



CIS 4190/5190: Lec 23 Mon Nov 25, 2024

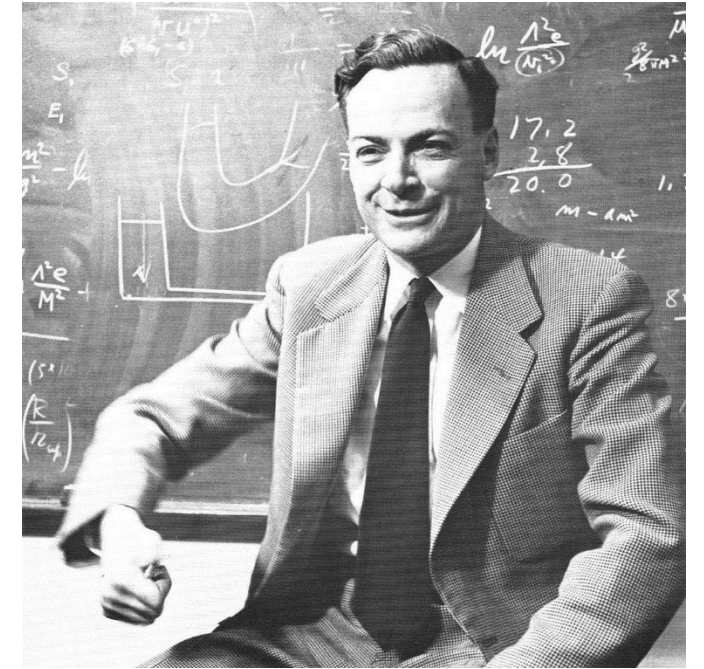
An Introduction to Generative Models

Or Unsupervised Learning Part 3

Introduction



What I cannot create,
I do not understand.



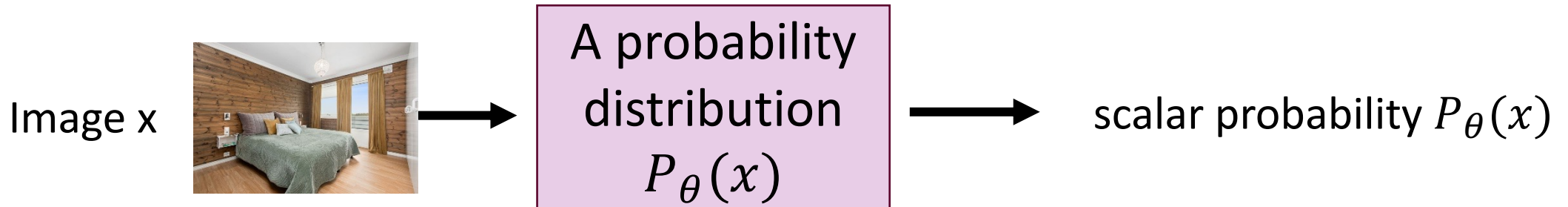
Richard Feynman: *“What I cannot create, I do not understand”*

Generative modeling: *“What I understand, I can **create**”*

Statistical Generative Models

A statistical generative model is a **probability distribution** $P_{\theta}(x)$

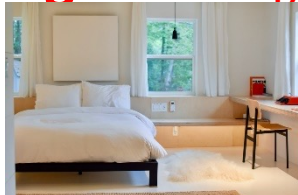
- **Data:** samples (e.g., images of bedrooms)
- **Prior knowledge:** parametric form (e.g., Gaussian?), loss function (e.g., maximum likelihood?), optimization algorithm, etc.



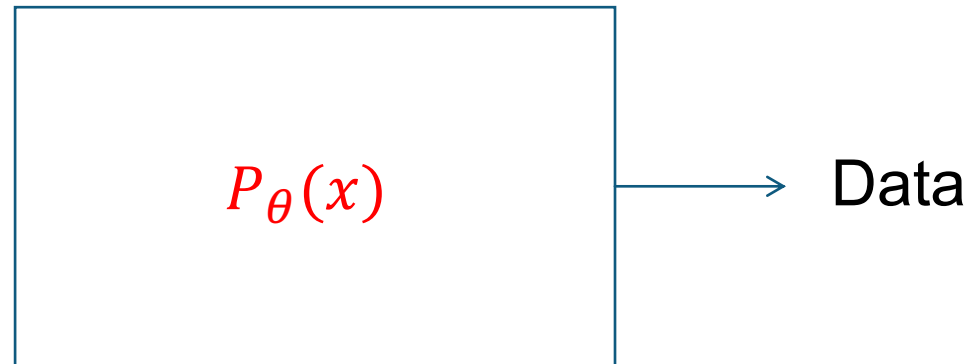
It is generative because **sampling from $P_{\theta}(x)$ generates new images**



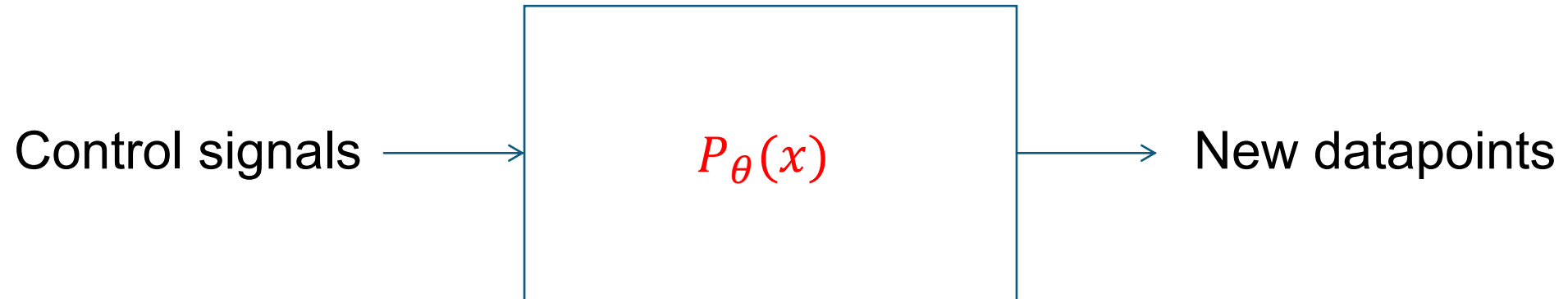
...



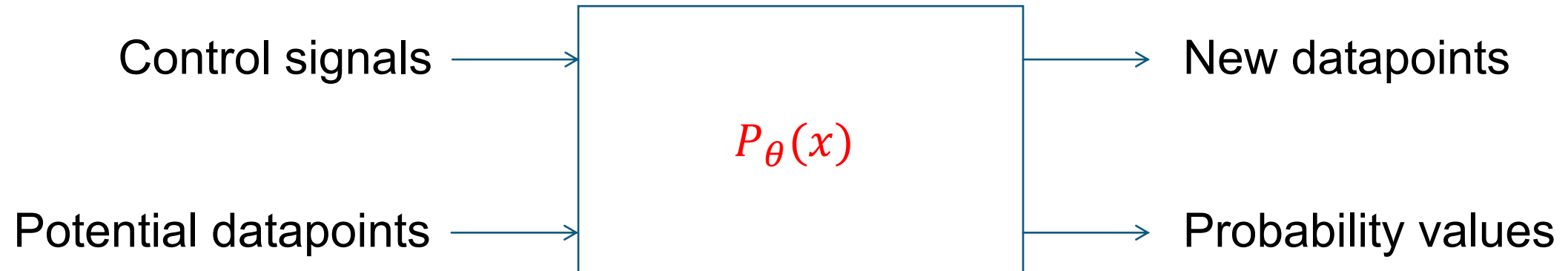
Building a simulator for the data generating process



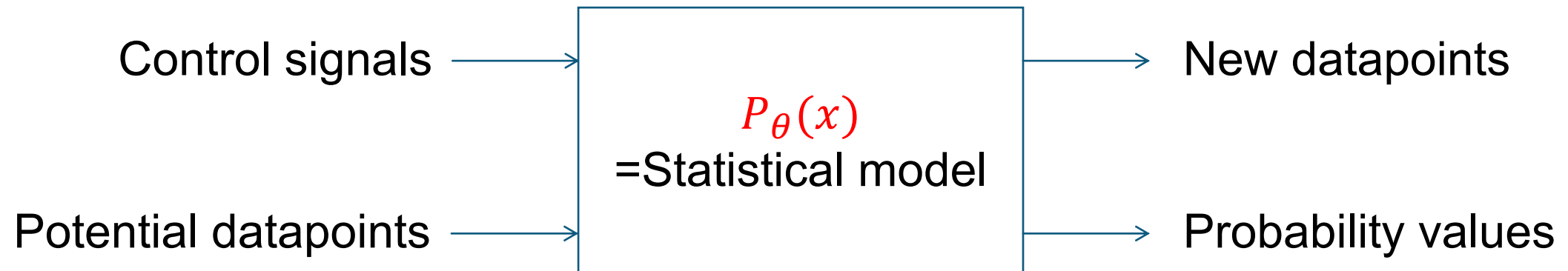
Building a simulator for the data generating process



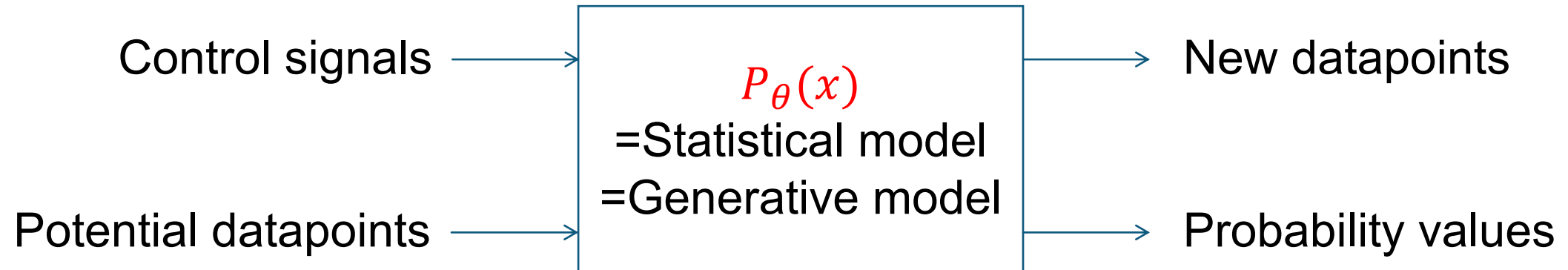
Building a simulator for the data generating process



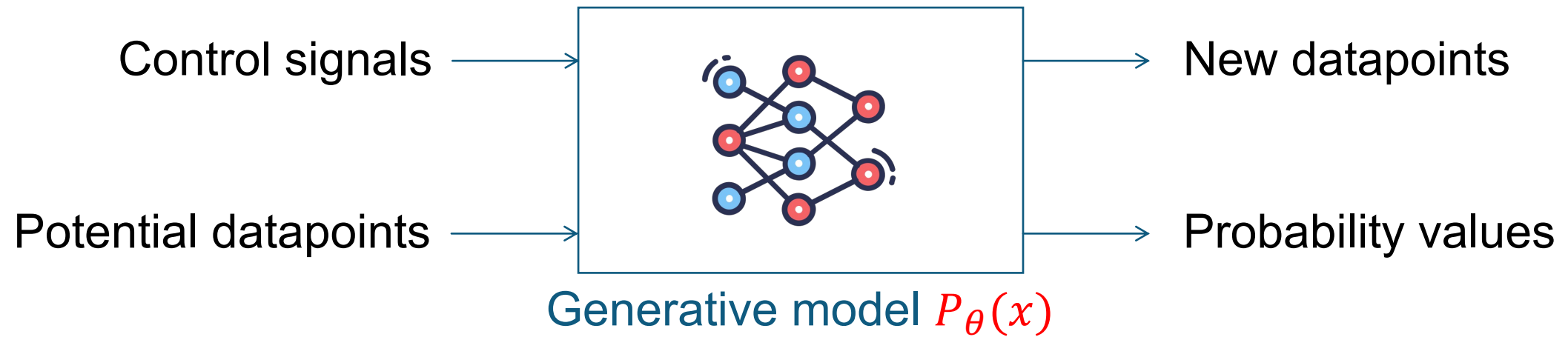
Building a simulator for the data generating process



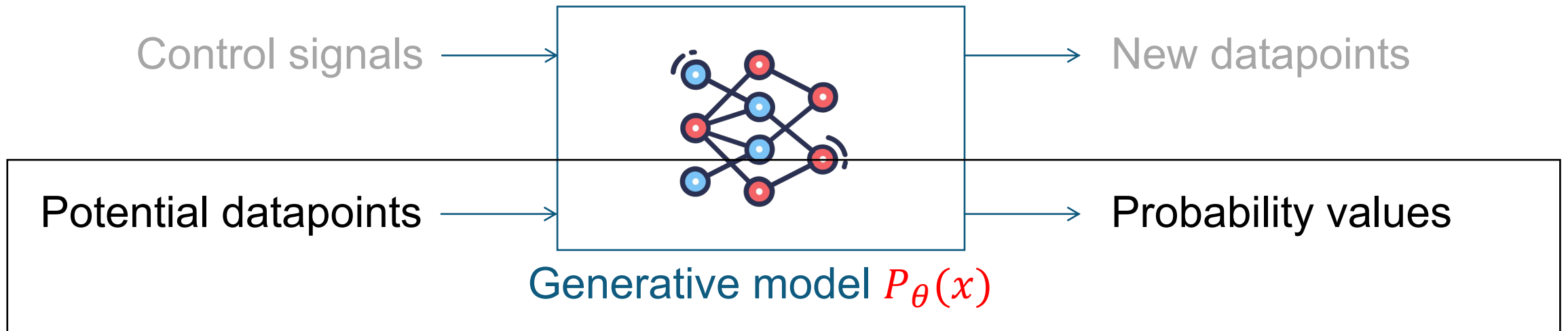
Building a simulator for the data generating process



Building a simulator for the data generating process

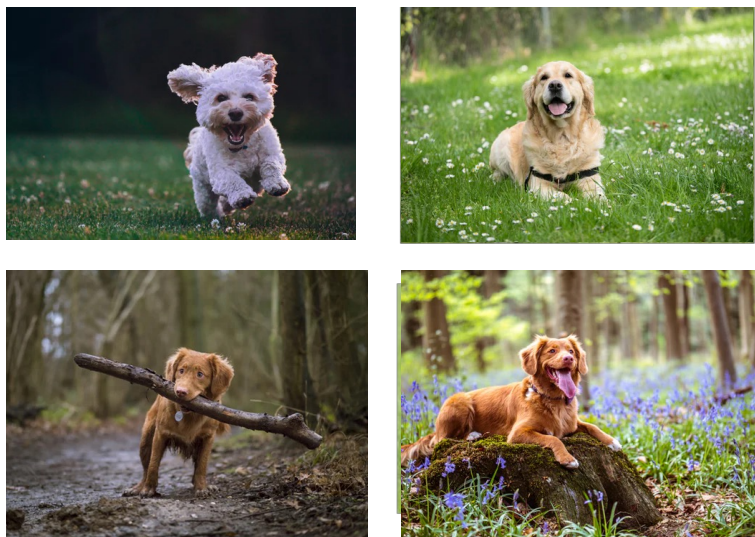


We Have Already Seen
Generative Models In Disguise!

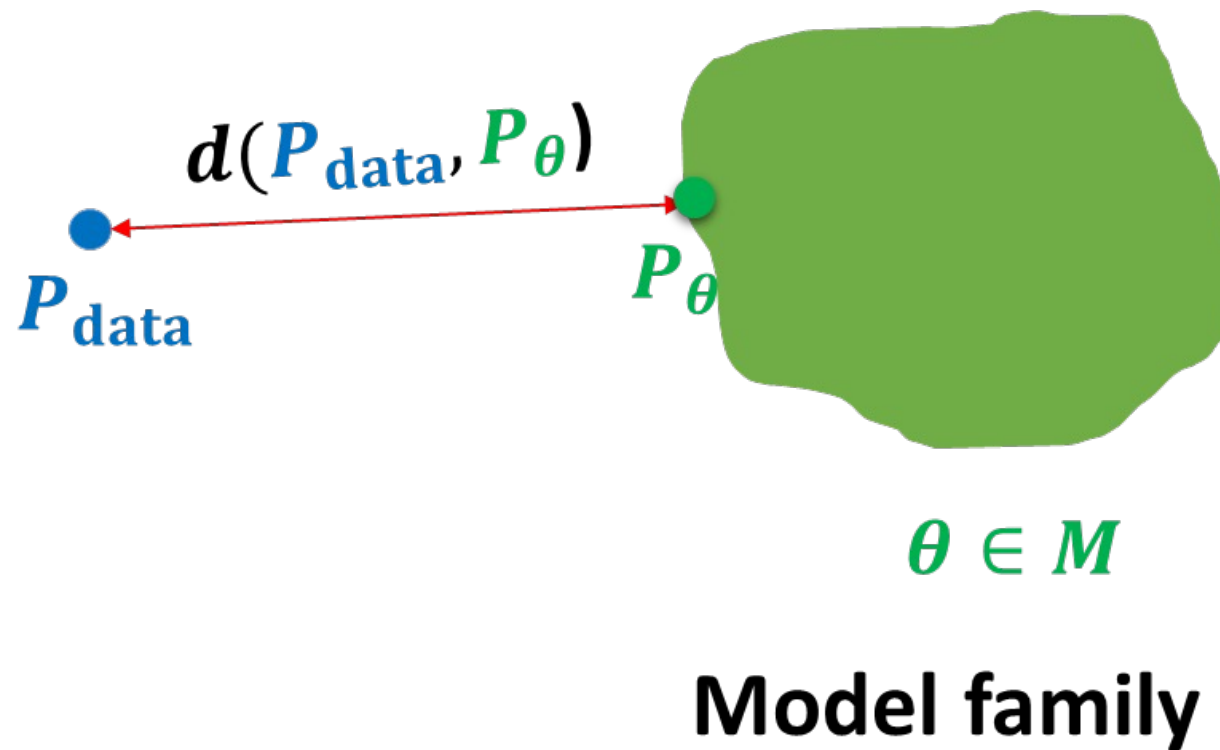


We want parameters θ such that model $P_{\theta}(x)$ assigns high probabilities to samples from the training data distribution, and low values to samples that are not in the data distribution.

Fit data distribution to data



$$\begin{aligned} X_i &\sim P_{\text{data}} \\ i &= 1, 2, \dots, n \end{aligned}$$



A Simple Generative Model: Fit A Gaussian Density

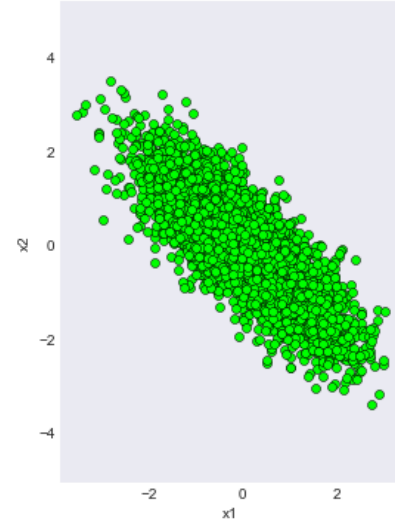
Suppose we are modeling a dataset $\mathcal{D} = \{x_1, x_2, x_3, \dots, x_n\}$, $x_i \in \mathbb{R}^d$

Suppose we select the gaussian family as our model class:

$$P_{\theta}(x) = \mathcal{N}(x; \mu = \theta_1, \Sigma = \theta_2)$$

Then we could fit parameters θ_1 and θ_2 to as:

$$\theta^* = \arg \max_{\theta} \prod_{i=1 \dots n} P_{\theta}(x_i) = \arg \max_{\theta} \sum_{i=1 \dots n} \log P_{\theta}(x_i)$$



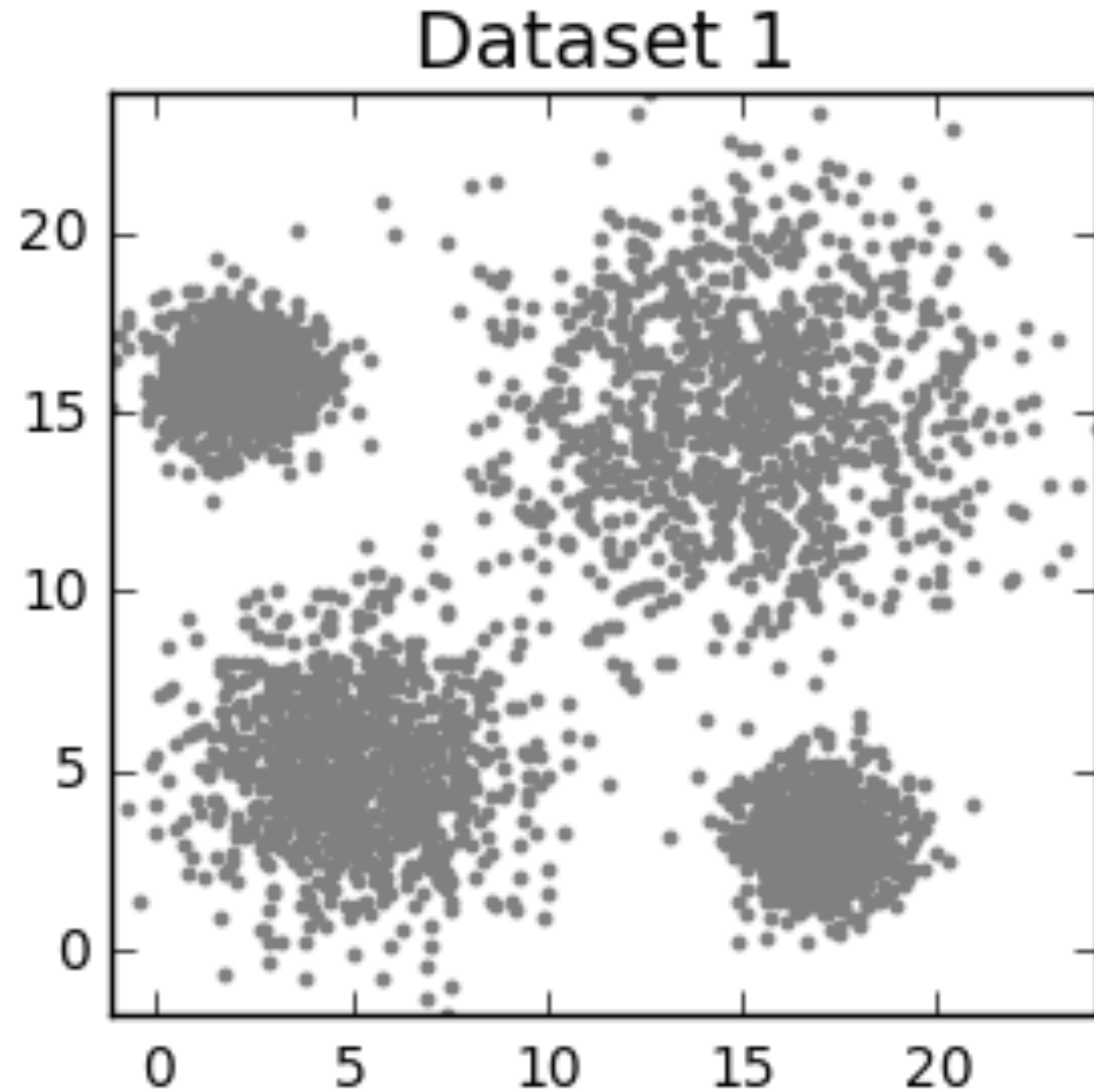
This is a “maximum likelihood” objective!

These work out to:

$$\theta_1 = \text{mean}(\{x_1, \dots, x_n\}), \theta_2 = \text{covariance}(\{x_1, \dots, x_n\})$$

Q: Can you show this to be true?

Modeling More Complex Data Distributions?



Recap: K-Means Clustering

Optimizer:

“Alternating Minimization”

K-Means (K, X)

- Randomly choose K cluster center locations (centroids)
- Loop until convergence, do:
 - Assign each point to the cluster of the closest centroid
 - Re-estimate the cluster centroids based on the data assigned to each cluster

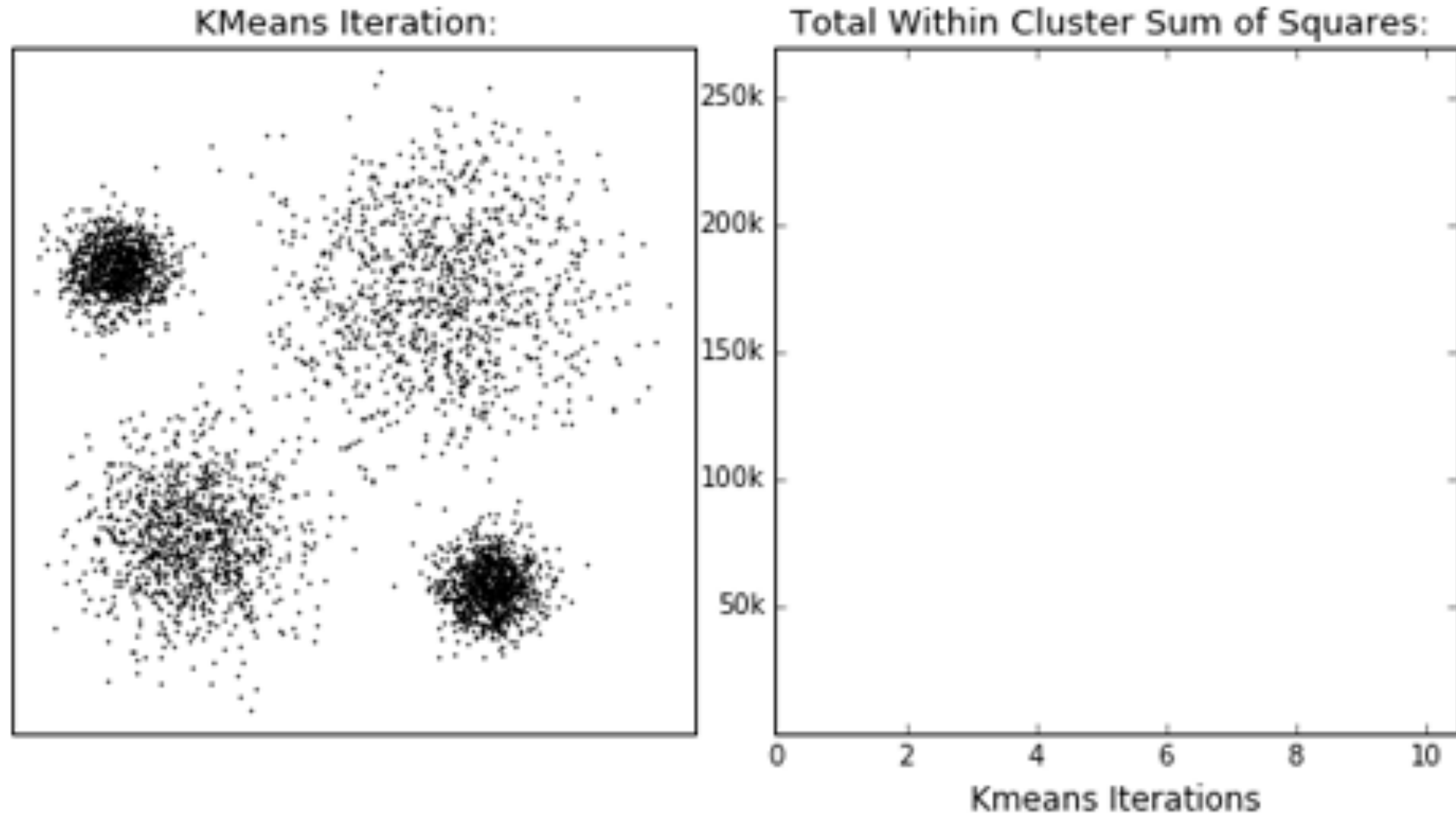
K-means finds a local optimum of the following objective function:

“Sum of squared distances”
loss function

$$\arg \min_{\mathcal{S}} \sum_{k=1}^K \sum_{x \in S_k} \|x - \mu_k\|_2^2$$

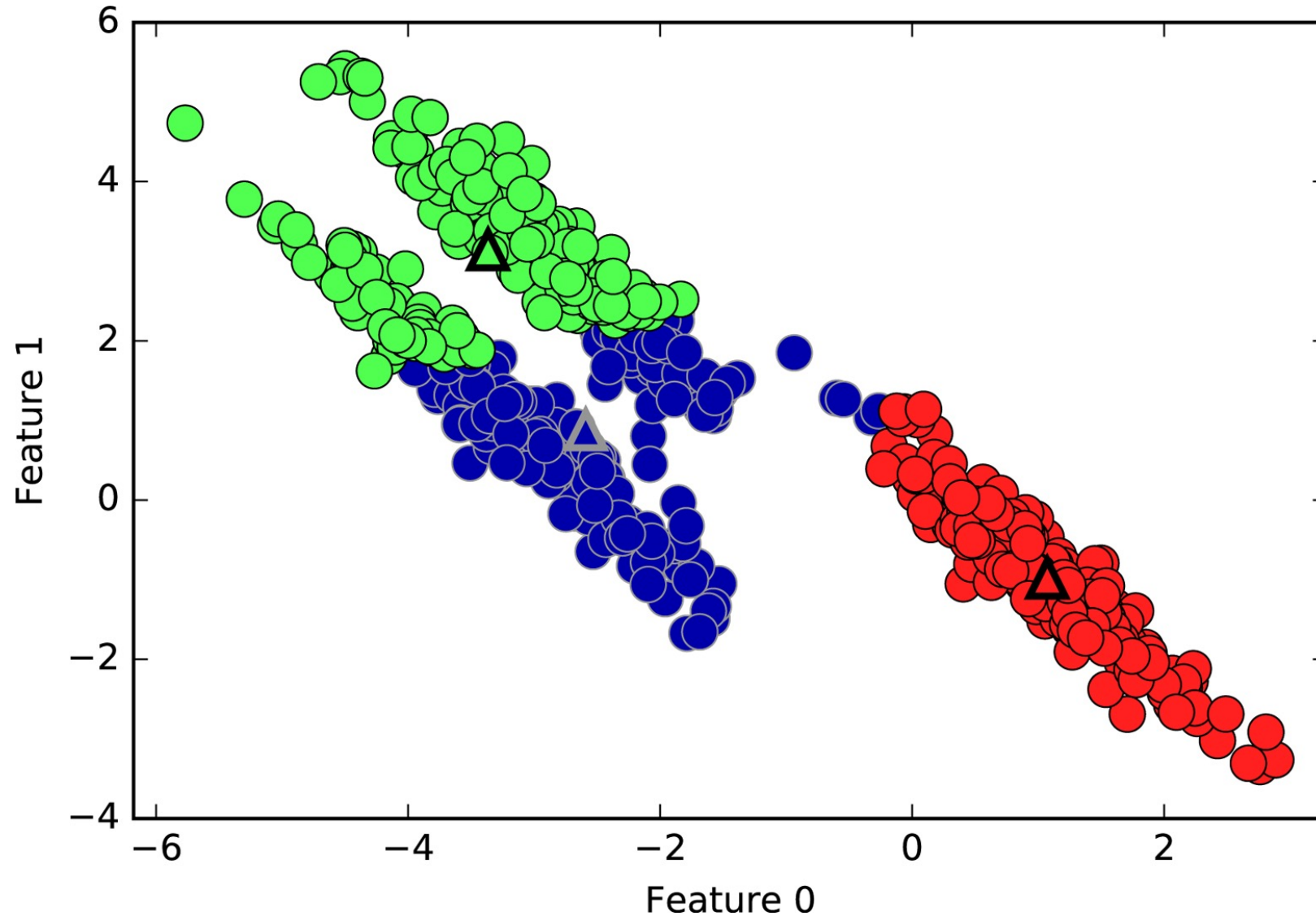
where $\mathcal{S} = \{S_1, \dots, S_K\}$ are sets corresponding to disjoint clusters, and the clusters together include all samples.

Recap: K-Means Clustering Convergence

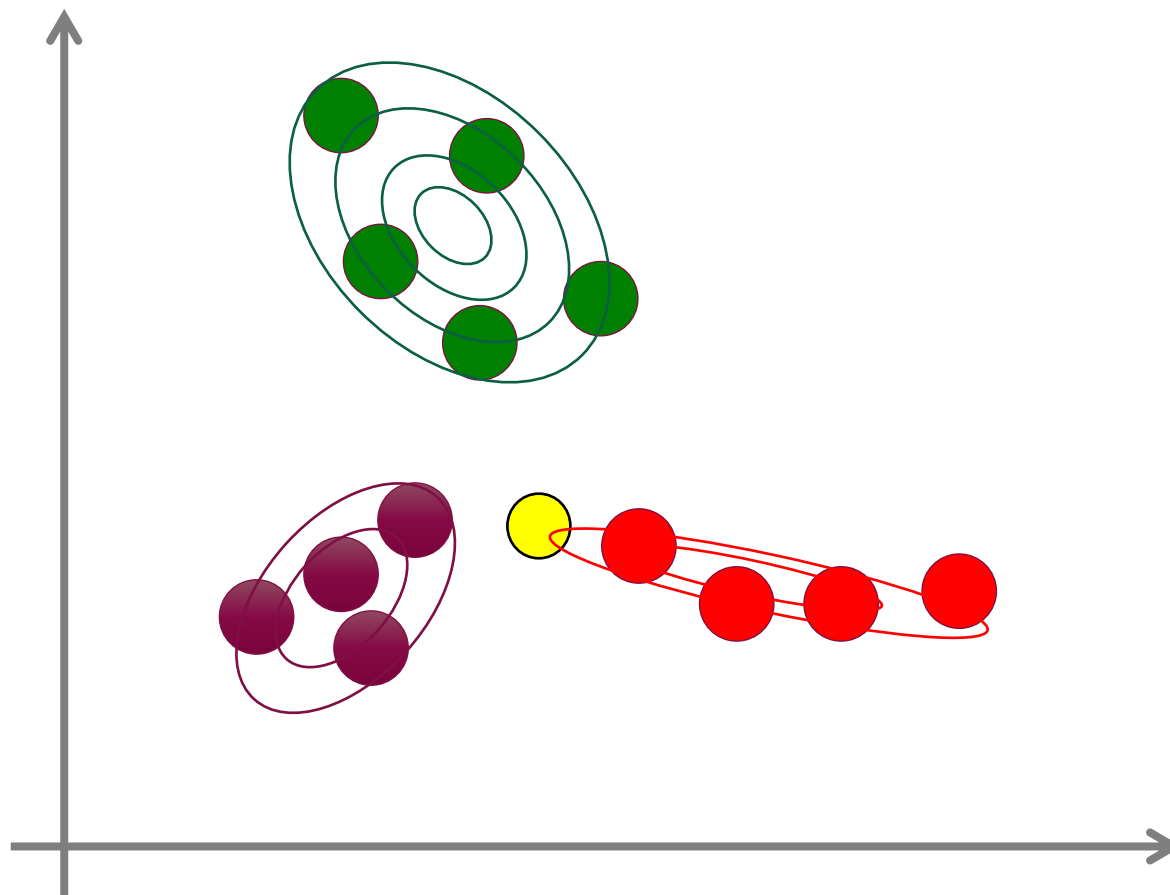


From K-Means to GMMs

K-Means on Non-Spherical Clusters?



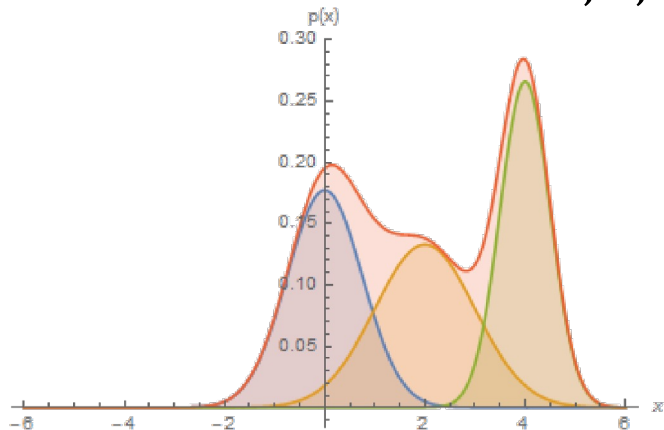
“Gaussian Mixture Models”



Gaussian Mixture Models

- The generative process works as:
 - Sample a “latent” $z \sim p_\theta(z)$ = a categorical distribution over $1, 2, \dots, K$
 - Then sample $x \sim p_\theta(x|z = k) = \mathcal{N}(x; \mu_\theta^k, \Sigma_\theta^k)$
- The resulting distribution is a “mixture of Gaussians”, whose likelihood is:

$$p_\theta(x) = \sum_{z=1, \dots, K} p_\theta(z) p_\theta(x|z) = \sum_{k=1}^K p_\theta(z = k) \mathcal{N}(x; \mu_\theta^k, \Sigma_\theta^k)$$

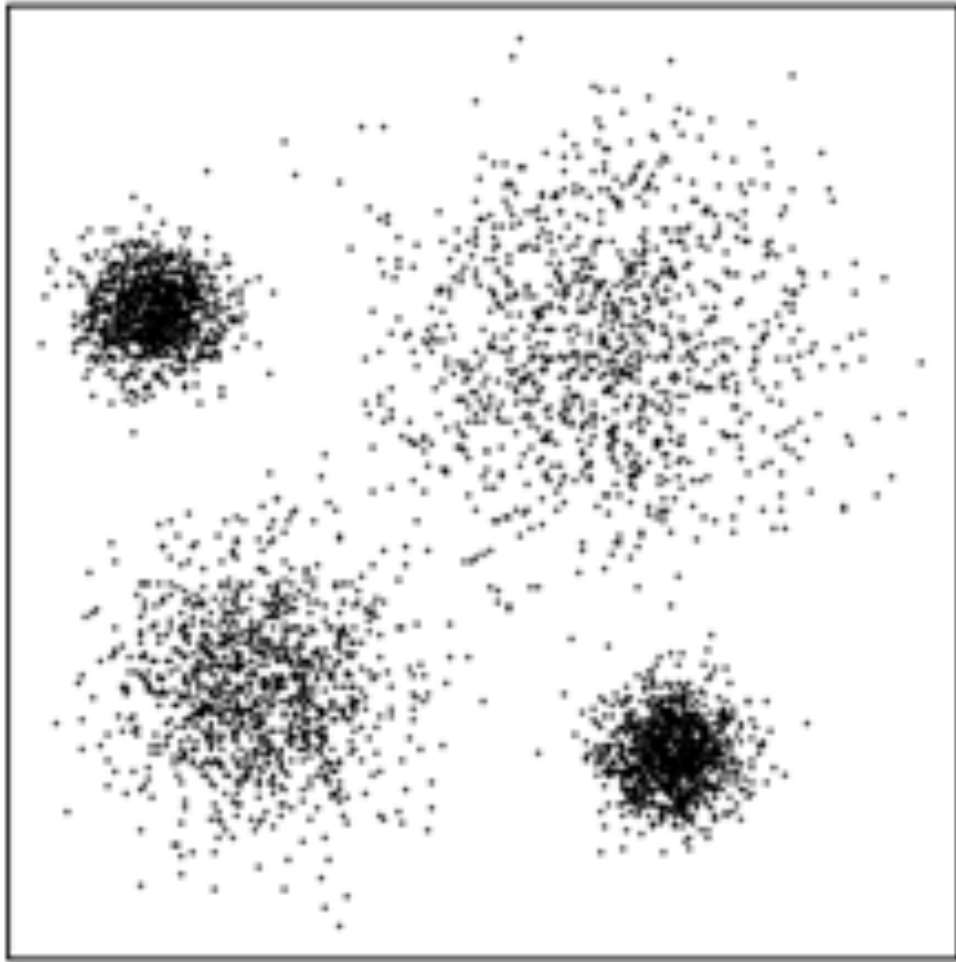


Permits a k-means-like “alternating optimization procedure”, called “Expectation maximization” which **approximates the maximum likelihood fit.**

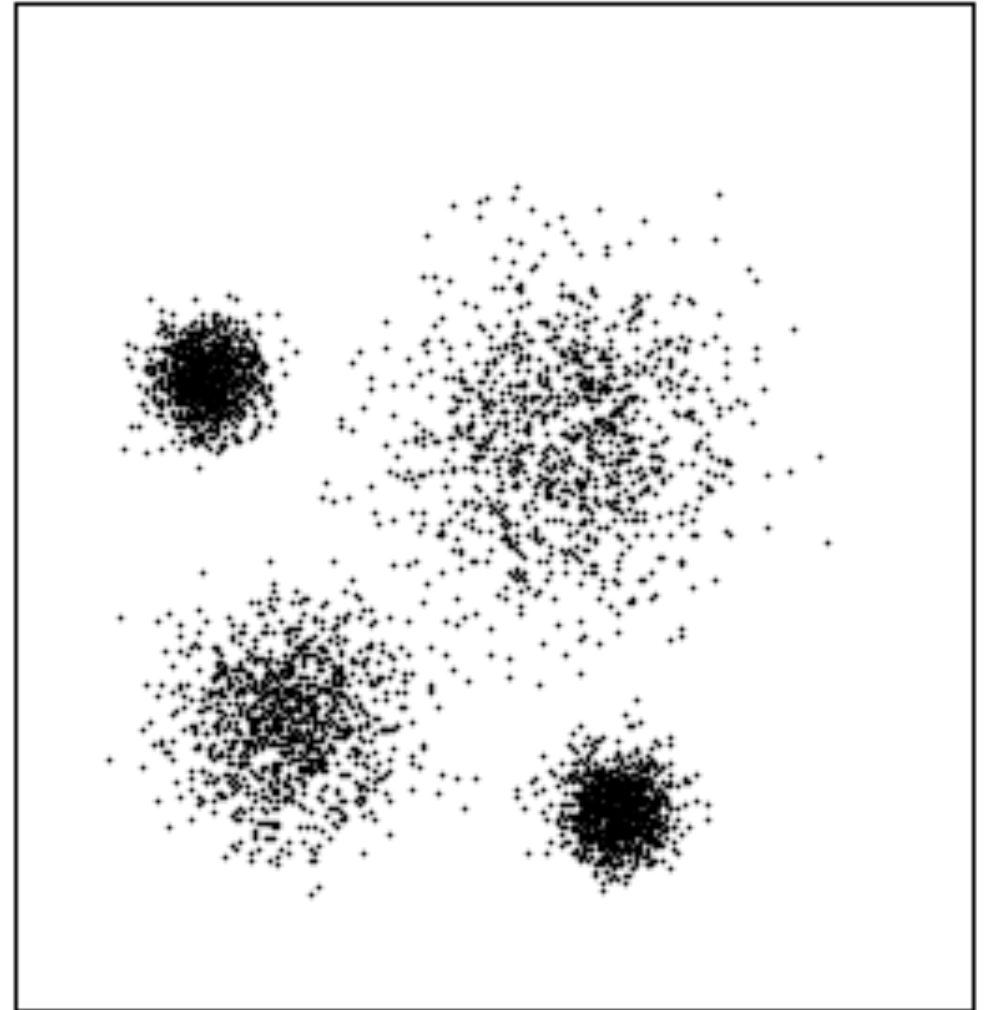
Q: What is the form of the likelihood function that K-Means approximately maximizes?

K-Means → Gaussian Mixture Models

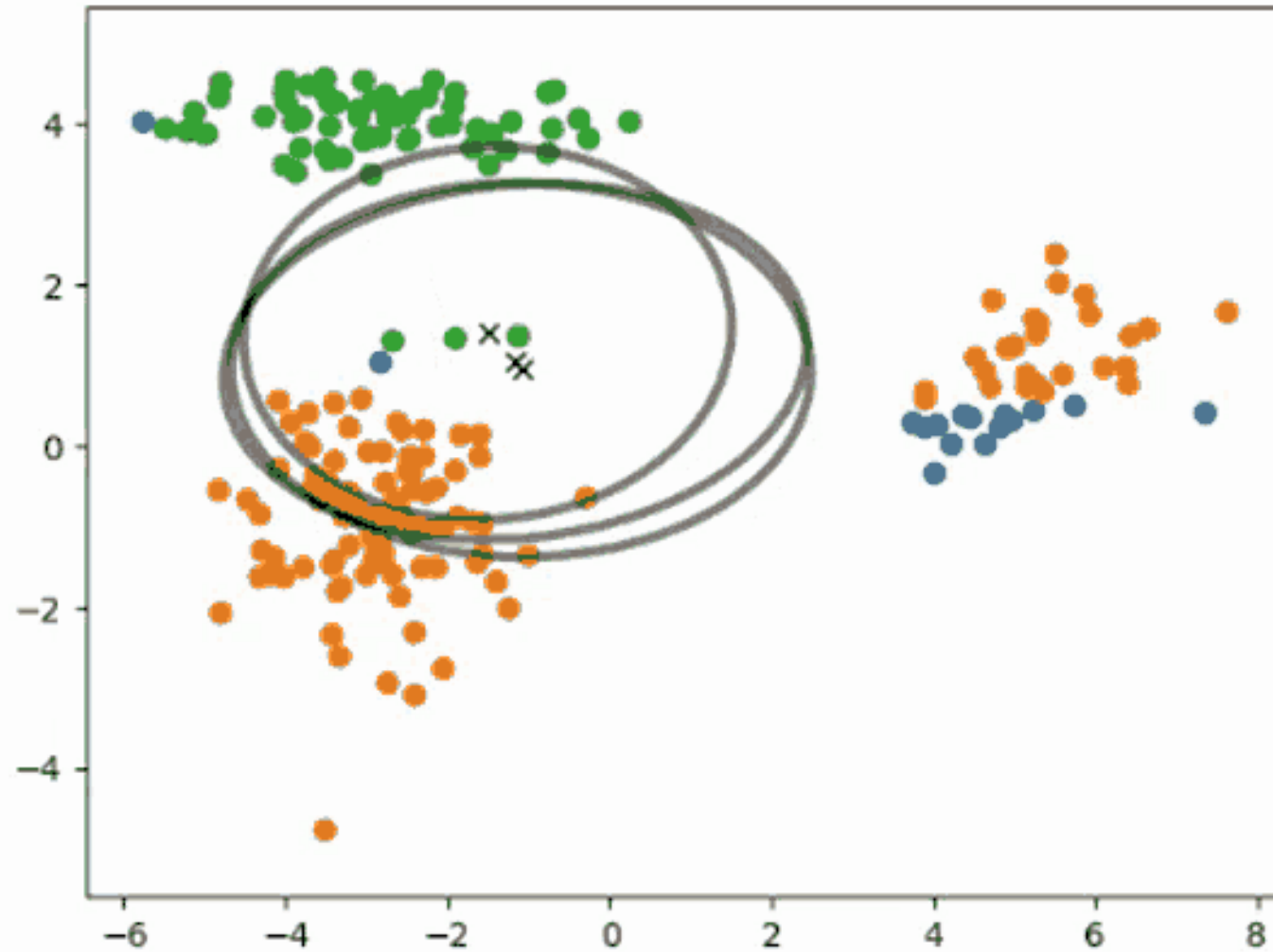
KMeans Iteration:



EM Iteration:



GMMs applied to non-spherical clusters



From GMMs to Variational Autoencoders

What About Much More Complex Distributions?

Images, text, audio, etc.?



From GMMs to Variational Autoencoders (VAEs)

- Sample a “latent” $z \sim p_\theta(z)$ = a categorical distribution over 1, 2, ..., K
- Then sample $x \sim p_\theta(x|z = k) = \mathcal{N}(x; \mu_\theta^k, \Sigma_\theta^k)$

- Sample a latent $z \sim \mathcal{N}(0, I)$.
- Then sample $x \sim p_\theta(x|z) = \mathcal{N}(x; \mu_\theta(z), \Sigma_\theta(z))$, where $\mu_\theta(\cdot)$ and $\Sigma_\theta(\cdot)$ are deep neural networks!
 - In particular, typically one neural network maps an input z to output $\mu_{j=1,\dots,d}, \sigma_{j=1,\dots,d}$ for all data dimensions j

A mixture of infinitely many
gaussians!

GMMs

VAEs

From GMMs to Variational Autoencoders (VAEs)

- Sample a “latent” $z \sim p_\theta(z)$ = a categorical distribution over 1, 2, ..., K
- Then sample $x \sim p_\theta(x|z = k) = \mathcal{N}(x; \mu_\theta^k, \Sigma_\theta^k)$

$$p_\theta(x) = \sum_{k=1}^K p_\theta(z = k) \mathcal{N}(x; \mu_\theta^k, \Sigma_\theta^k)$$

GMMs

- Sample a latent $z \sim \mathcal{N}(0, I)$.
- Then sample $x \sim p_\theta(x|z) = \mathcal{N}(x; \mu_\theta(z), \Sigma_\theta(z))$, where $\mu_\theta(\cdot)$ and $\Sigma_\theta(\cdot)$ are deep neural networks!
 - In particular, typically one neural network maps an input z to output $\mu_{j=1,\dots,d}, \sigma_{j=1,\dots,d}$ for all data dimensions j

$$p_\theta(x) = \int_{\mathbf{z}} p_\theta(z) \mathcal{N}(x; \mu_\theta(z), \Sigma_\theta(z)) dz$$

VAEs

Computing the likelihood $P_{\theta}(x)$ under a VAE

$$p_{\theta}(x) = \int_{z \sim \mathcal{Z}} p_{\theta}(z)p_{\theta}(x|z)dz \propto \mathbb{E}_{z \sim \text{uniform}(z)}[p_{\theta}(z)p_{\theta}(x|z)]$$

Integrating over all z is hard. We can approximate by averaging $p_{\theta}(z)p_{\theta}(x|z)$ over some randomly sampled z_1, z_2, \dots, z_M

$$p_{\theta}(x) \approx \frac{1}{M} \sum_i p_{\theta}(z_i)p_{\theta}(x|z_i)$$

In reality for high-dimensional problems like generating images, $p_{\theta}(x|z_i)$ is tiny for almost all values of z , so sampling like this is doomed to fail (requires millions of samples to be a good approximation to $p_{\theta}(x)$).

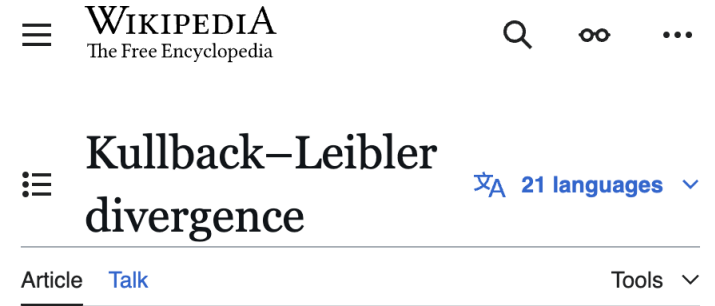
Would be awfully useful to have access to the distribution $p(z|x)$ of latents corresponding to a data point x . Then we could just sample z_i s from it. But we don't ...

The Evidence Lower Bound To Log Likelihood

“ELBO”

$$\begin{aligned} & \mathbb{E}_{q_i(z)} \left[\log \frac{p_\theta(x_i, z)}{q_i(z)} \right] \\ &= \log p_\theta(x_i) + \mathbb{E}_{q(z)} \left[\log \frac{p_\theta(z|x_i)}{q_i(z)} \right] \\ &= \underbrace{\log p_\theta(x_i)}_{\text{Log likelihood}} - \underbrace{D_{KL}(q_i(z) || p_\theta(z|x_i))}_{\text{Positive quantity}} \end{aligned}$$

Any arbitrary distribution



From Wikipedia, the free encyclopedia

In [mathematical statistics](#), the **Kullback–Leibler (KL) divergence** (also called **relative entropy** and **I-divergence**^[1]), denoted $D_{\text{KL}}(P \parallel Q)$, is a type of [statistical distance](#): a measure of how one reference [probability distribution](#) P is different from a second probability distribution Q .^{[2][3]} Mathematically, it is defined as

$$D_{\text{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right).$$

If we maximize this ELBO jointly over θ and all the $q_i(\cdot)$ s, then:
each $q_i(\cdot)$ would naturally approximate the posterior $p_\theta(z|x_i)$!
(Can you see why?)

Log likelihood \geq ELBO
(=ELBO for $q_i(z) = p_\theta(z|x_i)$)

Expanding the ELBO objective

$$\begin{aligned}\text{ELBO} &= \mathbb{E}_{q_i(z)} \left[\log \frac{p_\theta(x_i, z)}{q_i(z)} \right] \\ &= \mathbb{E}_{q_i(z)} \left[\log \frac{p_\theta(z)p_\theta(x_i|z)}{q_i(z)} \right] \\ &= \mathbb{E}_{q_i(z)} \left[\log \frac{p_\theta(z)}{q_i(z)} \right] + \mathbb{E}_{q_i(z)} [\log p_\theta(x_i|z)] \\ &= \underbrace{-D_{KL}(q_i(z) || p_\theta(z))}_{\text{KL divergence from prior}} + \underbrace{\mathbb{E}_{q_i(z)} [\log p_\theta(x_i|z)]}_{\text{“reconstruction loss”}}\end{aligned}$$

“Amortized” Variational Inference

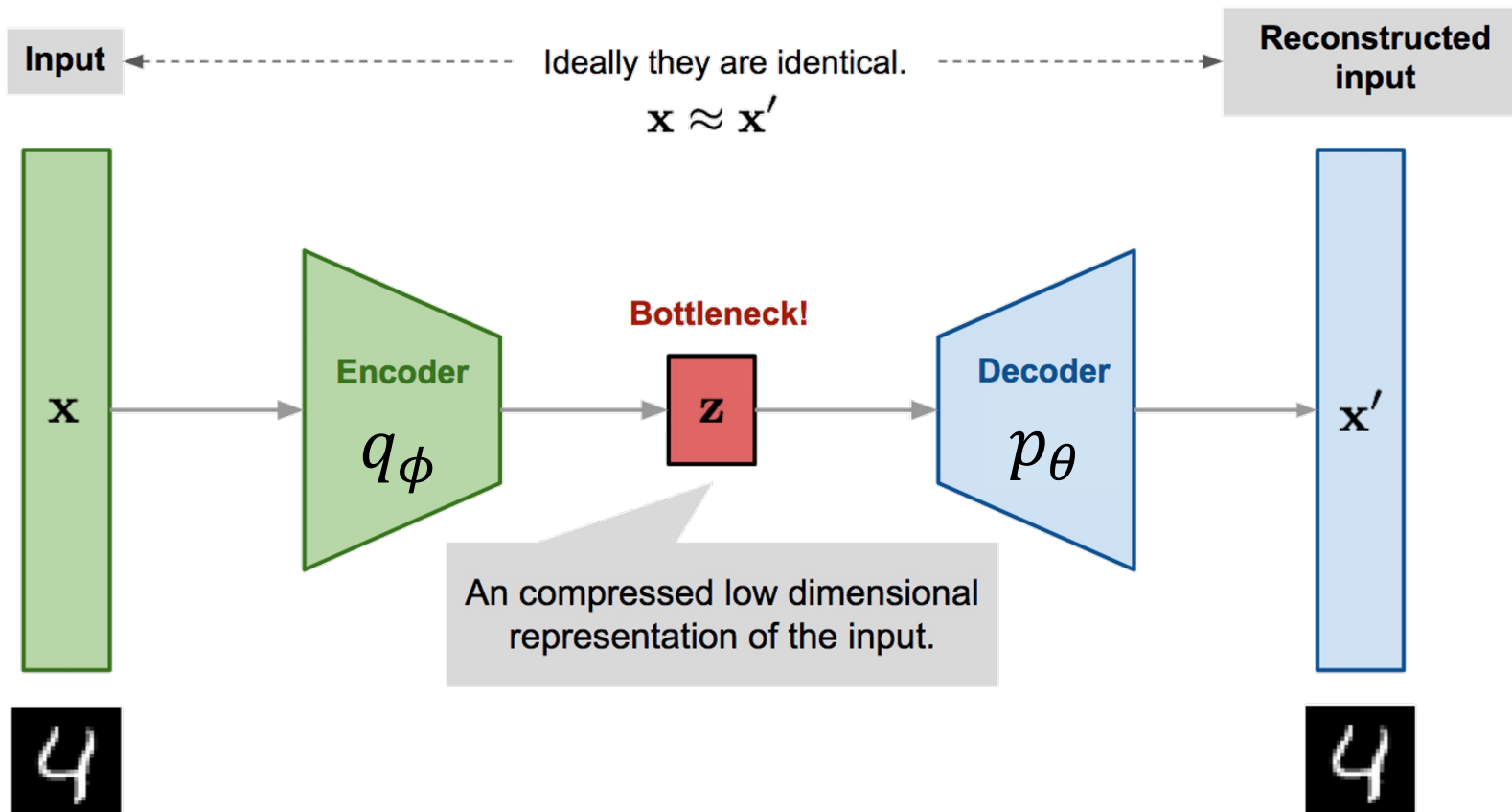
- Note that we were optimizing over not just the neural network parameters θ , but also over a separate distribution $q_i(z)$ for each training point!
- If $q_i(z)$ is even a simple distribution like a diagonal covariance gaussian, this means $2d$ parameters for each sample (assuming d -dimensional latents)
- If we train over lots of high-dimensional data points (e.g. images), typically requiring large latent dimension d), this becomes a problem.
- A simple solution is “amortized” inference. Rather than a look-up table $q_i(z)$ that looks up the appropriate distribution for each training sample i , we use a neural network with new parameters ϕ to perform this mapping $q_i(z) \leftarrow q_\phi(z|x_i)$

Final Form of VAE

Often set to $\mathcal{N}(0, I)$

$$\text{VAE training loss: } \sum_{x_i \in \mathcal{D}} -D_{KL} \left(q_{\phi}(z|x_i) || p_{\theta}(z) \right) + \mathbb{E}_{q_{\phi}(z|x_i)} [\log p_{\theta}(x_i|z)]$$

Often approximated with one sample during SGD



Sampling new x only requires the decoder.

Evaluating $P(x)$ for a sample? Use the ELBO, same as used in training objective.

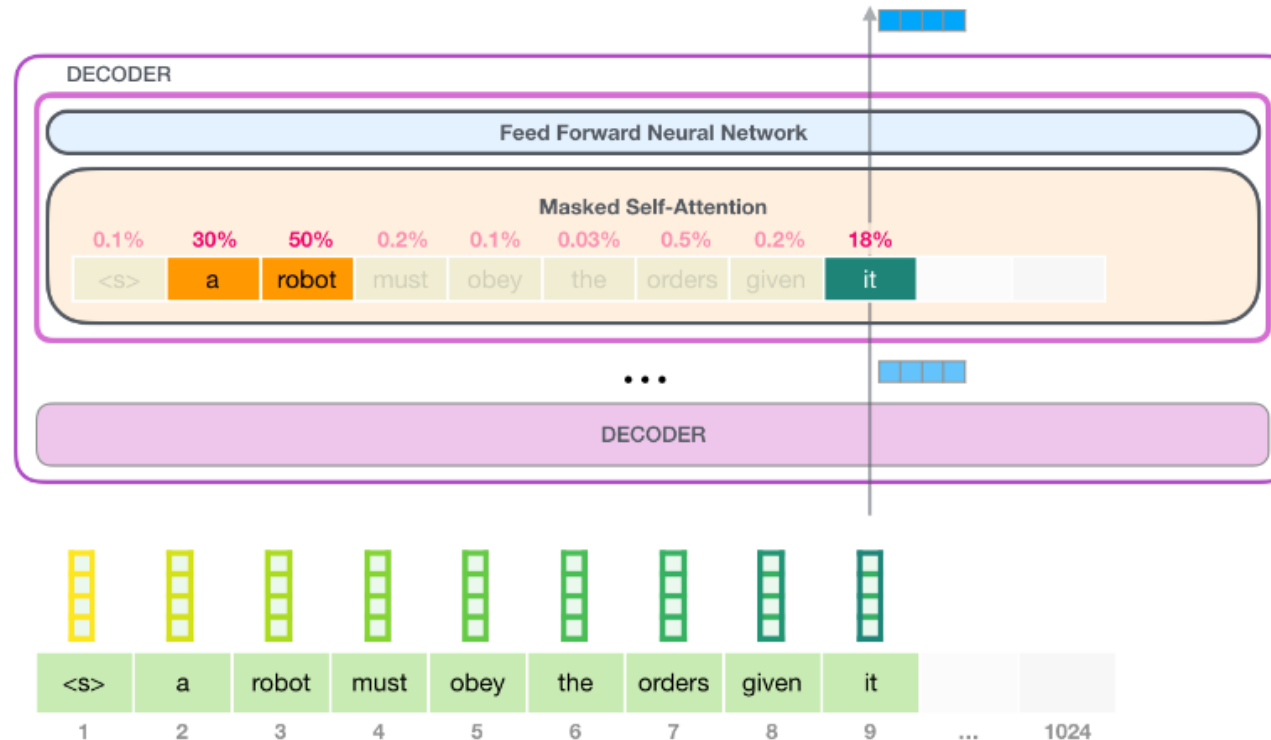
Lots of models springing from / connected to VAEs

- “Conditional” VAEs (where rather than modeling $p(x)$, we are modeling $p(x|y)$, e.g. to generate an image conditioned on a text input describing the image)
- “Vector Quantized” VAEs (where the latent is restricted to be one of a discrete set of vectors)
- “Diffusion Models” (where there is not one but many levels of latents: $z_1 \rightarrow z_2 \rightarrow \dots \rightarrow z_T = x$, and there is a very specific kind of posterior $p(z_i|x)$ produced by adding noise to x .)
- ...

Autoregressive models e.g. the GPT family

Autoregressive models recursively generate the next item (e.g. a word in a generated sentence), conditioned on the previous output(s):

$$p(x = [x_1, x_2, \dots, x_k]) = \prod_{i=1, \dots, k} p(x_i | x_{<i})$$



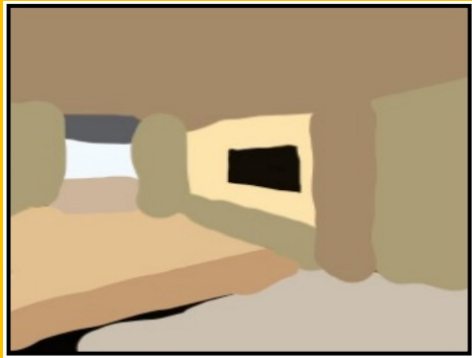
Applications!

Pretty Pictures and Beyond

The rest of this slide deck was not covered in class

- These slides are provided here for your reference. They will not be tested unless covered in a later lecture / homework, and are colored differently to indicate this.

Data generation in the real world



Generative model
of realistic images



Stroke paintings to realistic images
[Meng, He, Song, et al., ICLR 2022]

“Ace of Pentacles”



Generative model
of paintings



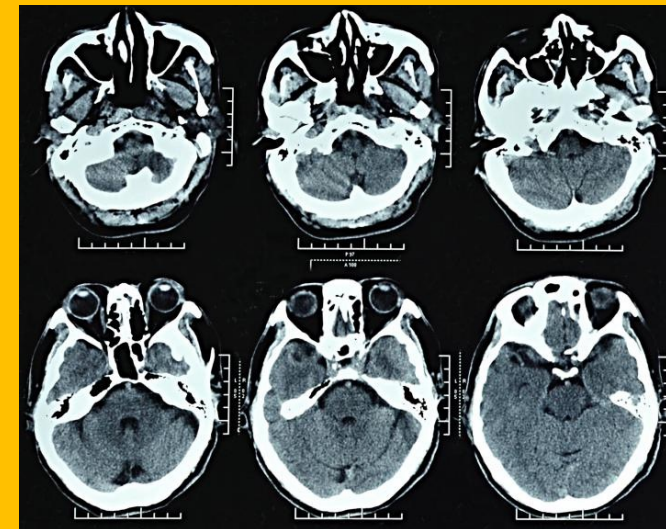
Language-guided artwork creation
<https://chainbreakers.kath.io> @RiversHaveWings

Solving inverse problems with generative models



Generative model
of medical images

Generate



Medical image reconstruction

[Song et al., ICLR 2022]

Outlier detection with generative models



High
probability
→



←
Low
probability

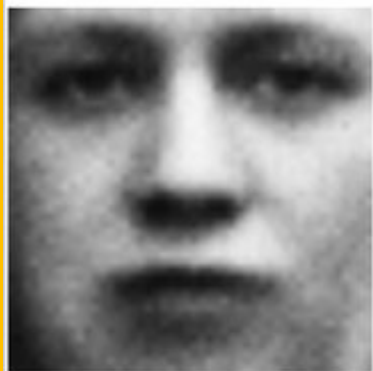


Generative model
of traffic signs



Outlier detection
[Song et al., ICLR 2018]

Progress in Generative Models of Images -- GANs



2014



2015



2016



2017



2018

Ian Goodfellow, 2019

Progress in Generative Models of Images – Diffusion Models



Song et al., Score-Based Generative Modeling through Stochastic Differential Equations, 2021

Slide courtesy: Stefano Ermon

Text2Image Diffusion Models

User input:

An astronaut riding a horse



Text2Image Diffusion Models

User input:

A perfect Italian meal



Text2Image Diffusion Models

User input:

泰迪熊穿着戏服，站在太和殿前唱京剧

A teddy bear, wearing a costume, is standing in front of the Hall of Supreme Harmony and singing Beijing opera



Dalle3

A minimap diorama of a cafe adorned with indoor plants. Wooden beams crisscross above, and a cold brew station stands out with tiny bottles and glasses



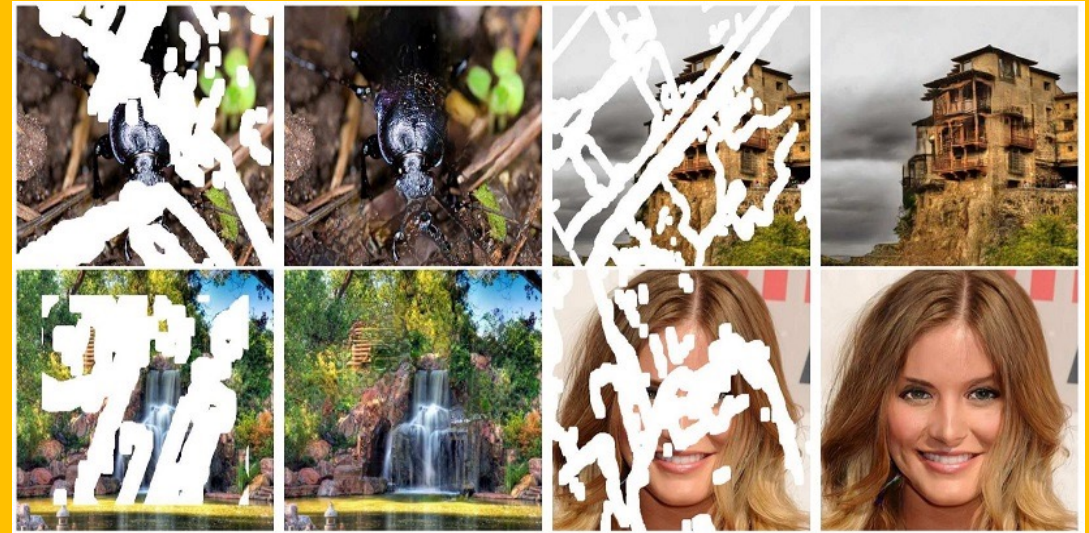
Progress in Inverse Problems

$P(\text{high resolution} \mid \text{low resolution})$



Menon et al, 2020

$P(\text{full image} \mid \text{mask})$



Liu et al, 2018

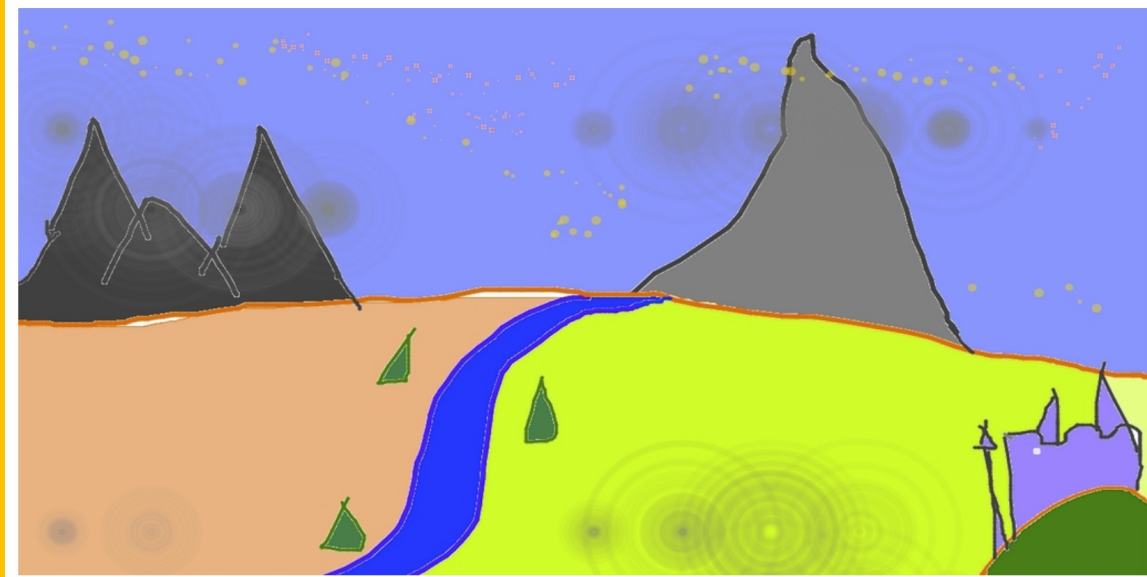
$P(\text{color image} \mid \text{greyscale})$



Antic, 2020

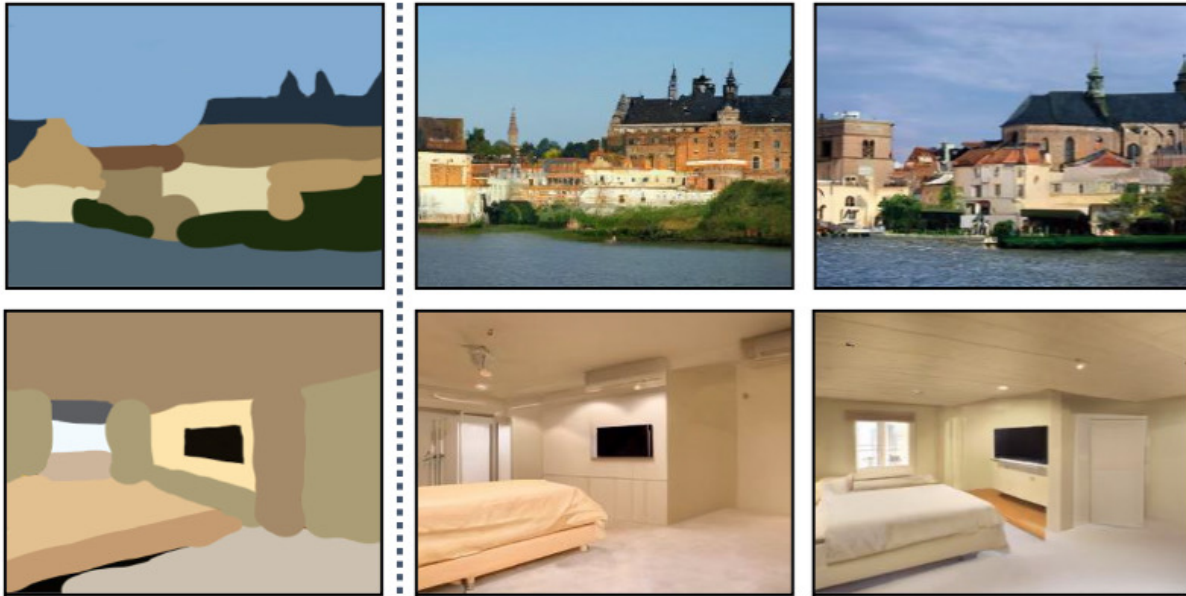
Progress in Inverse Problems

User input:



Progress in Inverse Problems

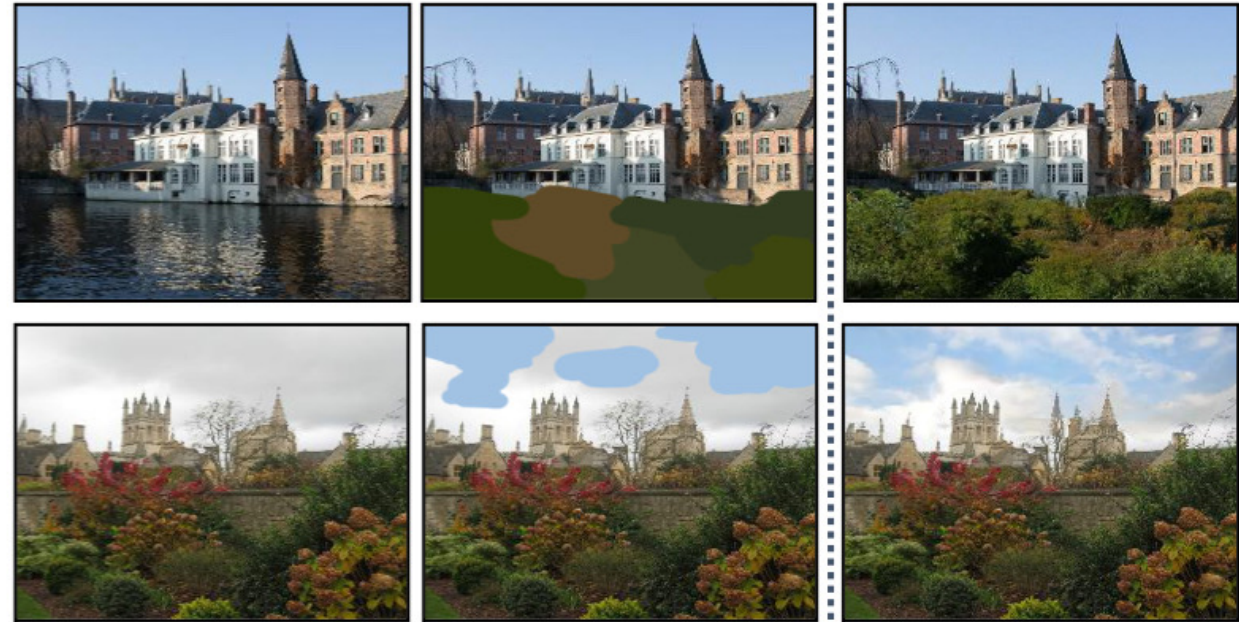
Stroke Painting to Image



Input

Output

Stroke-based Editing



Progress in Inverse Problems

Input Image



Edited Image



“A bird spreading wings”

Input Image



Edited Image



“Two kissing parrots”

Input Image



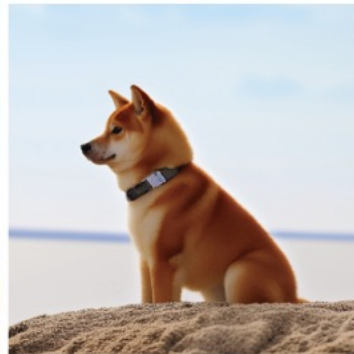
Edited Image



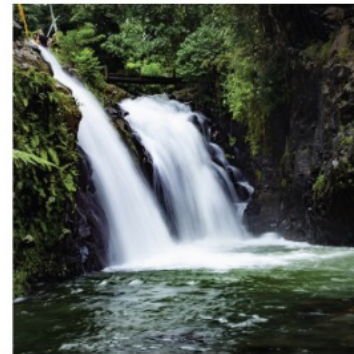
“A goat jumping over a cat”



“A photo of an open box”



“A photo of a sitting dog”

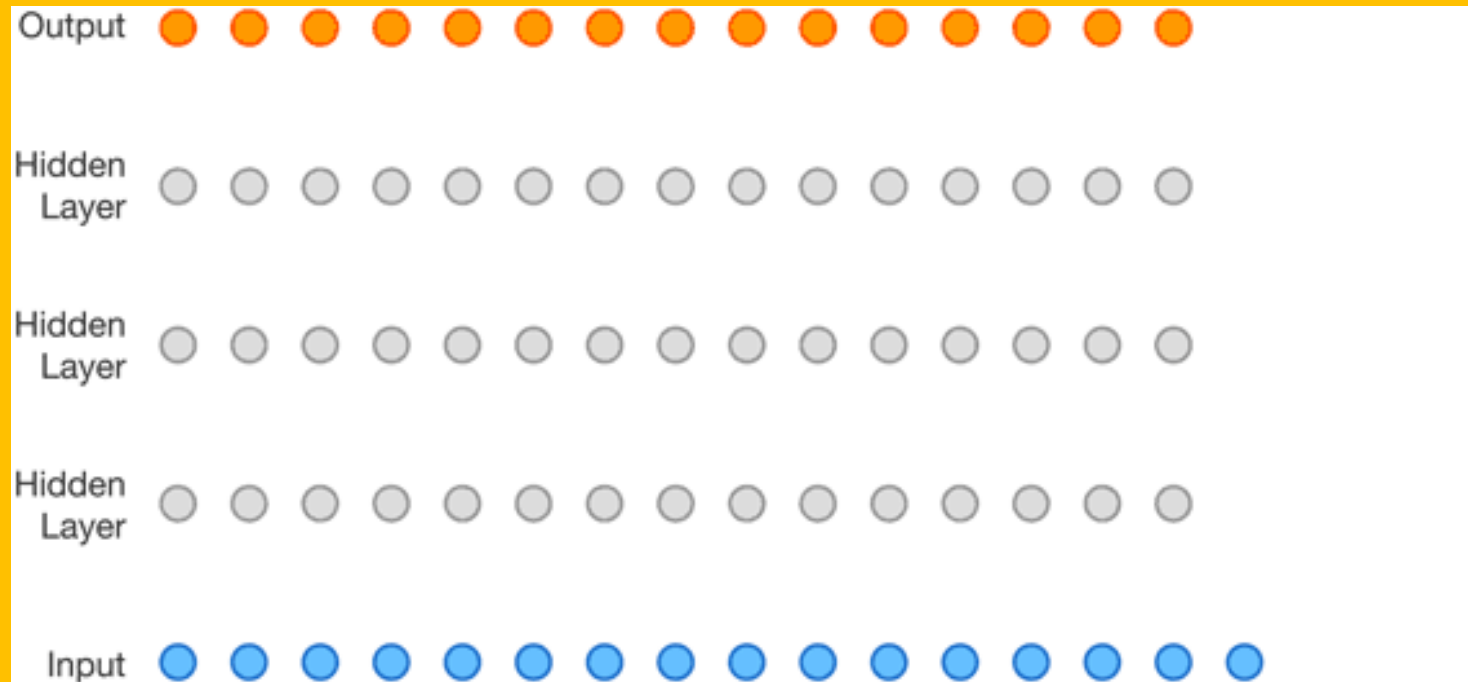


“A children’s drawing of a waterfall”

Kawar et al., 2023

WaveNet

Generative model of speech signals



Text to Speech



Parametric



Concatenative



WaveNet



Unconditional



Music

van den Oord et al, 2016c

Diffusion Text2Speech

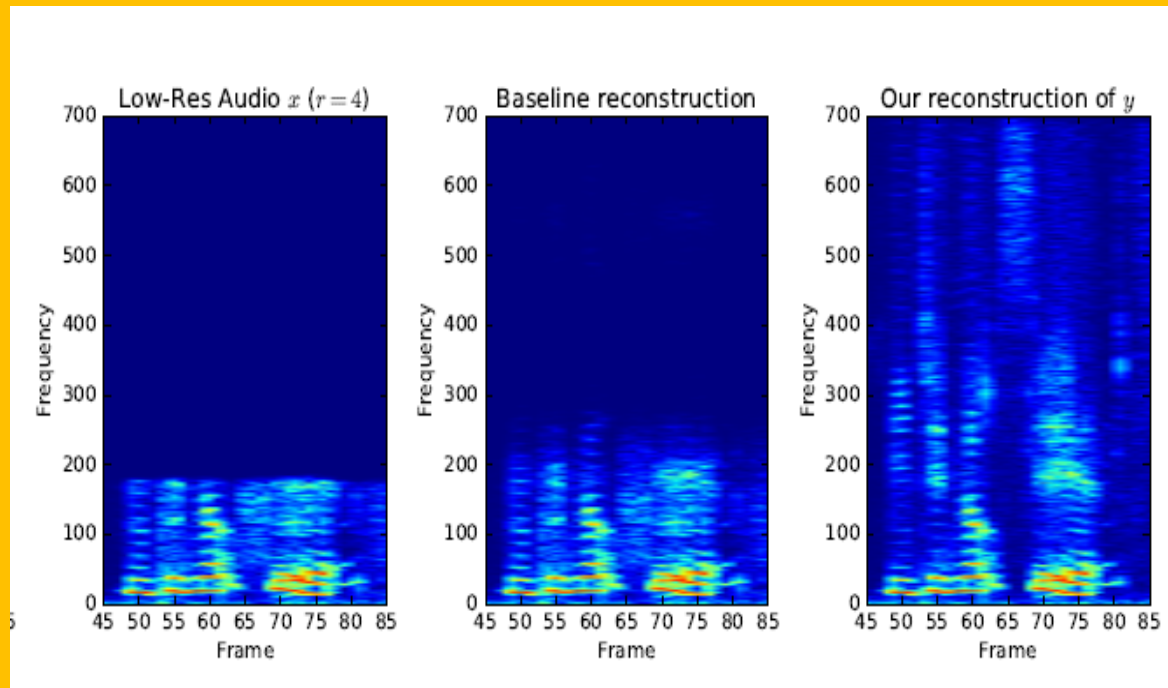
Generative model of speech signals



Betker, Better speech synthesis through scaling 2023

Audio Super Resolution

Conditional generative model $P(\text{high-res signal} \mid \text{low-res audio signal})$



Low res signal



High res audio signal

Kuleshov et al., 2017

Language Generation

Custom prompt

To get an A+ in deep generative models, students have to

Completion

To get an A+ in deep generative models, students have to be willing to work with problems that are a whole lot more interesting than, say, the ones that most students work on in class. If you're a great student, the question above can be avoided and you'll be able to do great work, but if you're not, you will need to go beyond the basics before getting good.

Now to be clear, this advice is not just for the deep-learning crowd; it is good advice for any student who is taking his or her first course in machine learning.

The key point is that if you have a deep, deep brain of a computer scientist, that's just as important to you.

$P(\text{next word} \mid \text{previous words})$

Radford et al., 2019

Demo from talktotransformer.com

Language Generation -- ChatGPT



What should I do to get an A+ in CS236 at Stanford?

To get an A+ in CS236 (Deep Generative Models) at Stanford, you will need to excel in both your understanding of the material and your performance in assignments and exams. Here are some general tips to help you achieve this:

1. **Attend Lectures and Engage Actively**: Attend all lectures and actively engage with the material. Take thorough notes, ask questions, and participate in discussions. This will help you understand the concepts better.
2. **Read the Assigned Material**: Make sure to read the assigned textbooks, papers, and supplementary materials. Understanding the theoretical foundations is crucial.
3. **Stay Organized**: Keep a well-organized notebook or digital notes. This will help you quickly review and understand the material.
4. **Seek Help When Needed**: Don't hesitate to ask questions if you're having trouble with a concept. You can ask the professor, teaching assistants, or classmates for clarification.
5. **Complete Assignments Thoroughly**: Take your time to complete assignments, ensuring you fully understand the requirements and concepts involved. Start early and seek help if you're stuck. High-quality assignments are often a significant portion of your grade.
6. **Collaborate, but Don't Plagiarize**: Collaboration is often encouraged, but make sure you understand your institution's policy on collaboration and plagiarism. Always give credit where it's due and submit original work.
- ...
15. **Meet with the Professor**: If you're aiming for an A+, consider scheduling meetings with the professor to discuss your progress and seek feedback.

Remember that getting an A+ can be highly competitive, and the specific grading criteria may vary from one course to another and one professor to another. It's essential to understand the grading policies and expectations of your instructor. Always aim for excellence, but also keep in mind that learning and understanding the material should be your primary goal. Good luck!

Machine Translation

Conditional generative model $P(\text{English text} | \text{Chinese text})$

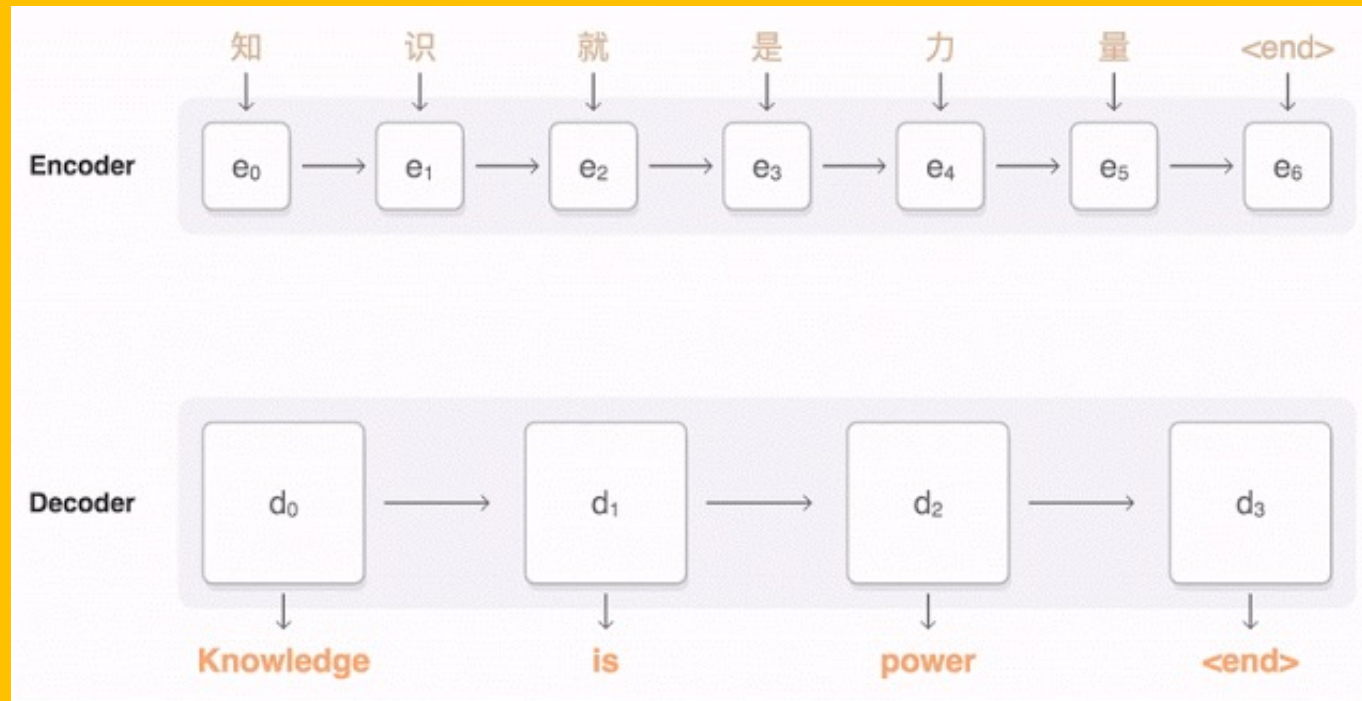
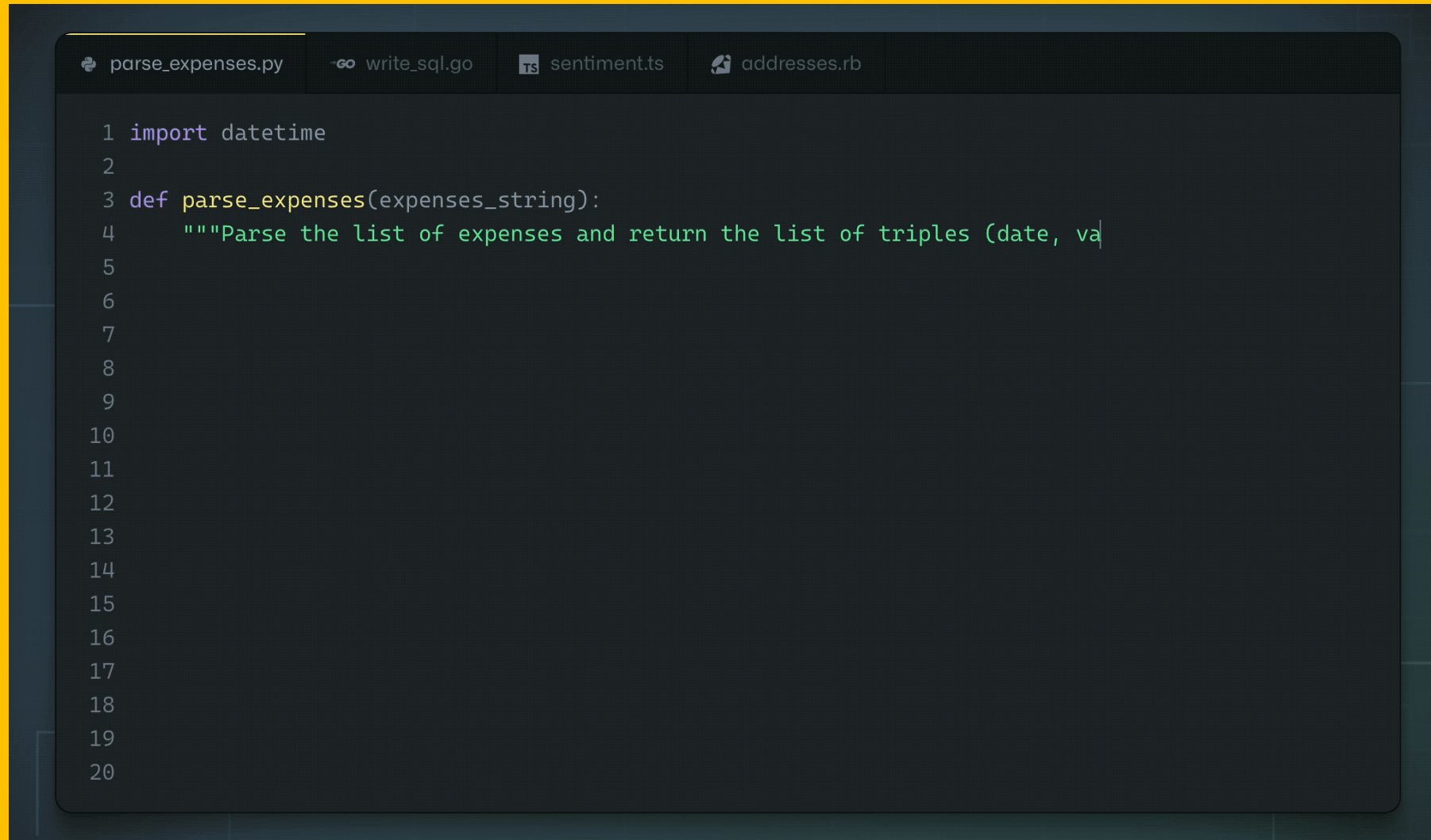


Figure from Google AI research blog.

Code Generation



The image shows a code editor window with a dark theme. At the top, there are four tabs: 'parse_expenses.py' (selected), 'write_sql.go', 'sentiment.ts', and 'addresses.rb'. The main area contains Python code for a function named 'parse_expenses'. The code is as follows:

```
1 import datetime
2
3 def parse_expenses(expenses_string):
4     """Parse the list of expenses and return the list of triples (date, va
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

Video Generation

Suddenly, the walls of the embankment broke and there was a huge flood



Video Generation

a couple sledding down a snowy hill on a tire
roman chariot style

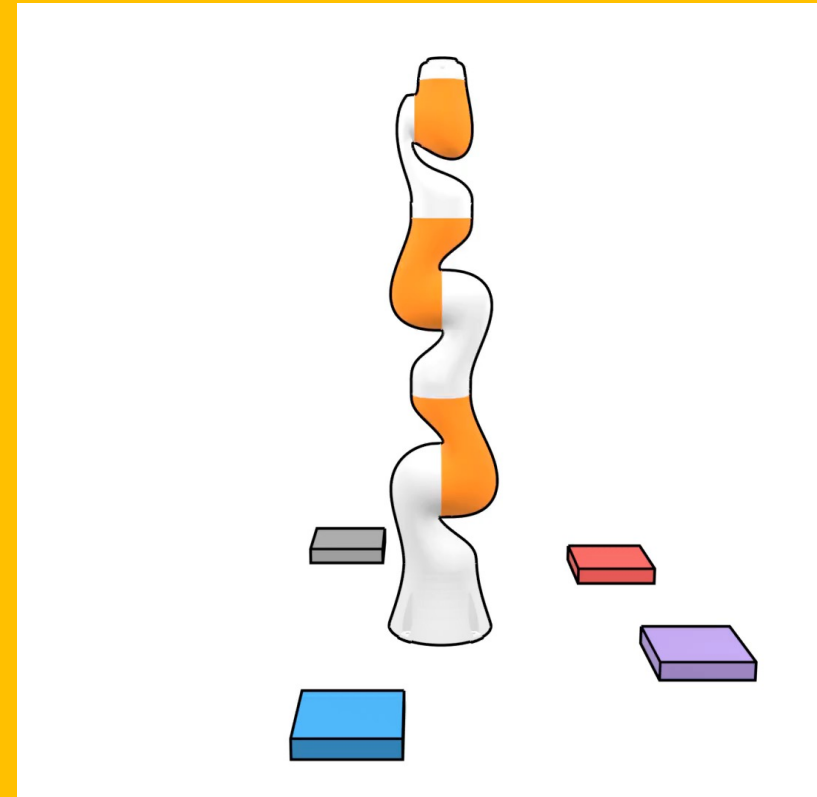


Video Generation

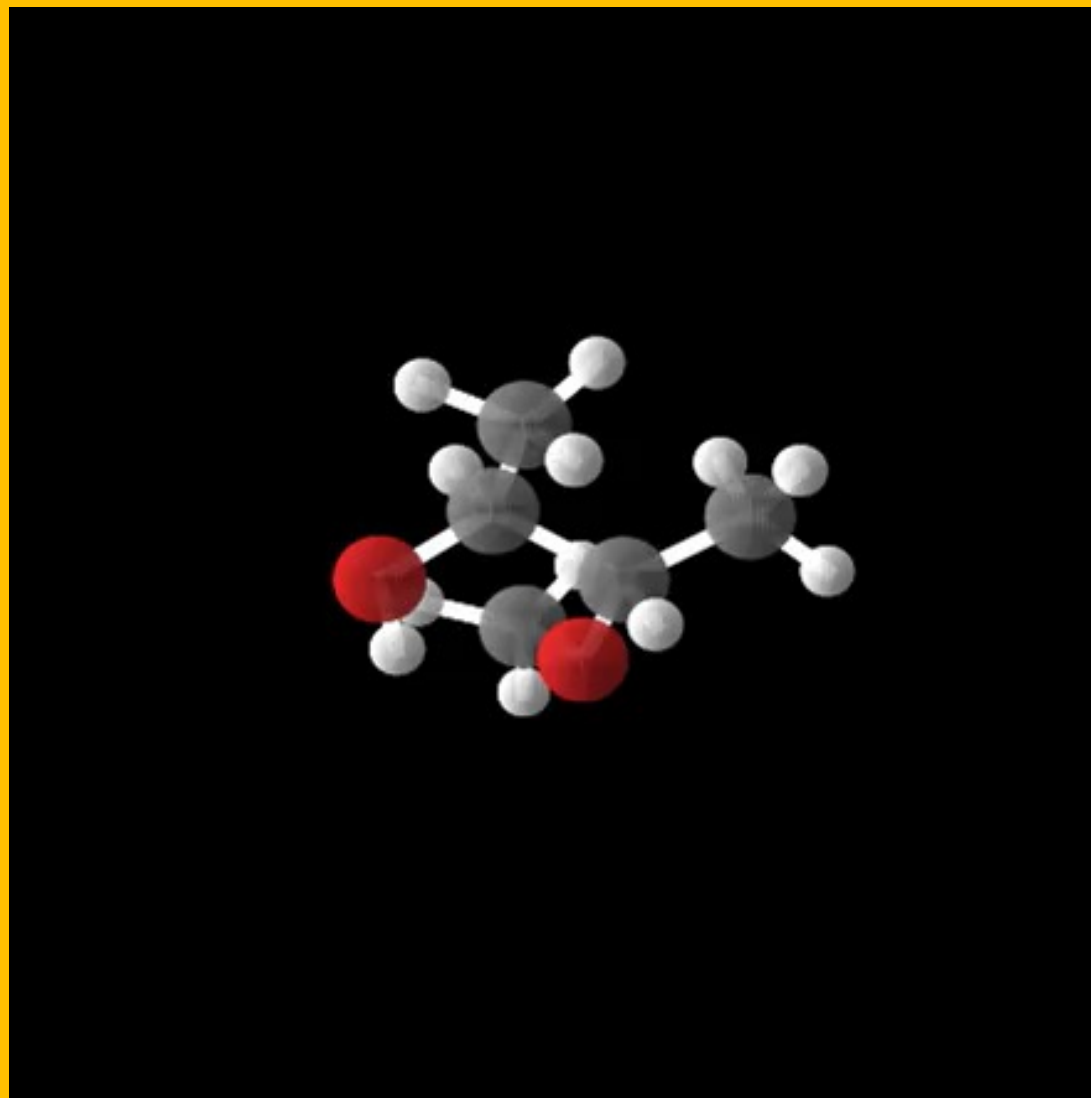


Imitation Learning

Conditional generative model $P(\text{actions} \mid \text{past observations})$

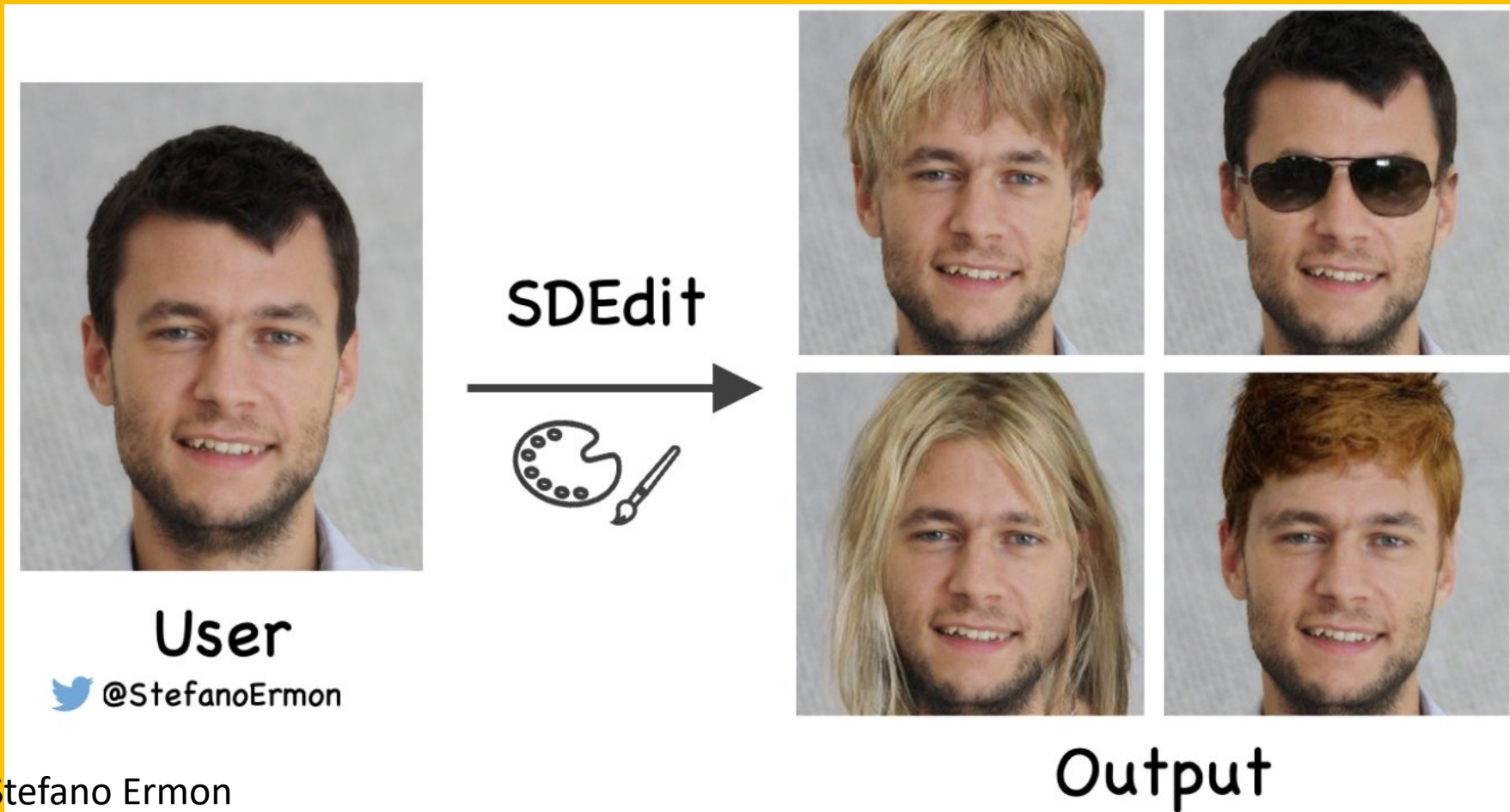


Molecule generation



DeepFakes

Which image is real?



DeepFakes

