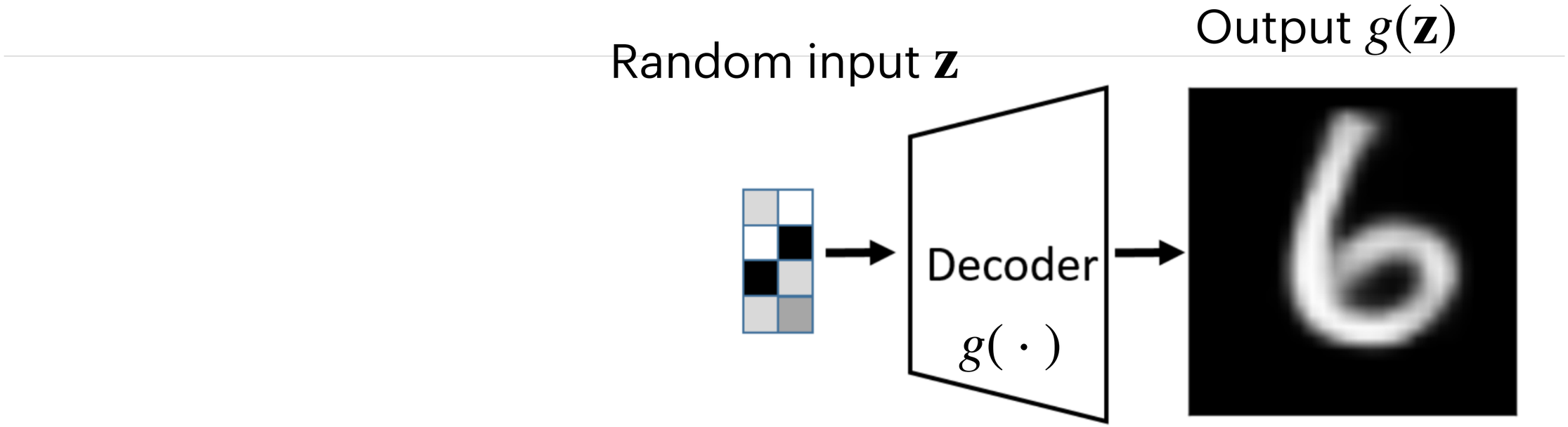
- Train with the reconstruction loss $L(\mathbf{x}, g(f(\mathbf{x})))$
  - The encoder/decoders are neural networks (or PCA if linear)

Basic Autoencoder



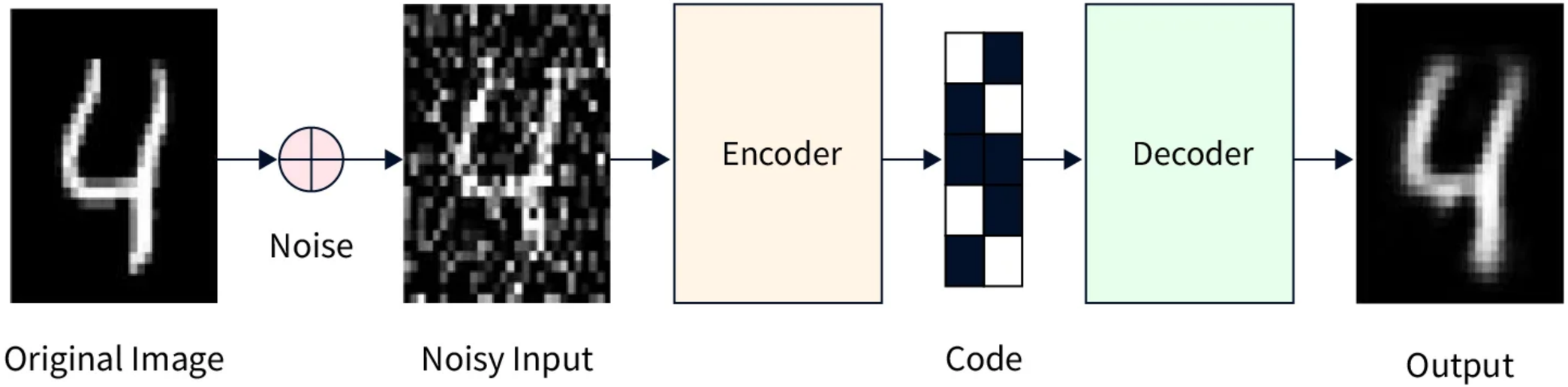
- **Problem.** A trivial solution when $f(\cdot) = g(\cdot) = \text{Identity}$
  - **Solution.** An "hourglass" structure / regularization
    (not a "generative" model yet; how do we sample?)

Basic Autoencoder



- **Hope.** Could this work like a “generator”?
  - Blind hope, with no mathematical grounding 😅

Denoising Autoencoder

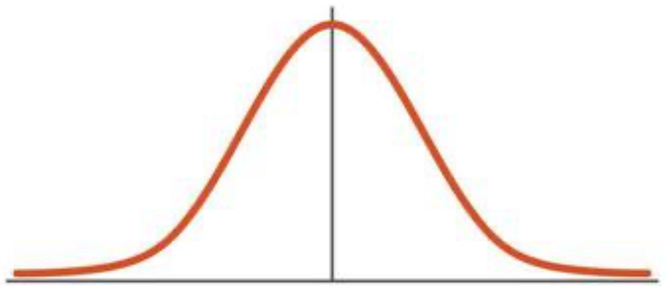


- **Idea.** Add noise to the input, and train to recover the clean image.
  - Still very limited capability as a generative model

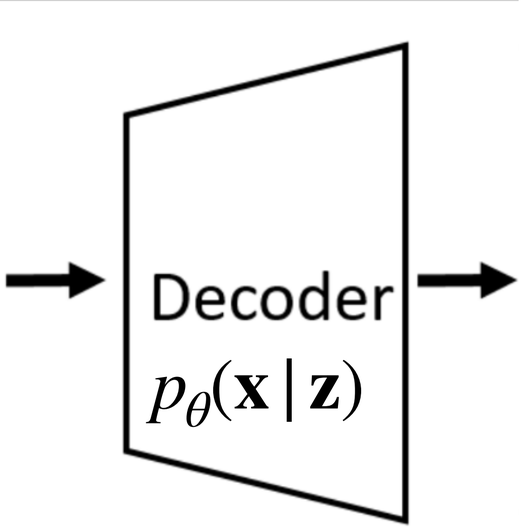
Note. This is what modern “diffusion models” do!

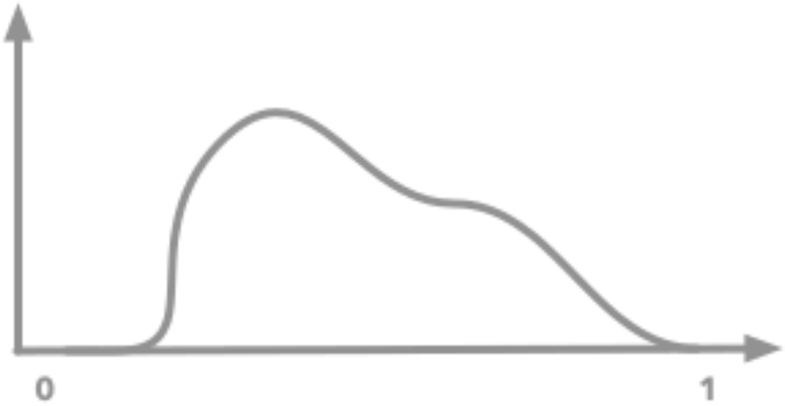
Variational Autoencoder — Rough Idea

- Train a decoder and a distribution such that
  if we send in a distribution, we get a data-generating distribution.



$p_\theta(\mathbf{z})$

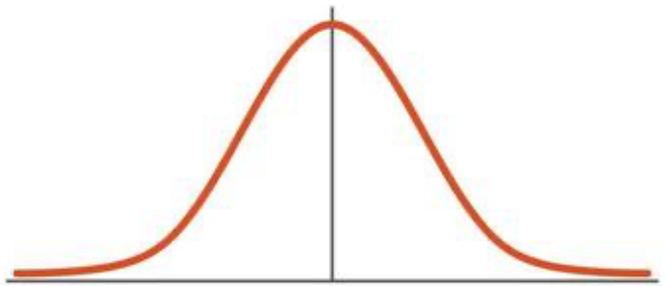


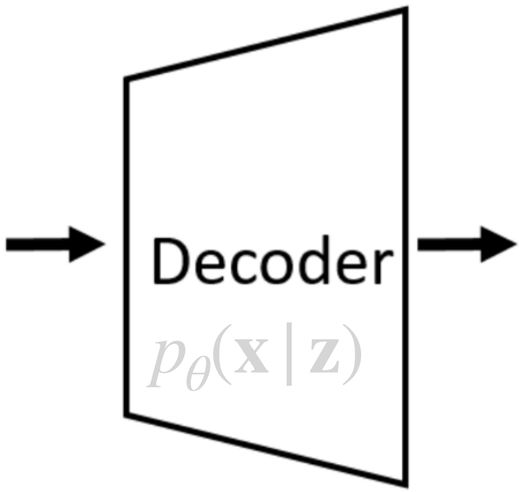


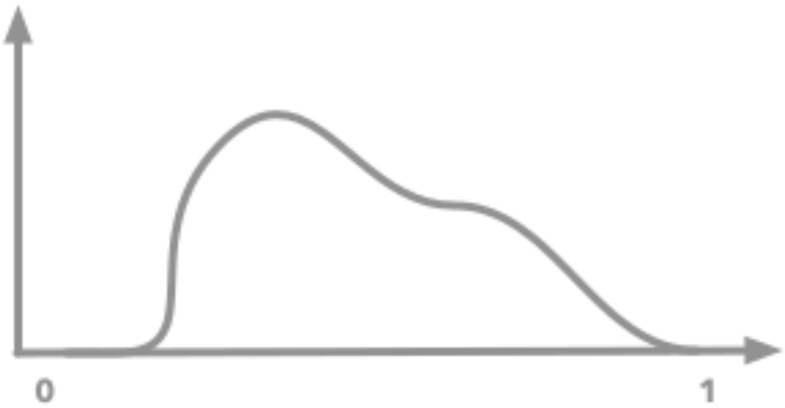
$p_\theta(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$

Variational Autoencoder — Rough Idea

- Train a decoder and a distribution such that
  if we send in a distribution, we get a data-generating distribution.







$p_\theta(\mathbf{z})$

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

$p_\theta(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$

Cheers

- *Next up.* VAE (continued), GANs, Diffusion models