GDC

Towards Deep Generative Models in Game Development

Jorge del Val
Research Engineer / SEED (Electronic Arts)

SEED // SEARCH FOR EXTRAORDINARY EXPERIENCES DIVISION

GAME DEVELOPERS CONFERENCE
MARCH 18–22, 2019 | #GDC19

SEED

Agenda

1. Motivation and fundamentals
2. Variational autoencoders (VAE)
3. Generative adversarial networks (GAN)
4. Conditional generative models
5. Some applications to game development

In a sentence…

Models that generate or remix stuff

In a better sentence…

Models that learn the data probability distribution and are able to sample from it

But… why in games?

(Thanks A. Opara ☺)

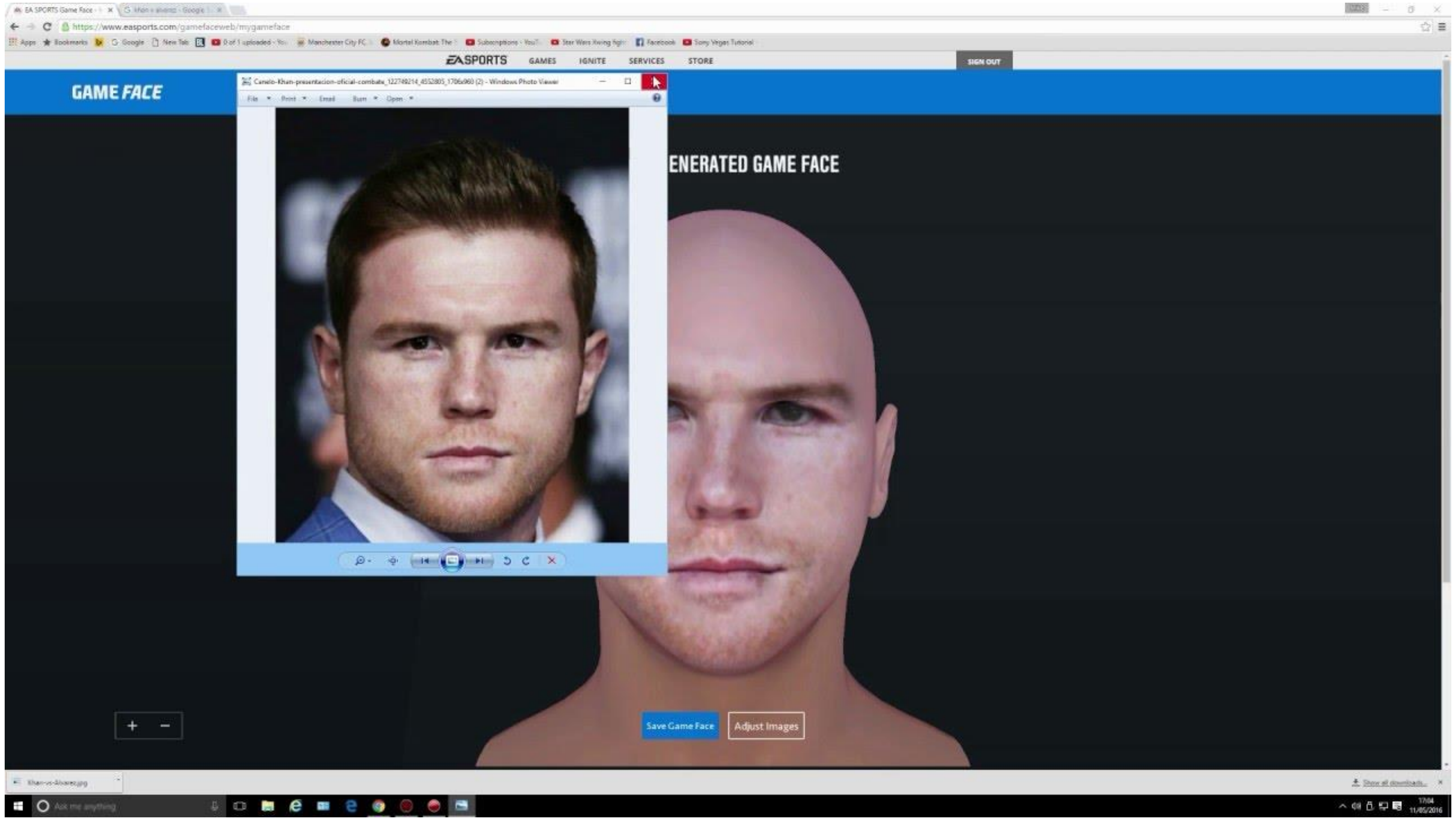

GAME FACE
ENERATED GAME FACE
Save Game Face
Adjust Images

Photo Wake-Up: 3D Character Animation from a Single Photo. Weng et al. 2018

Which is real?

A Style-Based Generator Architecture for Generative Adversarial Networks. Karras et al. 2018 (NVIDIA)

Which is real?

FAKE

FAKE

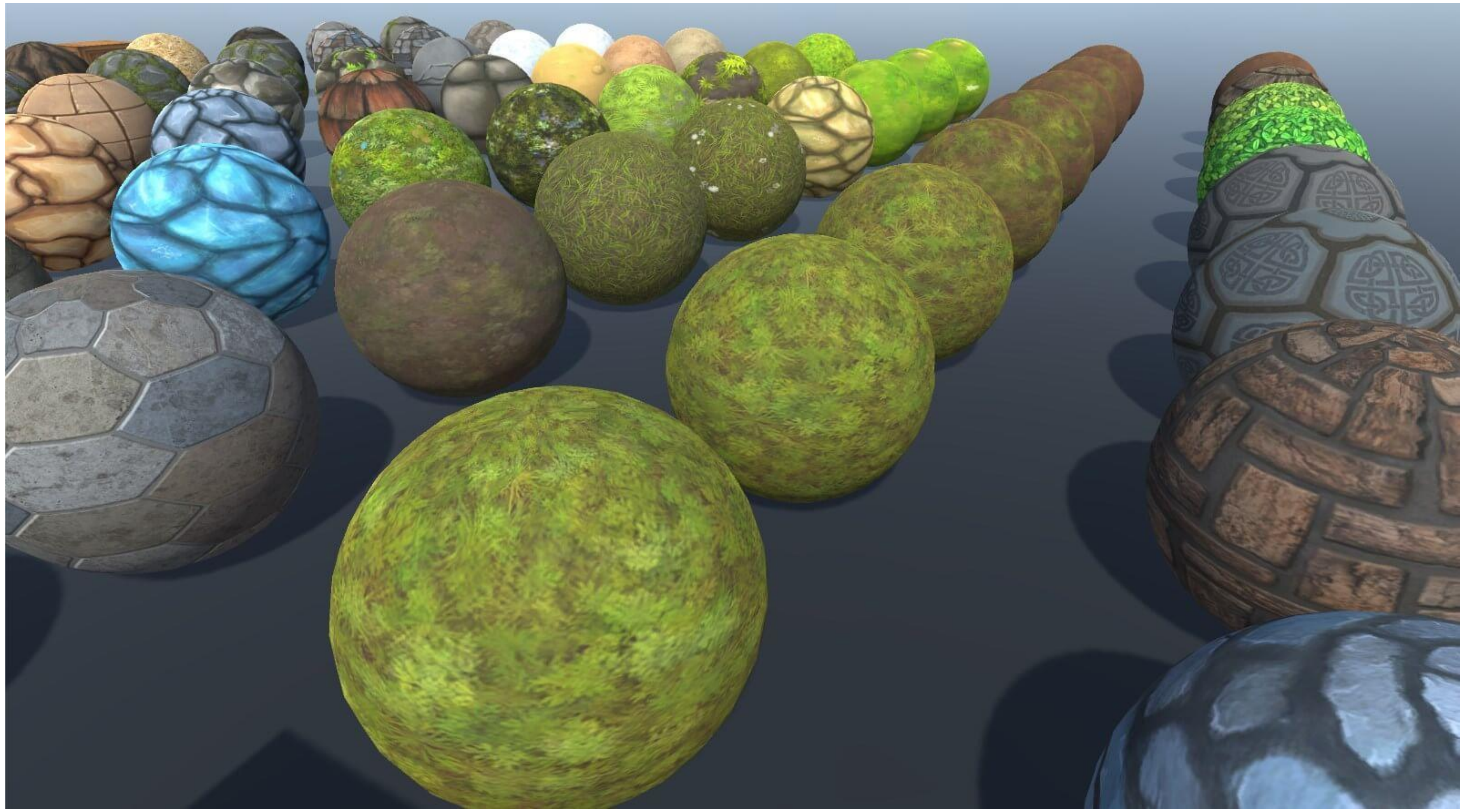
REAL

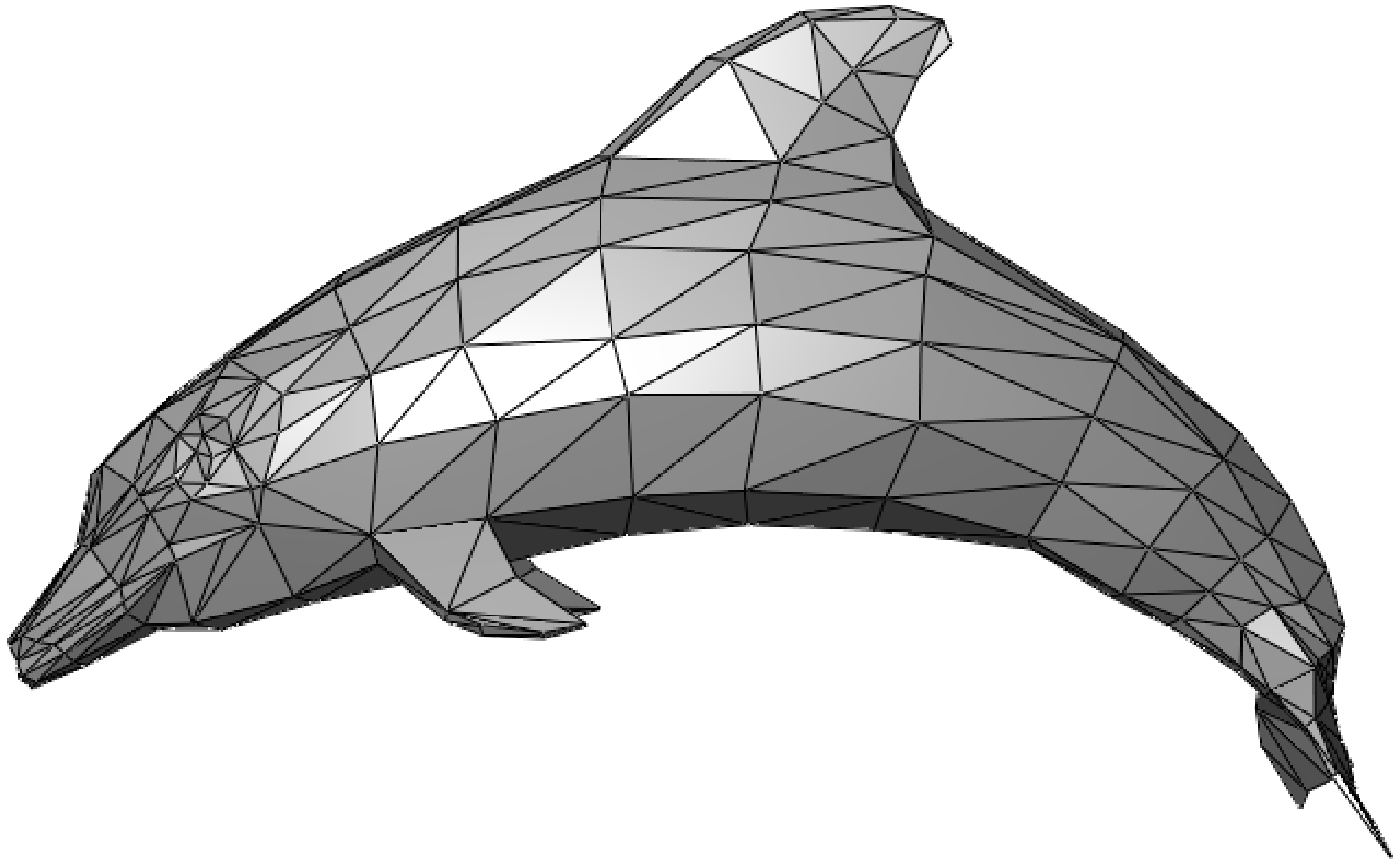
A Style-Based Generator Architecture for Generative Adversarial Networks. Karras et al. 2018 (NVIDIA)

https://thispersondoesnotexist.com

(A Style-Based Generator Architecture for Generative Adversarial Networks. Karras et al. 2018)

So what do they actually do?





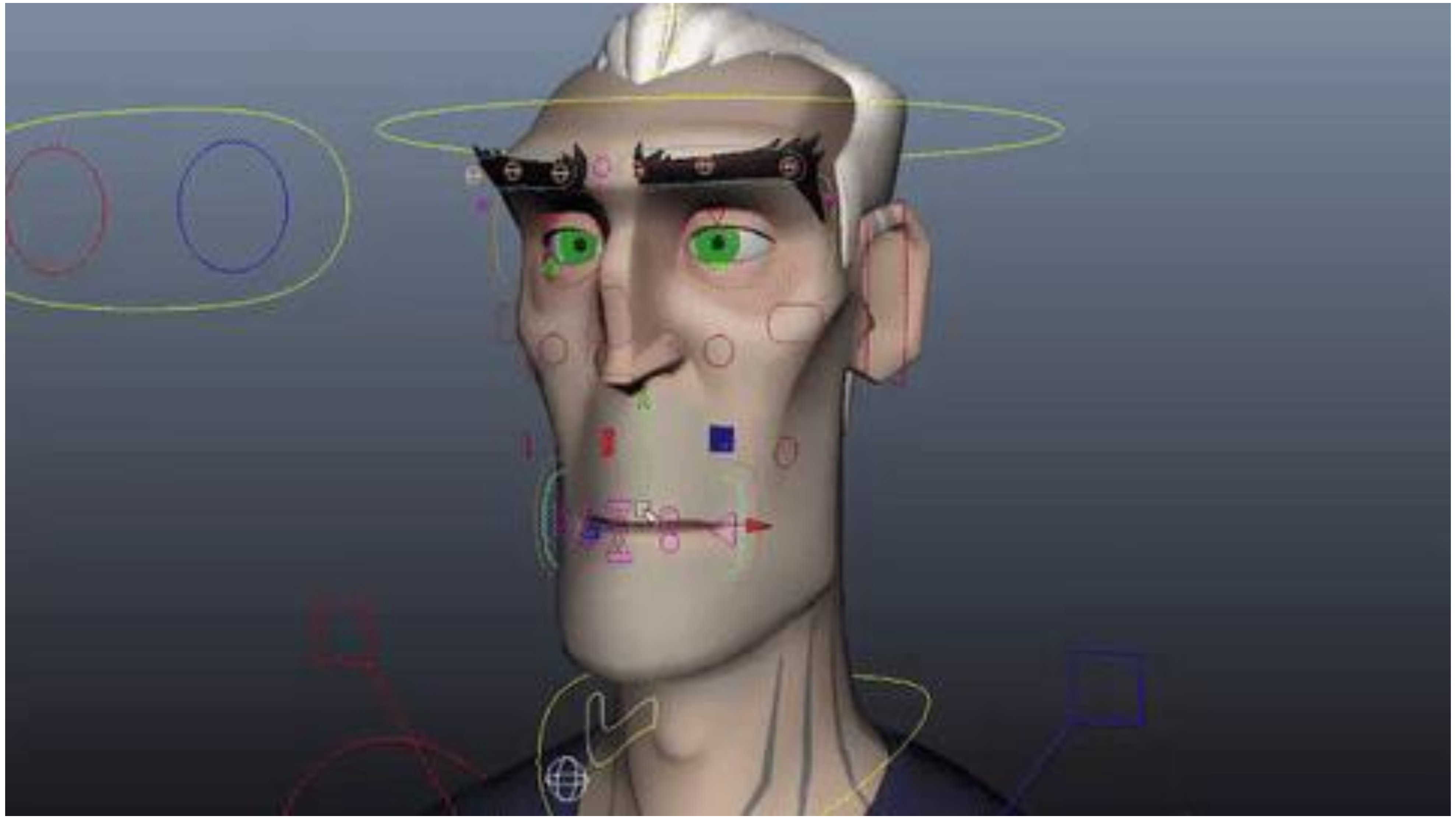
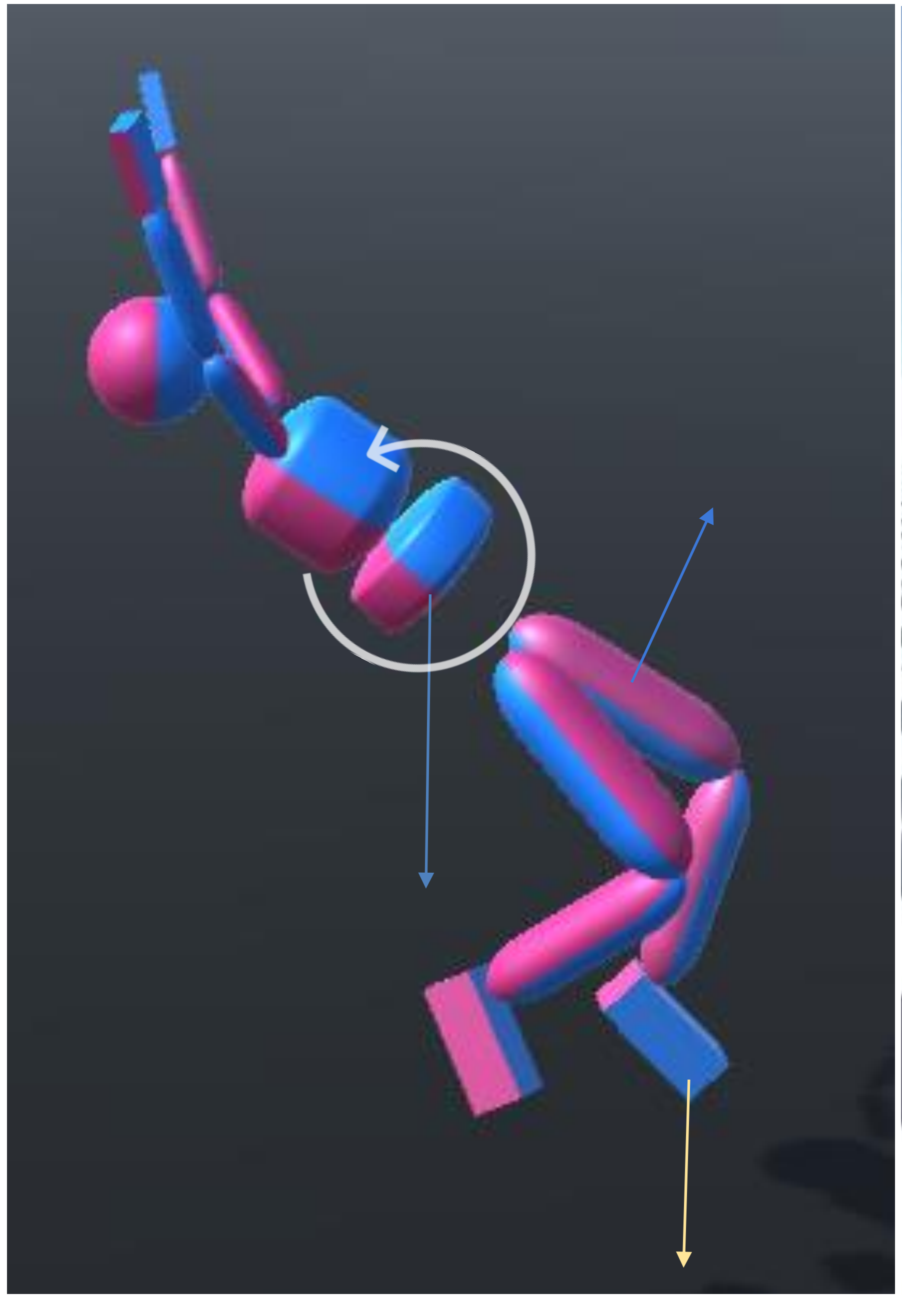
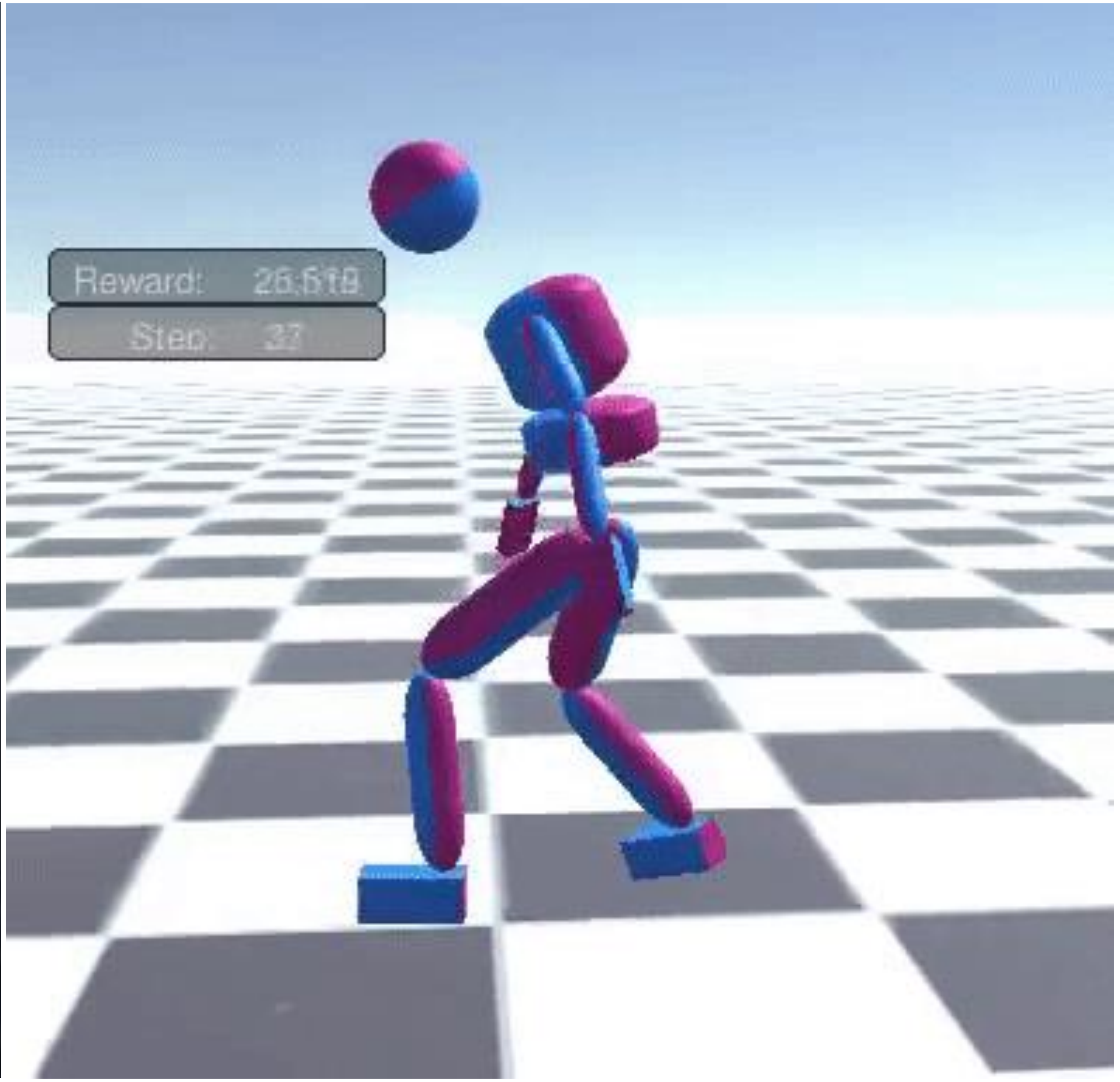


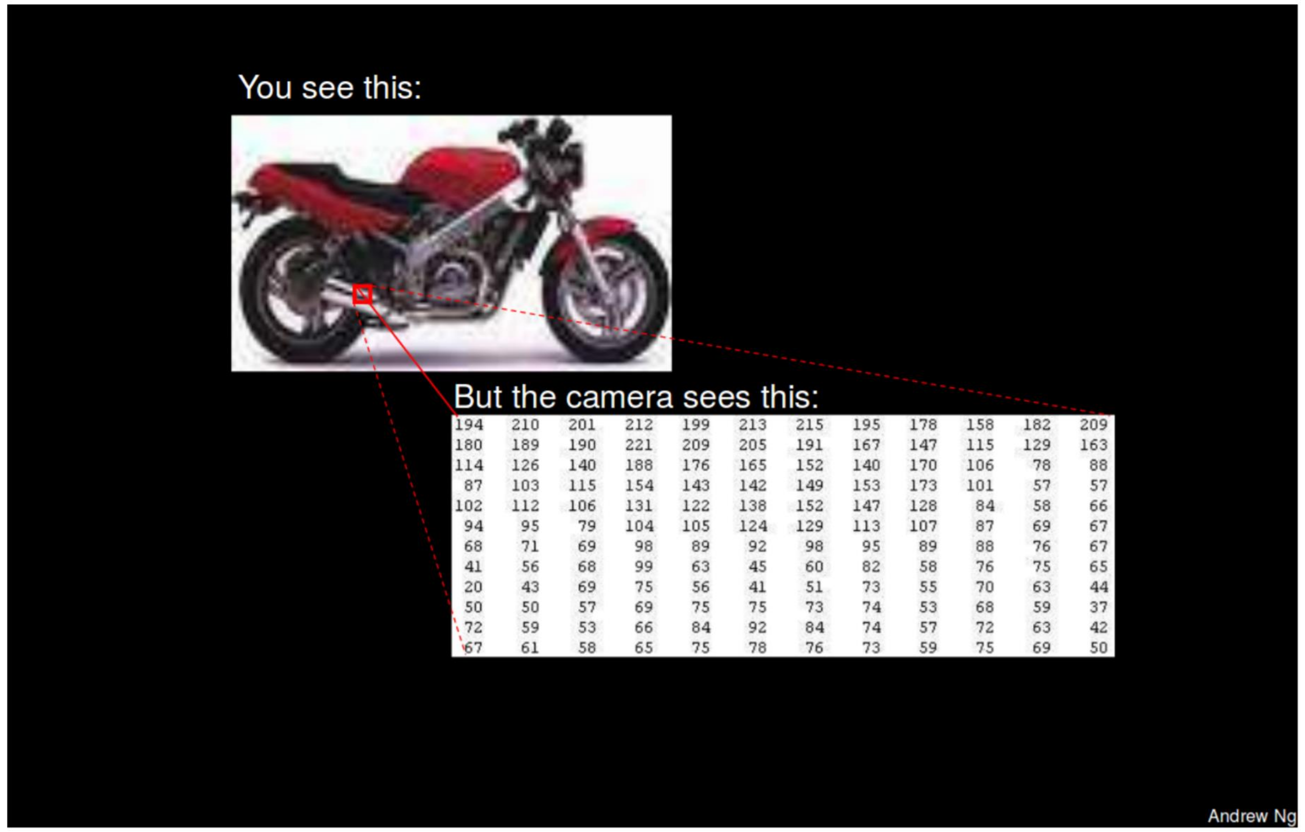
Image credit Animation Mentor

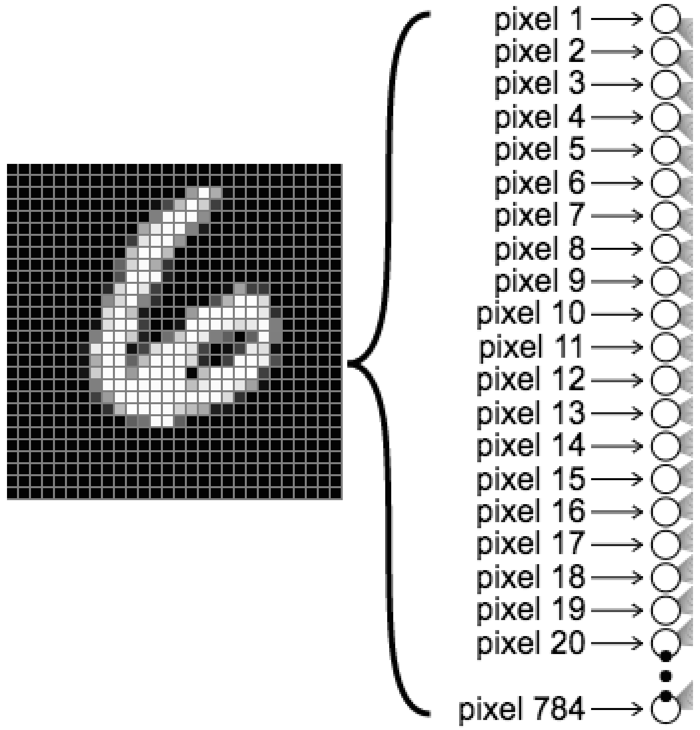




Reward:
Step:

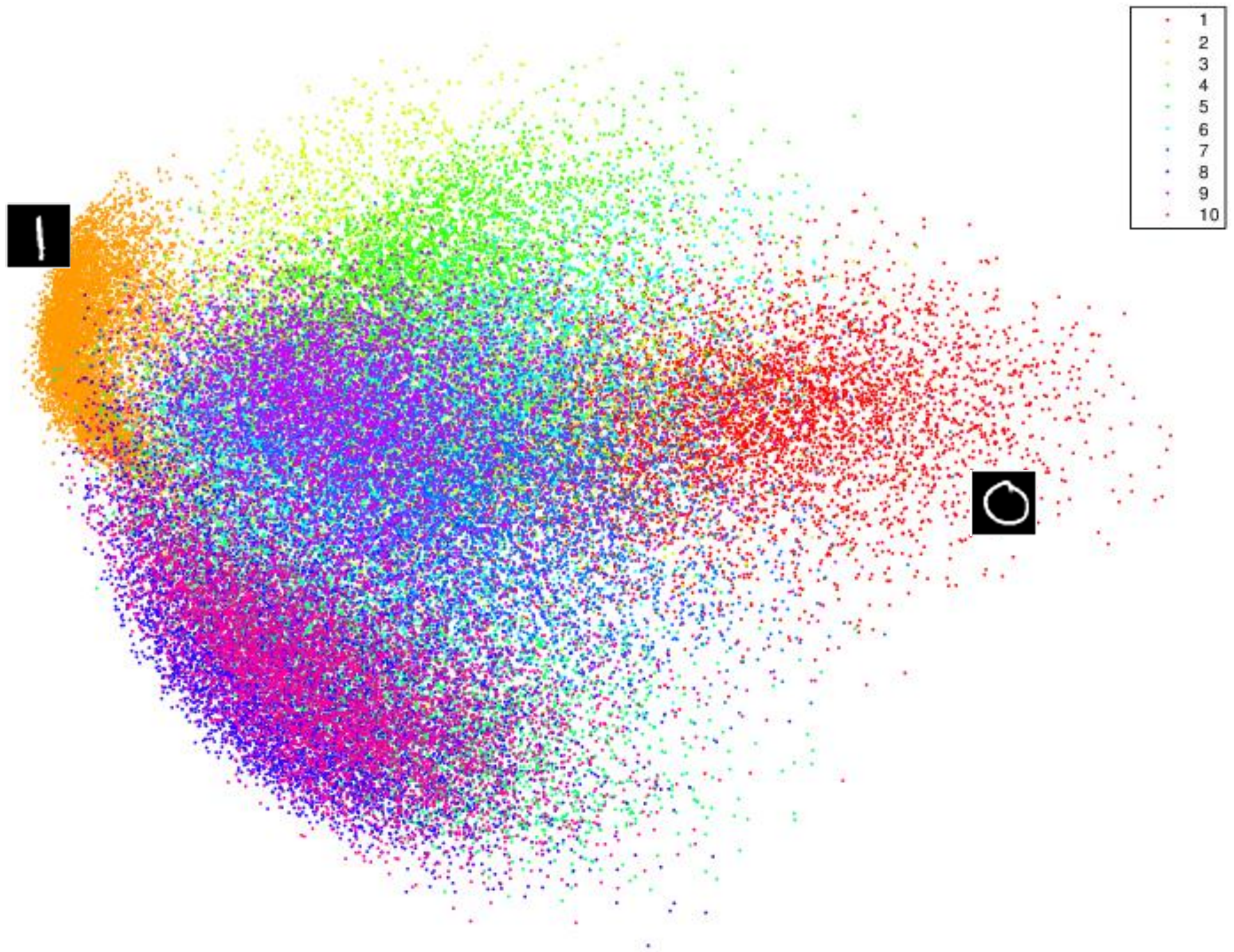
In the end… It's all numbers

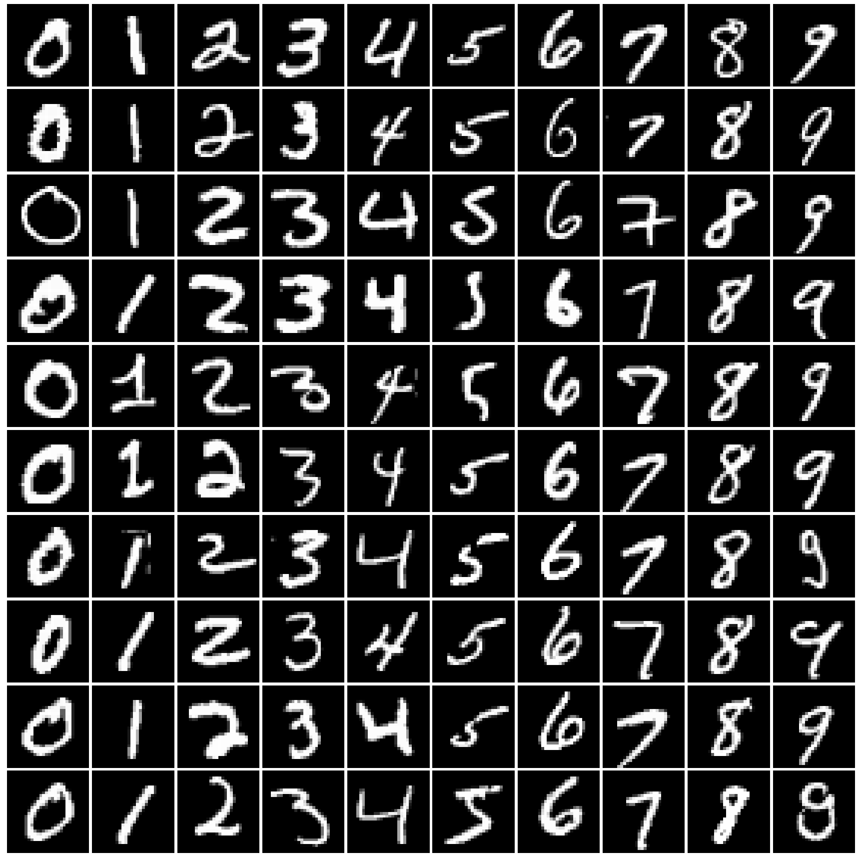




… in particular, M-dimensional vectors in $\mathcal{X} \subset \mathbb{R}^M$.

Data is far from random

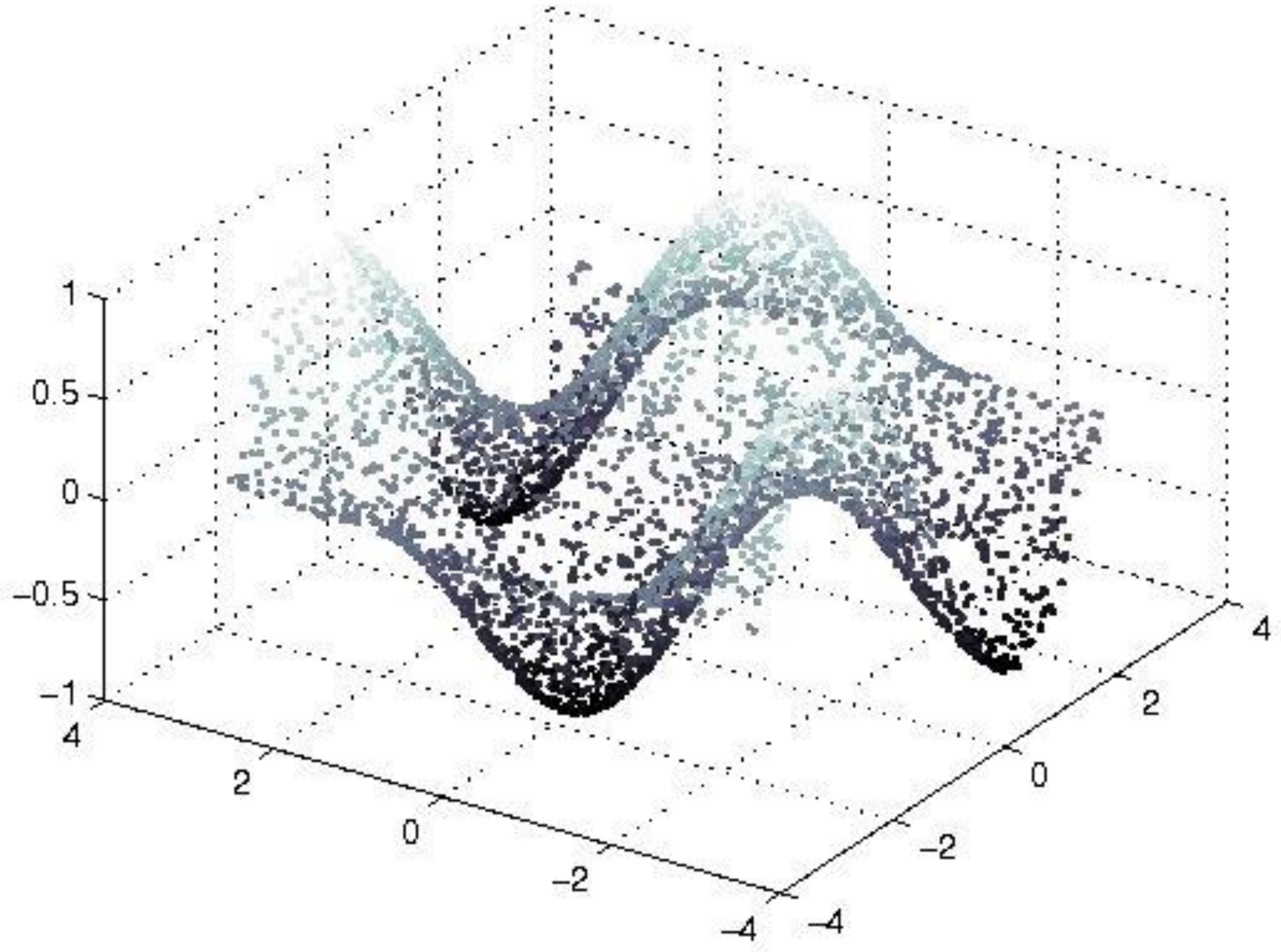




Do we need *M* pixels to represent a face?

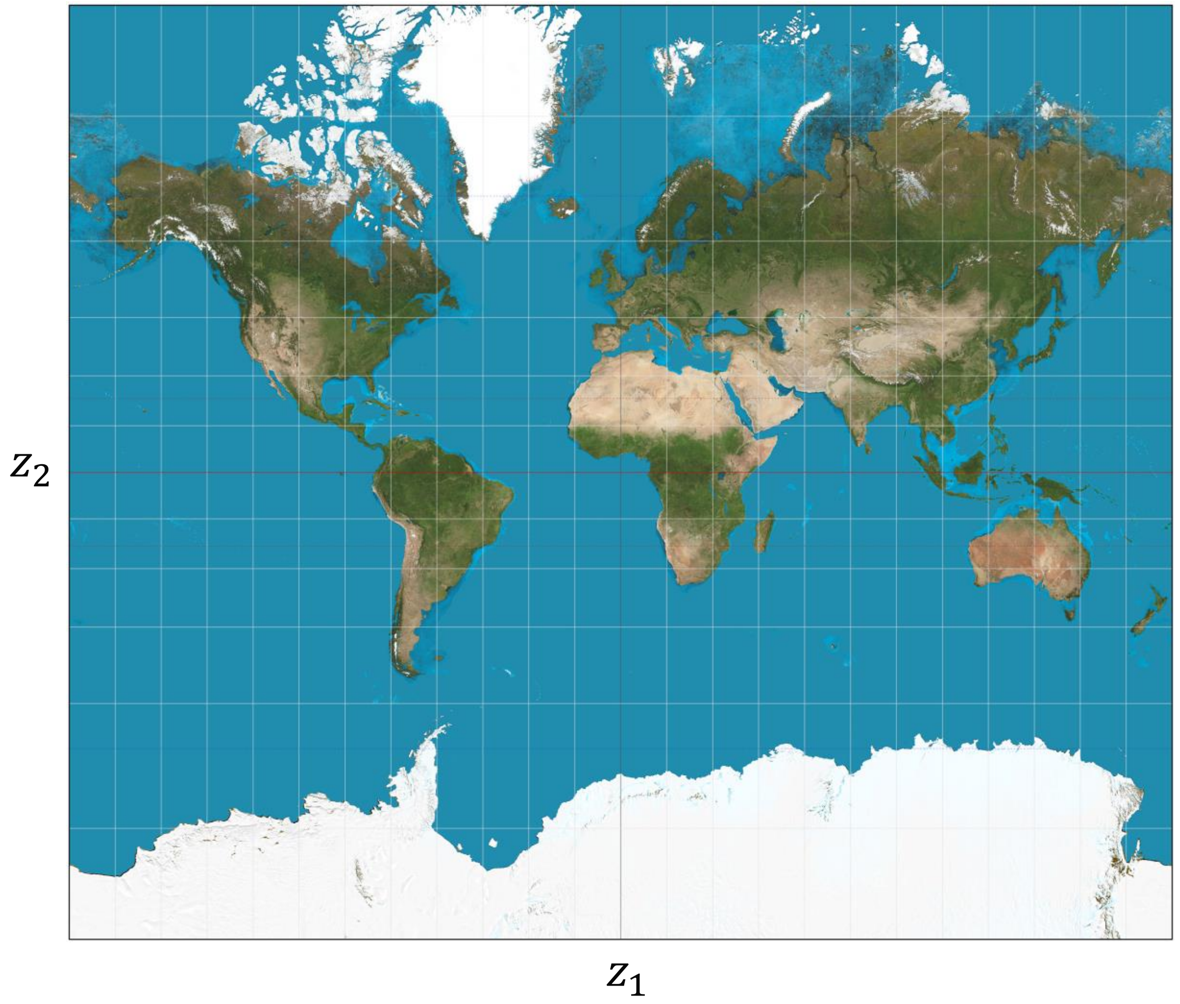
M=1000000 pixels!

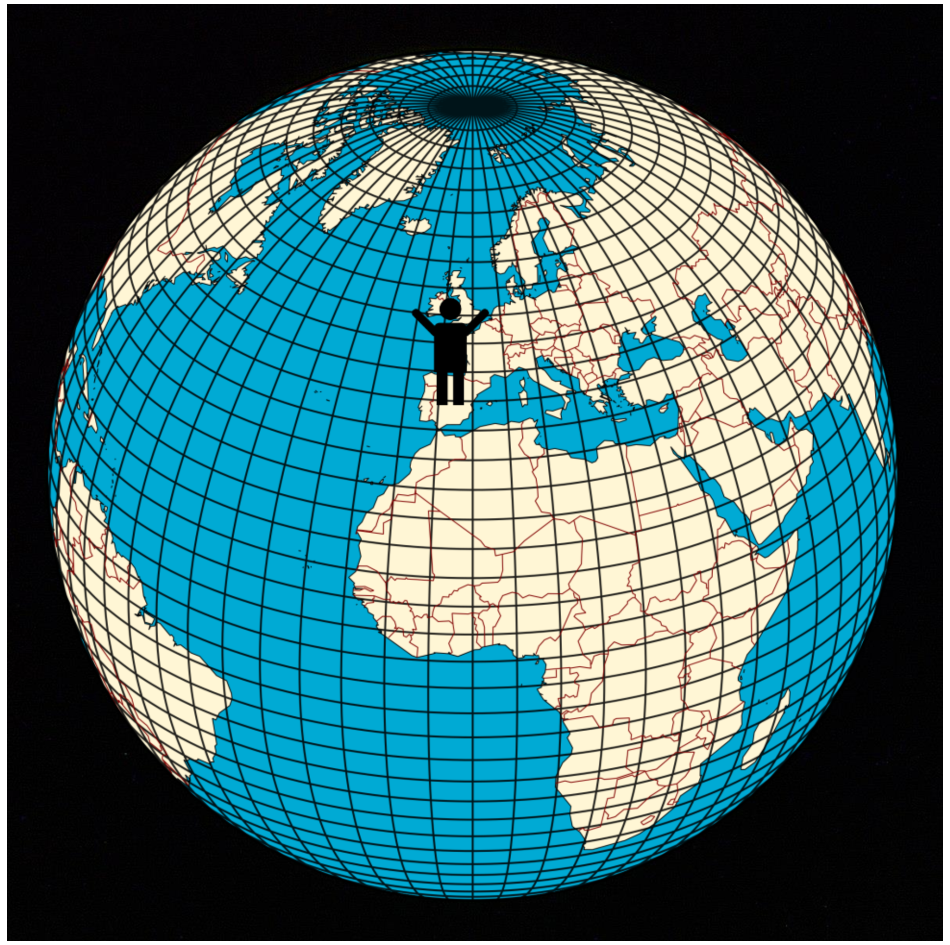
Data is not really *M*-dimensional



It rather lays on a lower dimensional *manifold*!

Manifold?

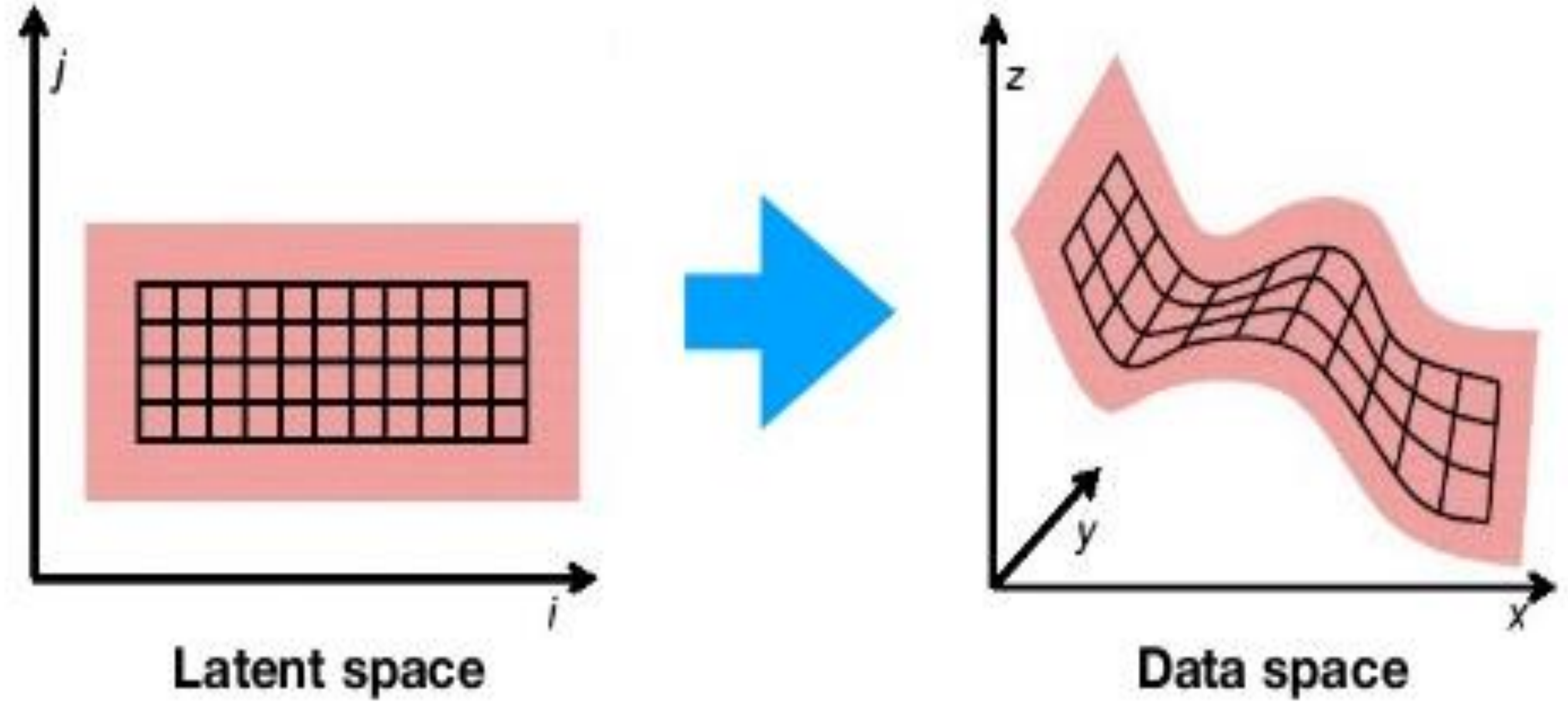




$z = (z_1, z_2)$

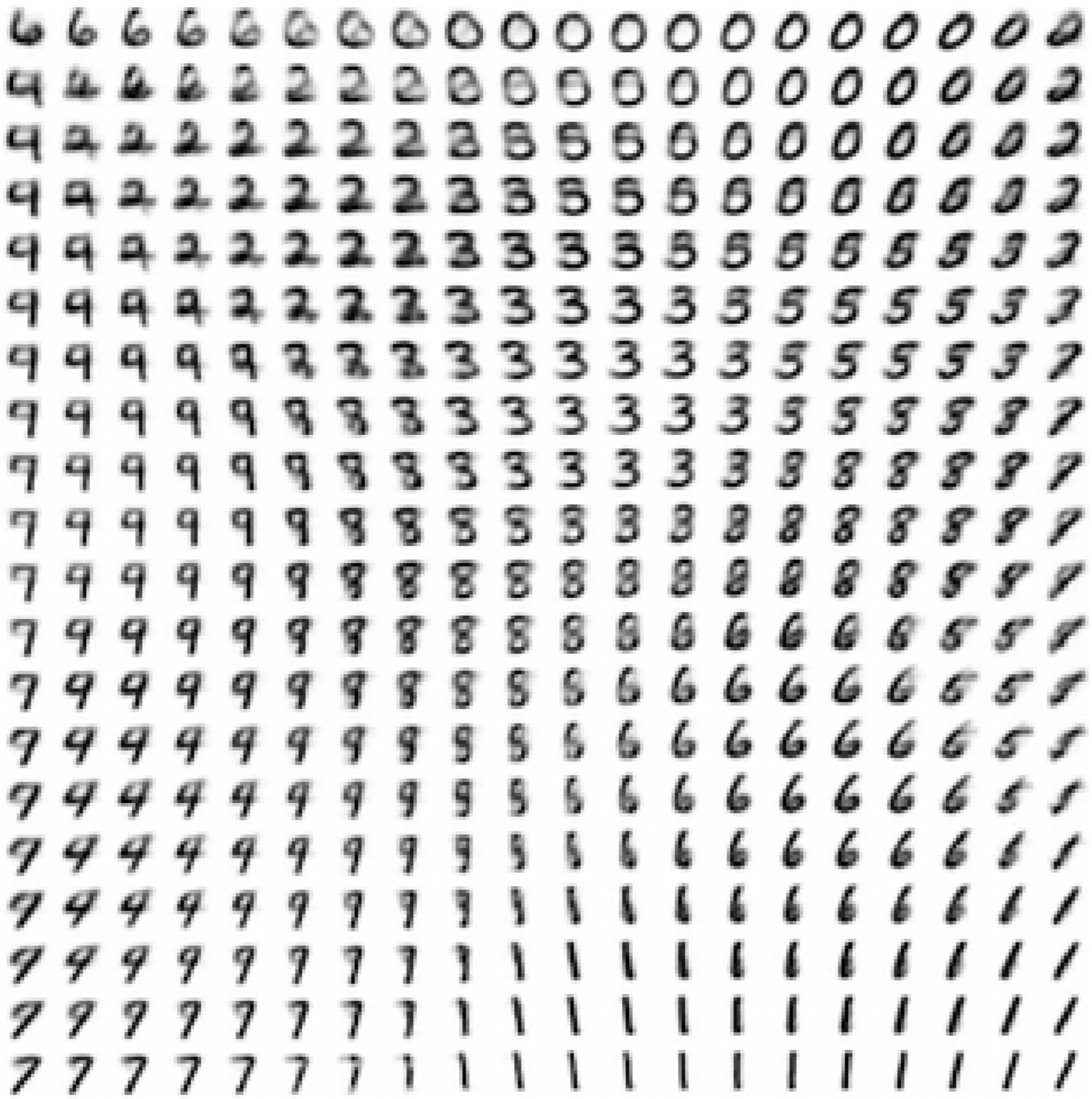
$x = (x_1, x_2, x_3)$

Latent dimensions of data



*Spoiler: Generative models learn both: the intrinsic geometry and the probability distribution!

Images credit Ward A.D. et al 2007



z_2 z_2

z_1 z_1

Auto-Encoding Variational Bayes. Kingma et al. 2013

A walk through the latent space

A Style-Based Generator Architecture for Generative Adversarial Networks. Karras et al. 2018 (NVIDIA)

And how do they work?

Random variables and generative modelling

For us, each datapoint x_i is just a *realization* of an underlying random variable.

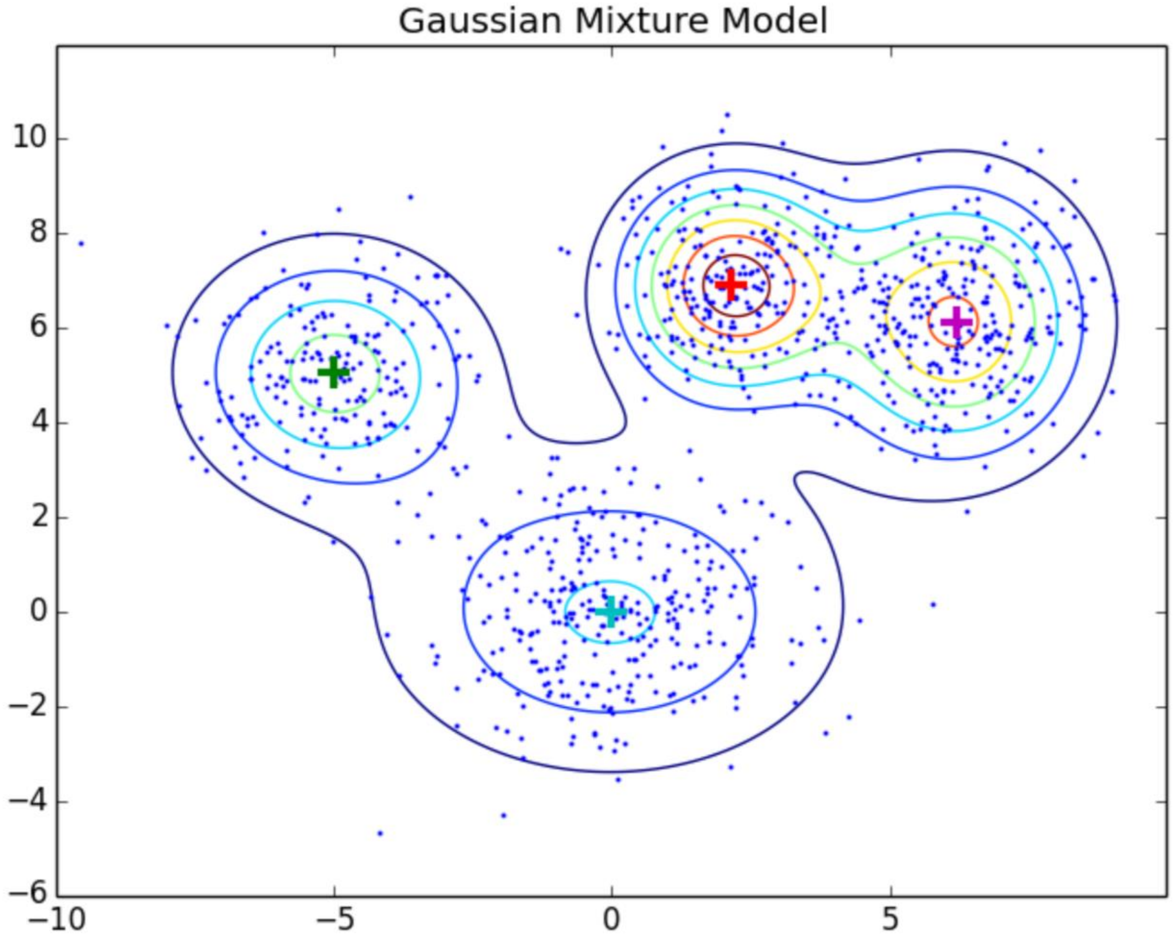
$$\mathbf{x} \sim p(x)$$

- *Unsupervised learning* is the field which attempts to infer properties of **x** from samples.
- *Generative modelling* is a subset of unsupervised learning which attempts to <u>approximate</u> **x** as some parametrized combination of "simple" random variables which you can sample.

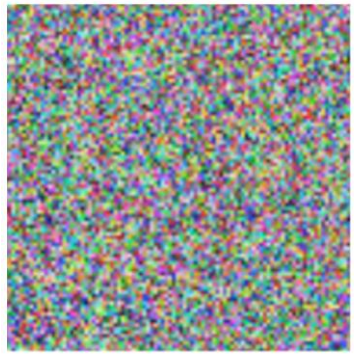
$$\mathbf{x} \approx f_\theta(\mathbf{z_1}, \mathbf{z_2}, \dots, \mathbf{z_K}) \triangleq \hat{\mathbf{x}}$$

Example: Gaussian Mixtures

Here every $\mathbf{z}_i$ is normal (gaussian) $\mathcal{N}(\mu_i, \Sigma_i)$ and the combination $f_\theta(\cdot)$ is a *mixture*.



Latent variable models



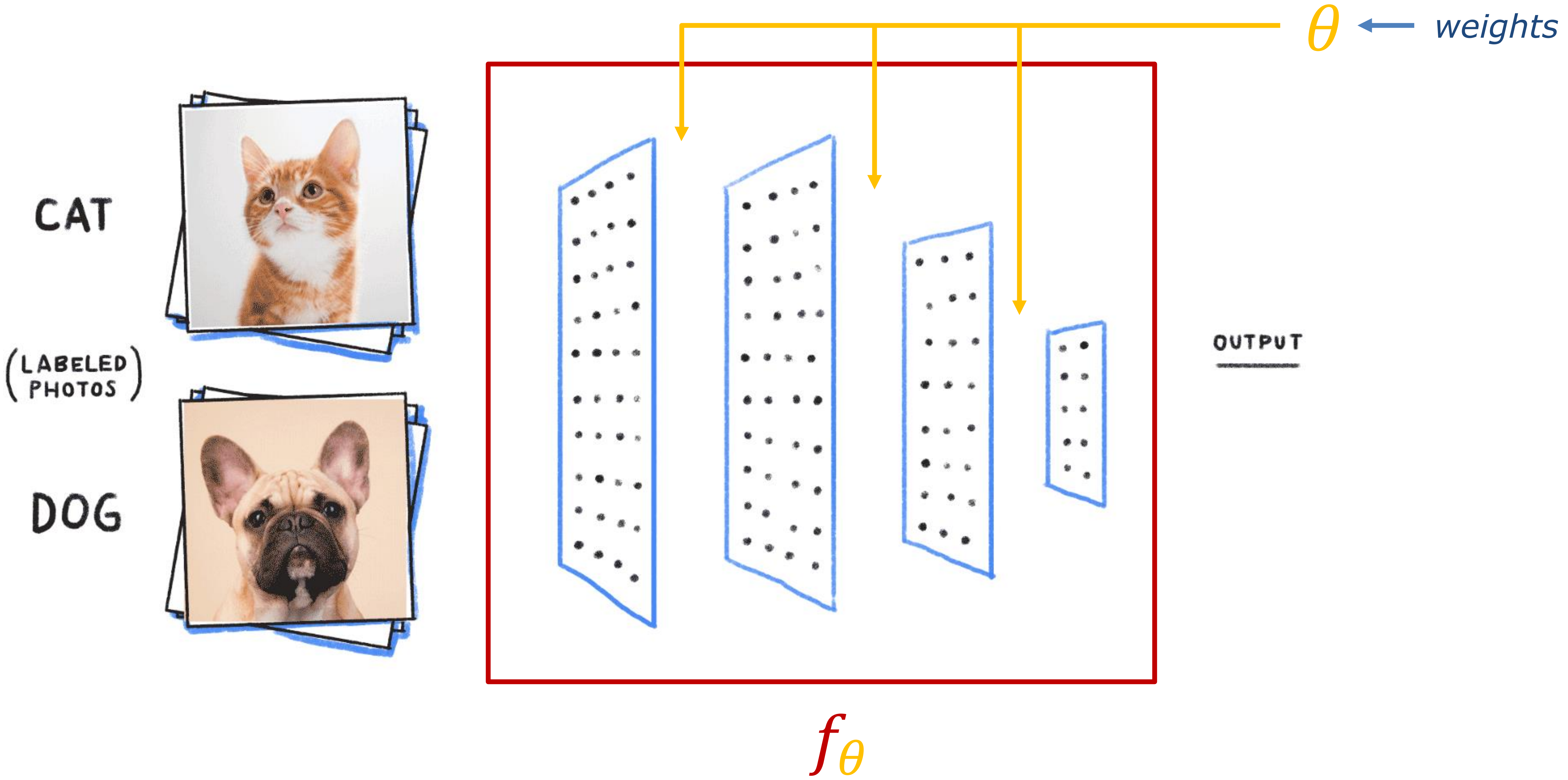
$\mathbf{z} \sim p(z)$

Prior distribution

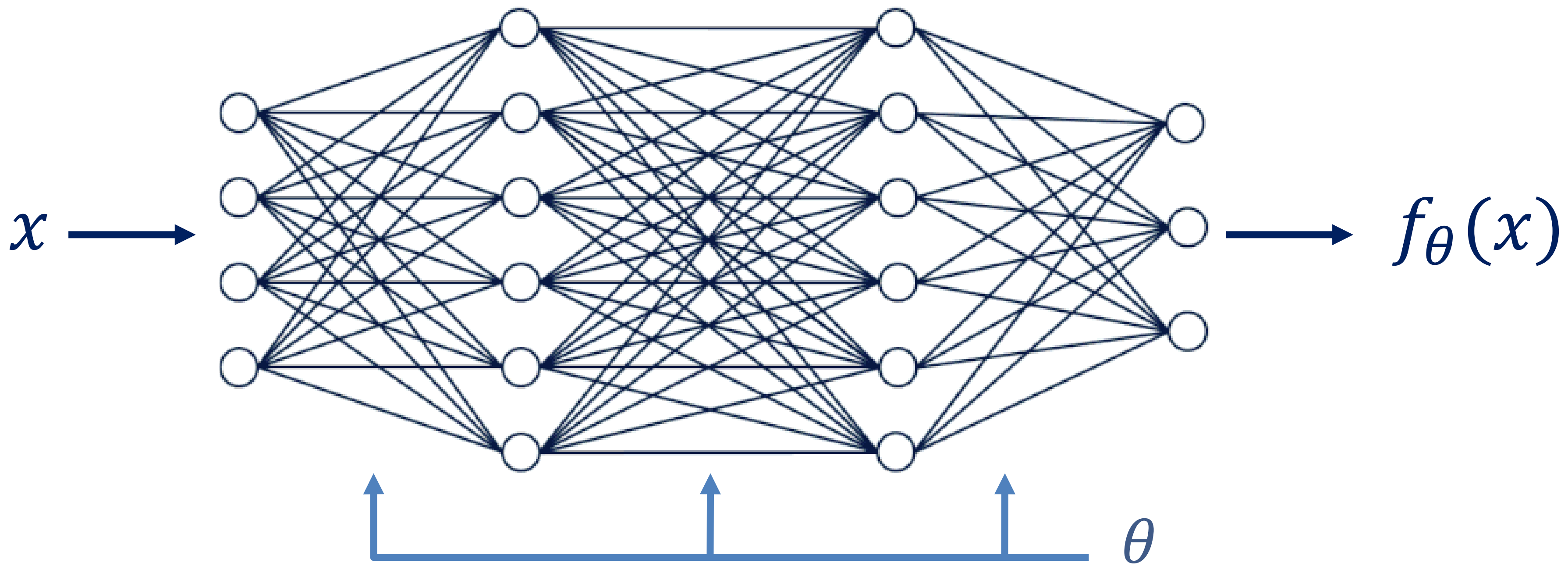
Transformation

$\hat{\mathbf{x}}$

Architectures: neural networks



Architectures: neural networks



Neural networks can approximate any function $f(x)$ *to any precision!* *

**G. Cybenko 1989, K. Hornik 1991, Z. Lu et al 2017, B. Hanin 2017*

Image credit Sydney Firmin

But to find the right θ (training)?

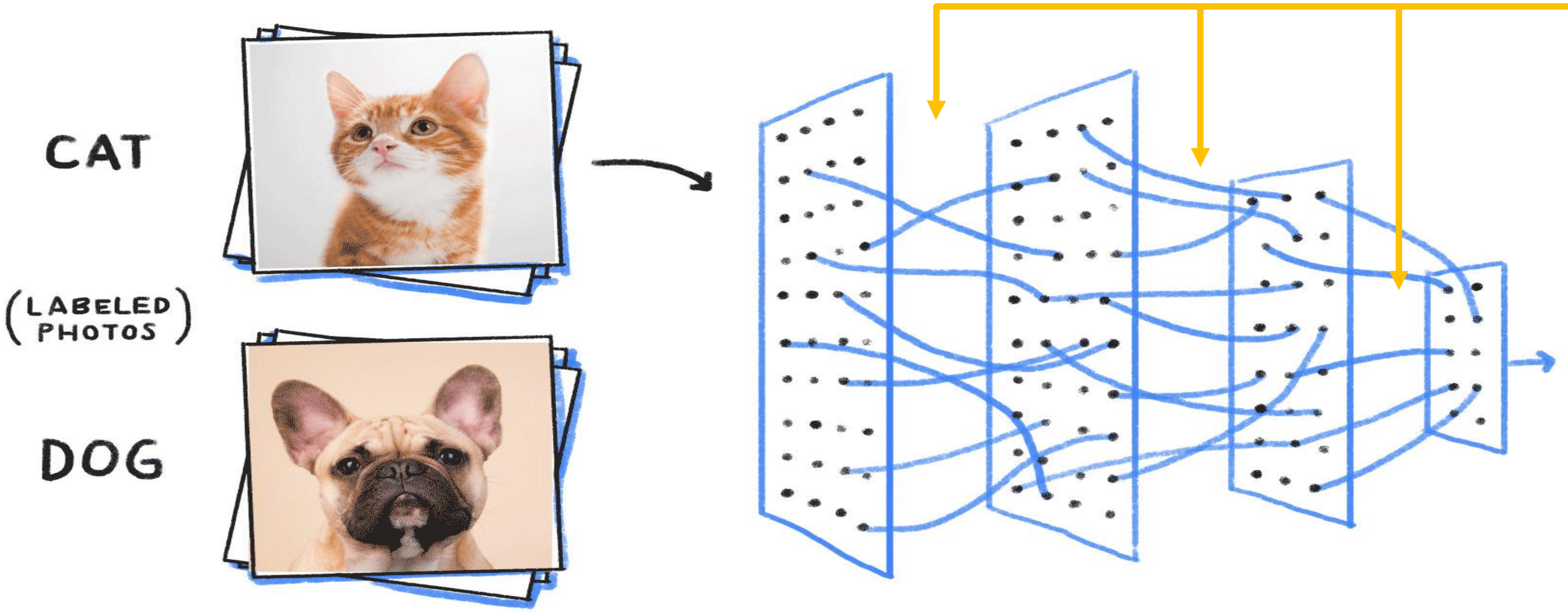
You optimize some loss (error) function!

$$\min_{\theta} L(\theta; \text{data})$$

E.g. Classifier

$y_{cat} \in \{0,1\}$

x



$y_{dog} \in \{0,1\}$

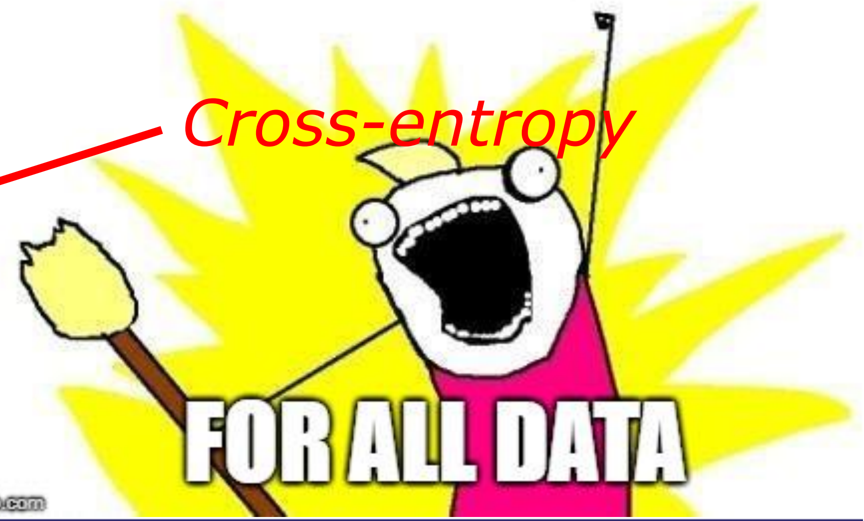
θ

$f_\theta(x) \xrightarrow{\text{I want it to be...}} p(cat)$

MINIMIZE

Cross-entropy

$$\boxed{L(\theta; x, y) = -y_{cat} \log f_\theta(x) - y_{dog} \log\left(1 - f_\theta(x)\right)}$$



E.g. Classifier

$y_{cat} \in \{0,1\}$

CAT

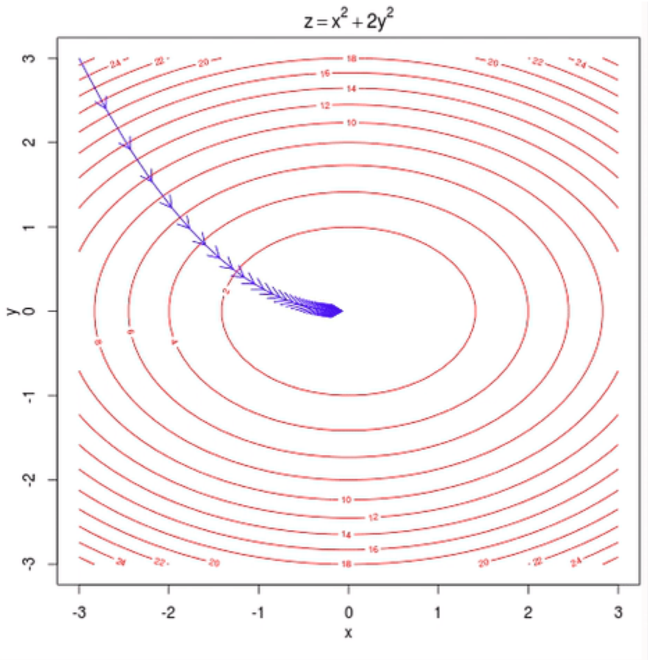
x (LABELED PHOTOS)

DOG

be... $\rightarrow$ $p(cat)$

Cross-entropy

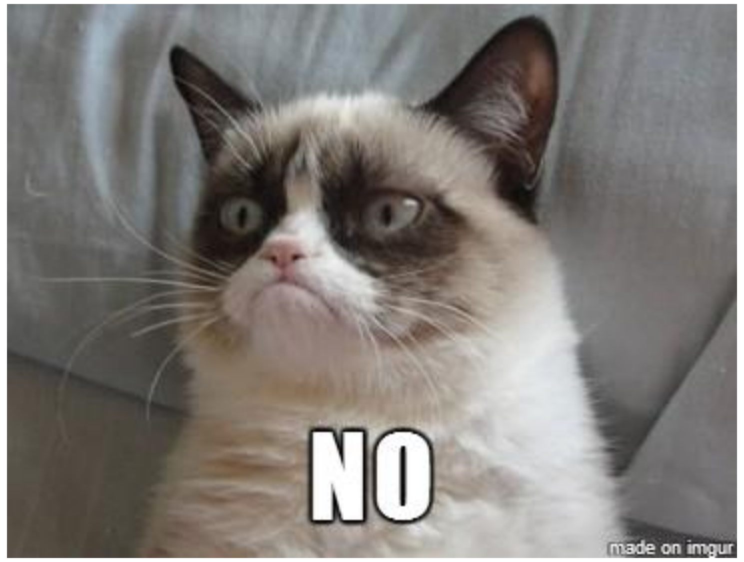
But to find the right θ (training)?

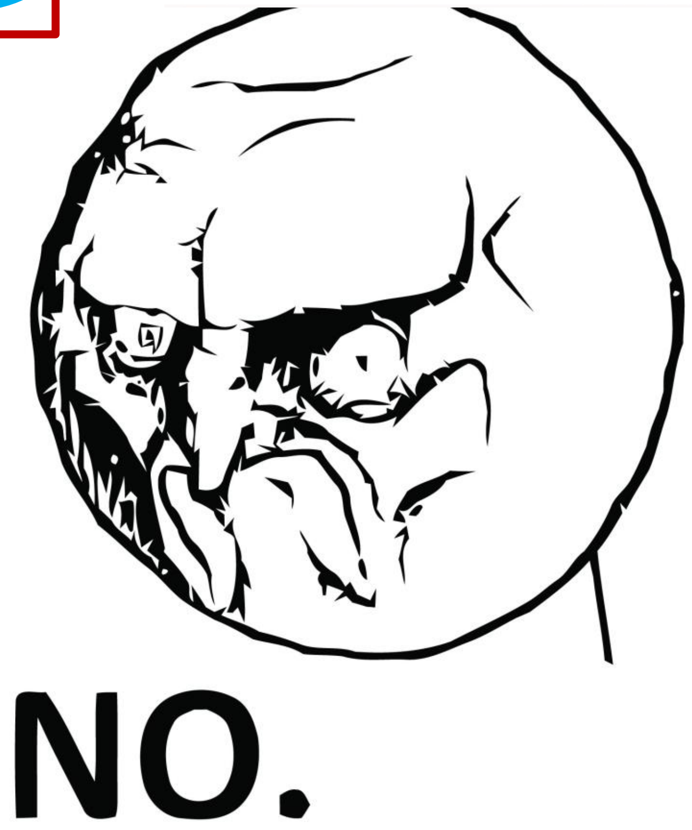


You optimize some loss (error) function!

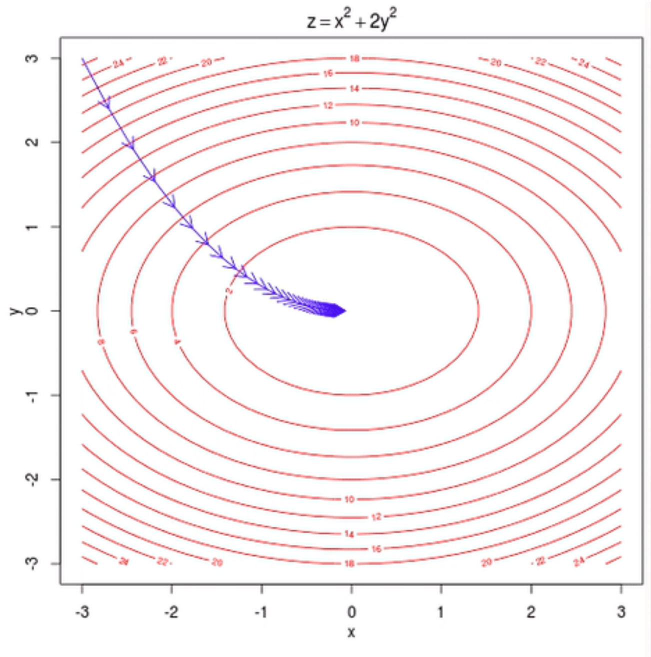
Stochastic Gradient Descent

$$\min_{\theta} L(\theta;\mathrm{data}) \longrightarrow \boxed{\theta_{t+1} = \theta_t - \alpha\nabla_\theta L}$$





But to find the right θ (training)?



You optimize some loss (error) function!

Stochastic Gradient Descent

$$\min_{\theta} L(\theta; \text{data}) \longrightarrow \boxed{\theta_{t+1} = \theta_t - \alpha \nabla_\theta L}$$

```python
model = MyNetwork()
theta = model.parameters()
optimizer = torch.optim.Adam(theta, lr=0.001)

for x, y in dataloader:
    y_pred = model(x)
    loss = myloss(y, y_pred)
    loss.backwards()
    optimizer.step()
```

PyTorch

```python
model = MyNetwork()
theta = model.trainable_variables
optimizer = tf.train.AdamOptimizer(lr = 0.001)

for x, y in dataset:
    with tf.GradientTape() as g
        y_pred = model(x)
        loss = myloss(y, y_pred)
    grads = g.gradient(loss, theta)
    optimizer.apply_gradients(zip(grads, model.trainable_variables))
```

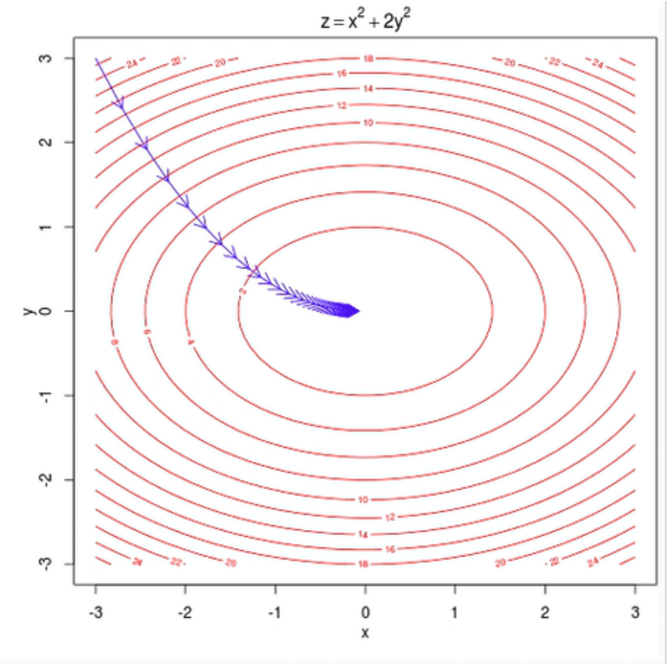
TensorFlow

But to find the right θ (training)?

You optimize some loss (error) function!

Stochastic Gradient Descent

$$\min_{\theta} L(\theta; \text{data}) \longrightarrow \boxed{\theta_{t+1} = \theta_t - \alpha \nabla_\theta L}$$




```
model = MyNetwork()
theta = model.parameters()
optimizer = torch.optim.Adam(theta, lr=0.001)

for x, y in dataloader:
    y_pred = model(x)
    loss = myloss(y, y_pred)
    loss.backwards()
    optimizer.step()
```

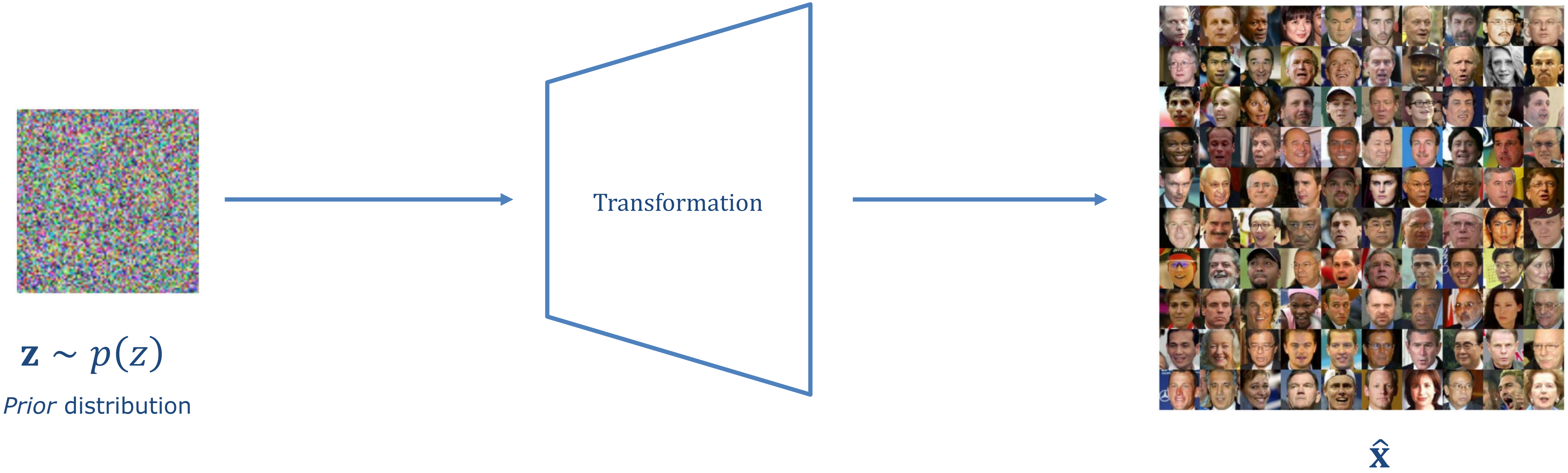
PyTorch

```
model = MyNetwork()
theta = model.trainable_variables
optimizer = tf.train.AdamOptimizer(lr = 0.001)
        y_pred = model(x)
        loss = myloss(y, y_pred)
    grads = g.gradient(loss, theta)
    optimizer.apply_gradients(zip(grads, model.trainable_variables))
```

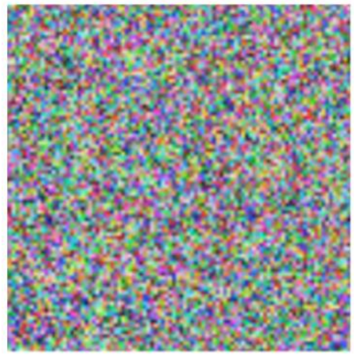
TensorFlow

Easy to gradient descent any function with current frameworks!! ☺

Latent variable models

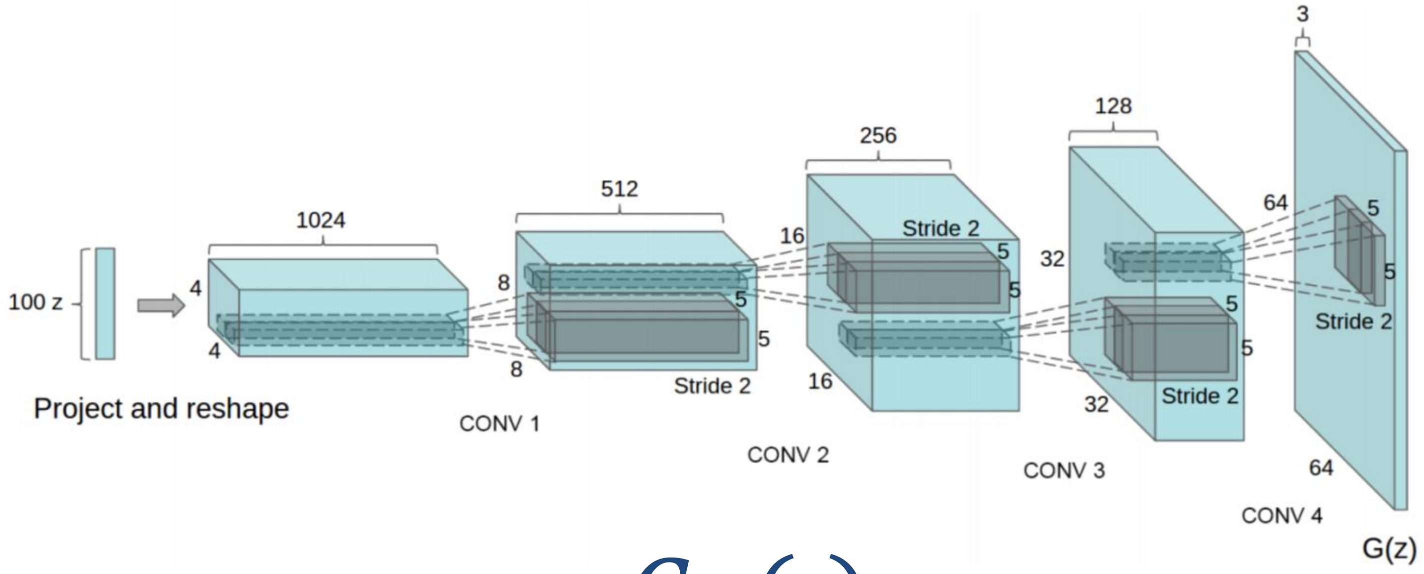


Deep latent variable models



$\mathbf{z} \sim p(z)$

Prior distribution



$G_\theta(\cdot)$

$\hat{\mathbf{x}} = G_\theta(\mathbf{z})$

Training

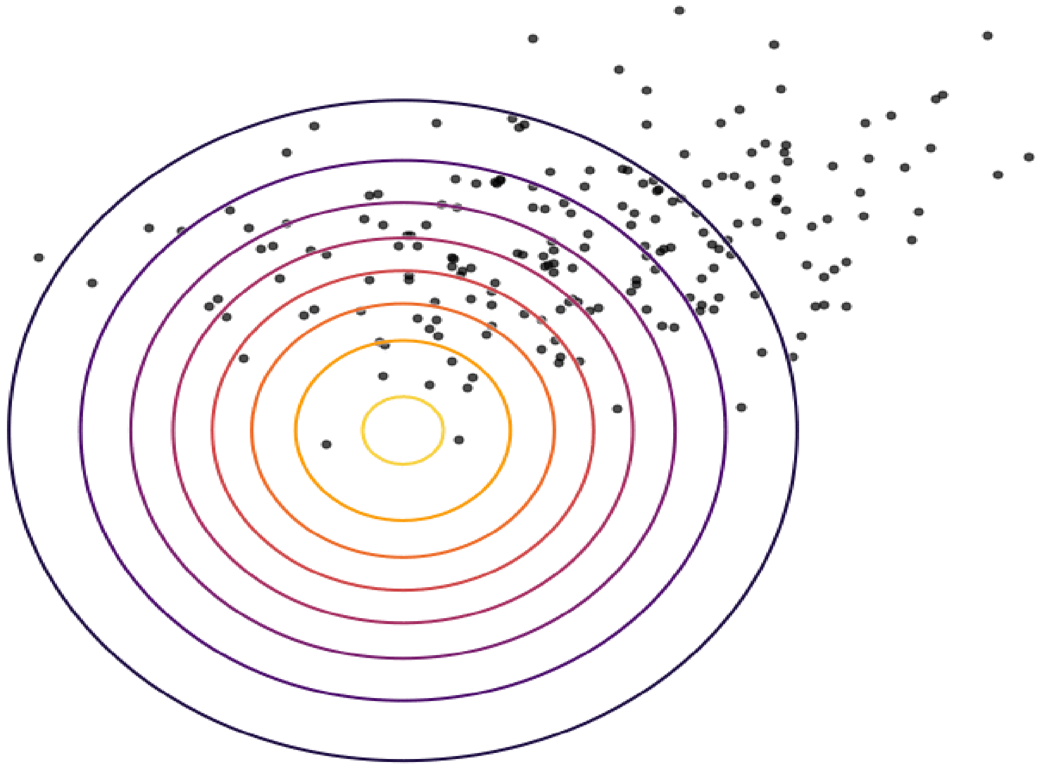
We want to approximate $\mathbf{x}$ as $\hat{\mathbf{x}} = G_\theta(\mathbf{z})$. How do we find the optimal θ?

Maximize the likelihood of the data!

Probability that the model would generate x_i

$$\max_\theta \mathcal{L}(\theta | x_{train}) = \max_\theta \prod_{i=1}^{N} p_\theta(x_i)$$

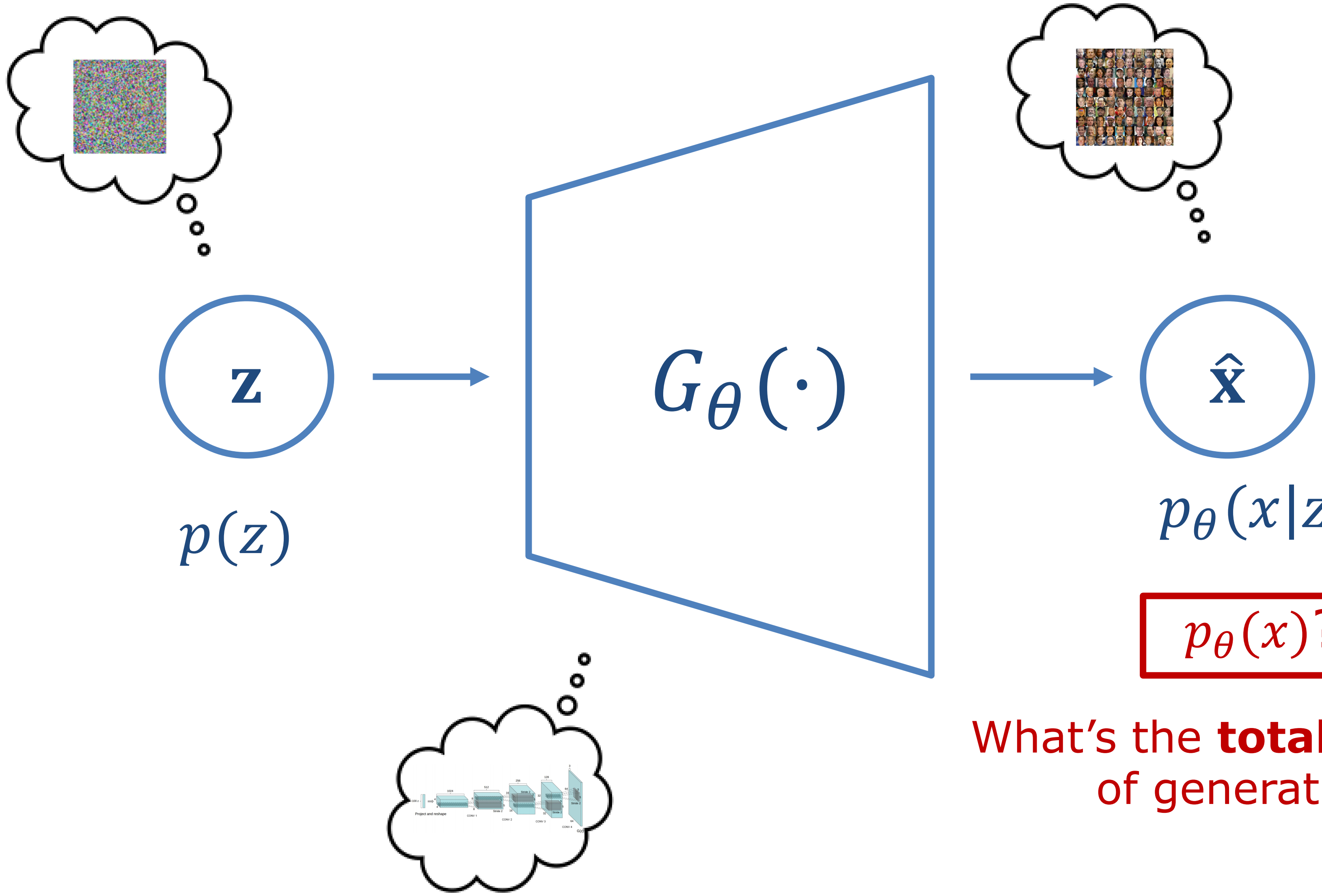
$$\max_\theta \log \mathcal{L}(\theta | x_{train}) = \max_\theta \sum_{i=1}^{N} \log p_\theta(x_i)$$



But… we need $p_\theta(x)$ explicitly!

Image credit: Colin Raffel

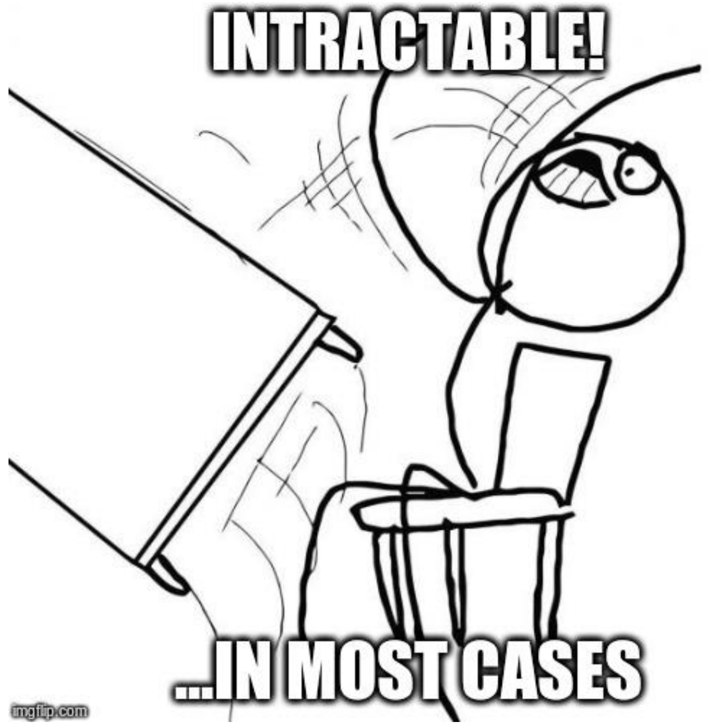
What about deep latent variable models?



$$p_\theta(x) = \int p_\theta(x|z)\,p(z)dz$$

$\boxed{p_\theta(x)??}$

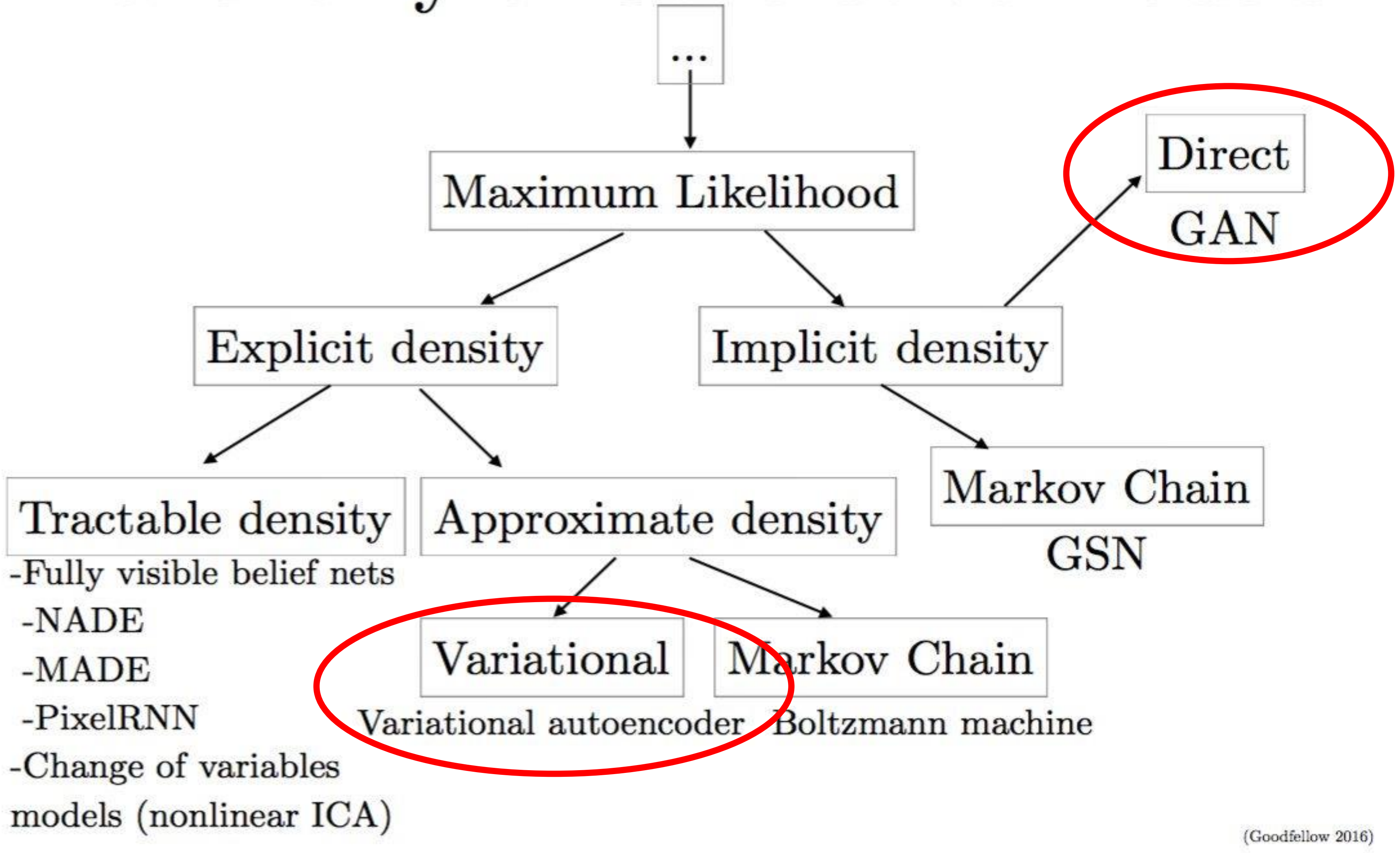
What's the **total** probability of generating x?



Different models – different methods

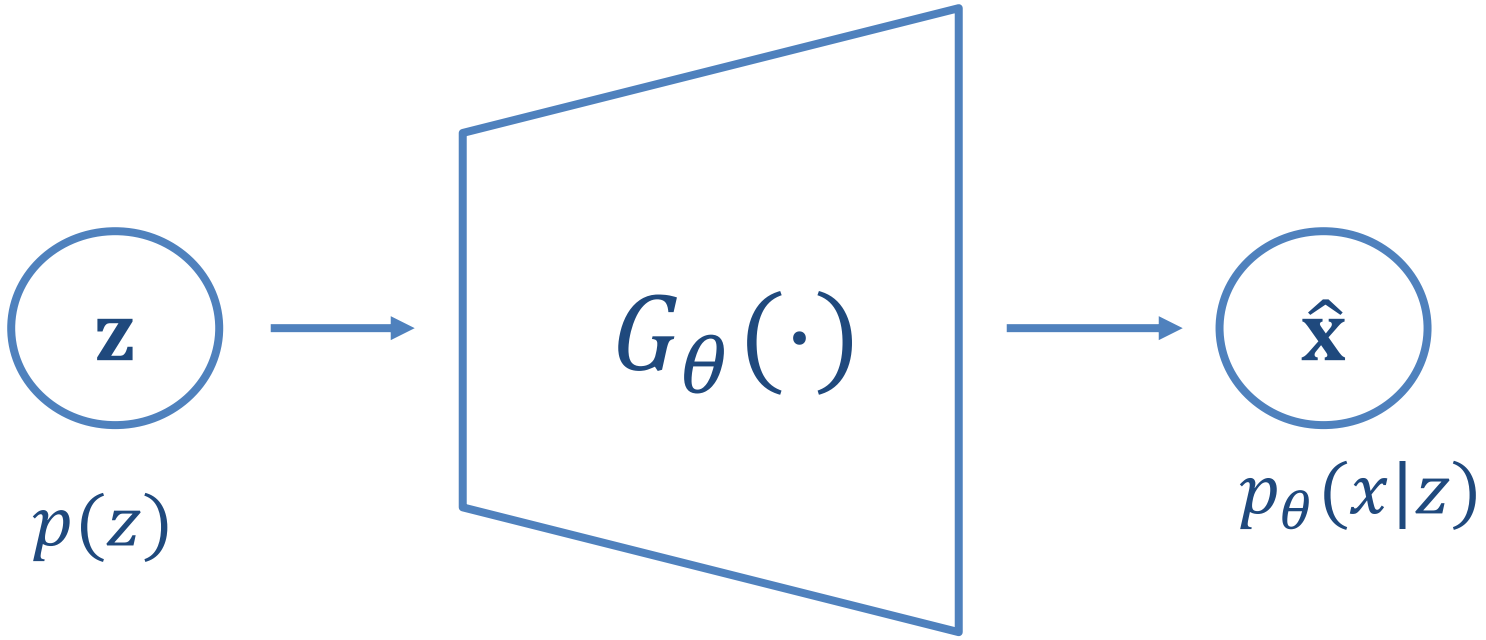
1. We have $p_\theta(\hat{x})$ explicitly: **maximize the likelihood**.
2. $p_\theta(\hat{x})$ is intractable: we can approximate it instead
   - Markov Chain Monte Carlo (MCMC) methods
   - Variational methods (e.g. Variational Autoencoders)
3. We don't need $p_\theta(\hat{x})$; it's implicit.
   - Adversarial methods (e.g. GANs)

Taxonomy of Generative Models

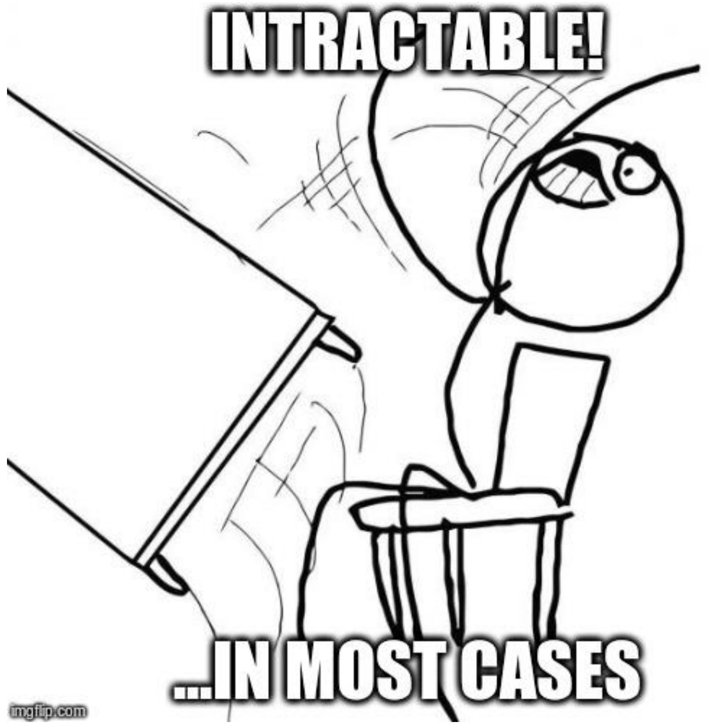


Goodfellow. 2016

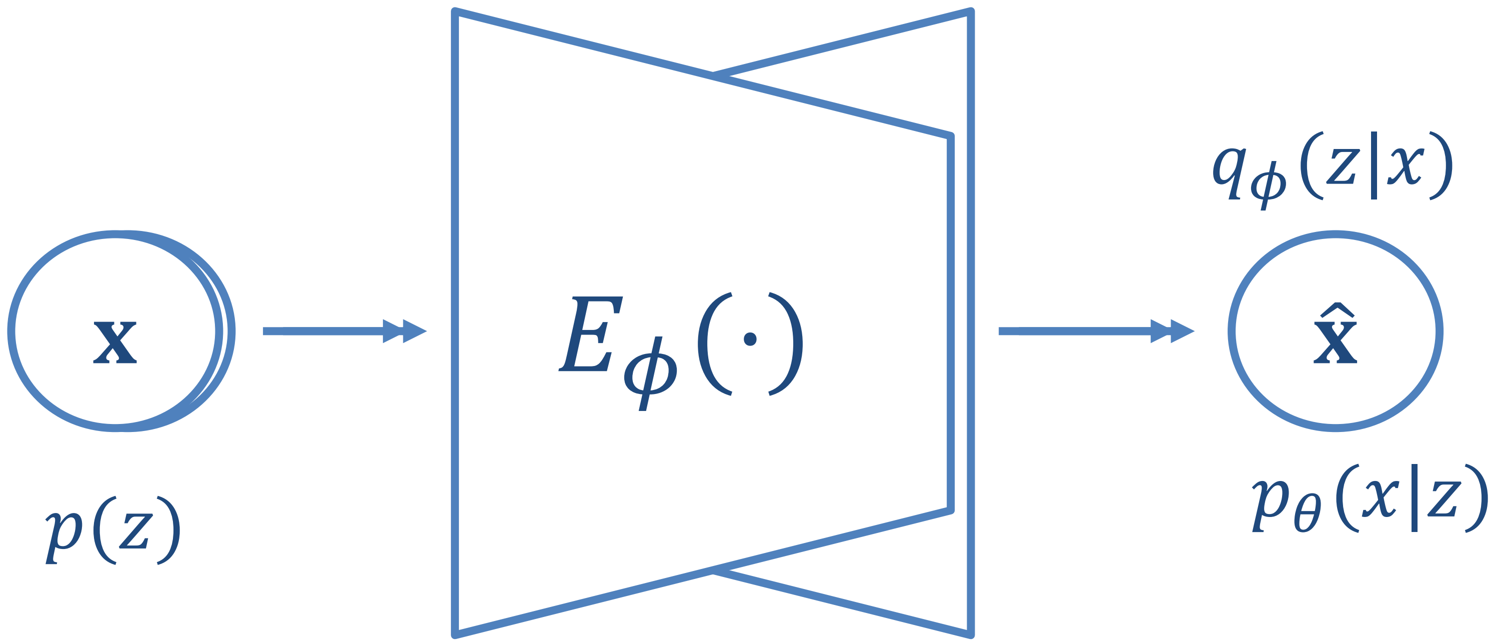
Variational autoencoder (VAE)



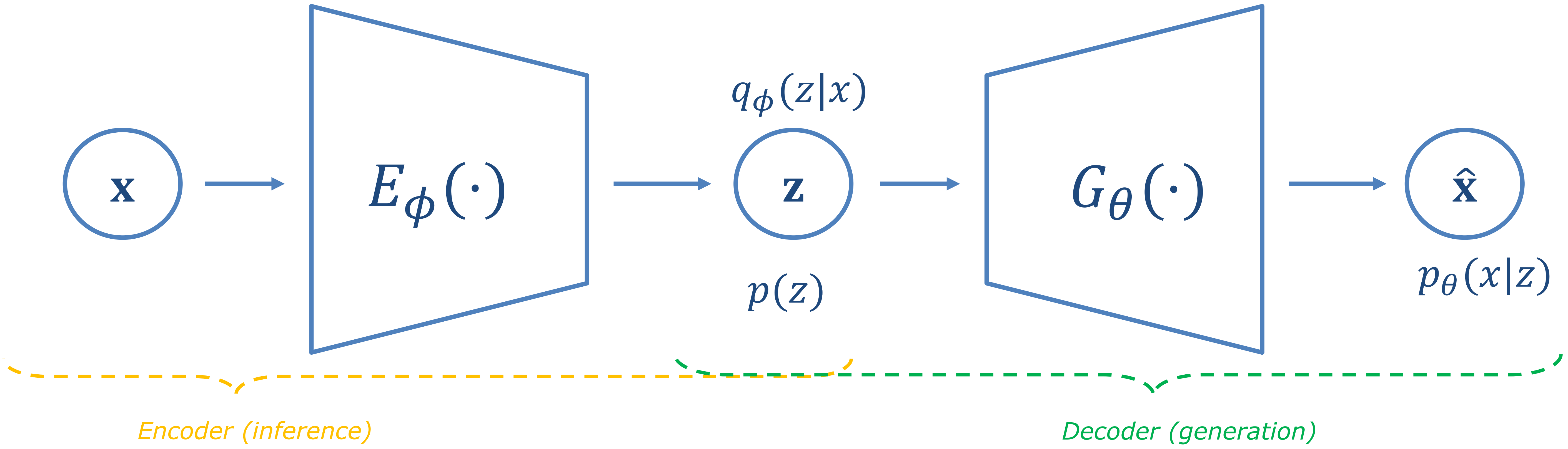
$$p_\theta(x) = \int p_\theta(x|z)\, p(z) dz$$



Variational autoencoder (VAE)

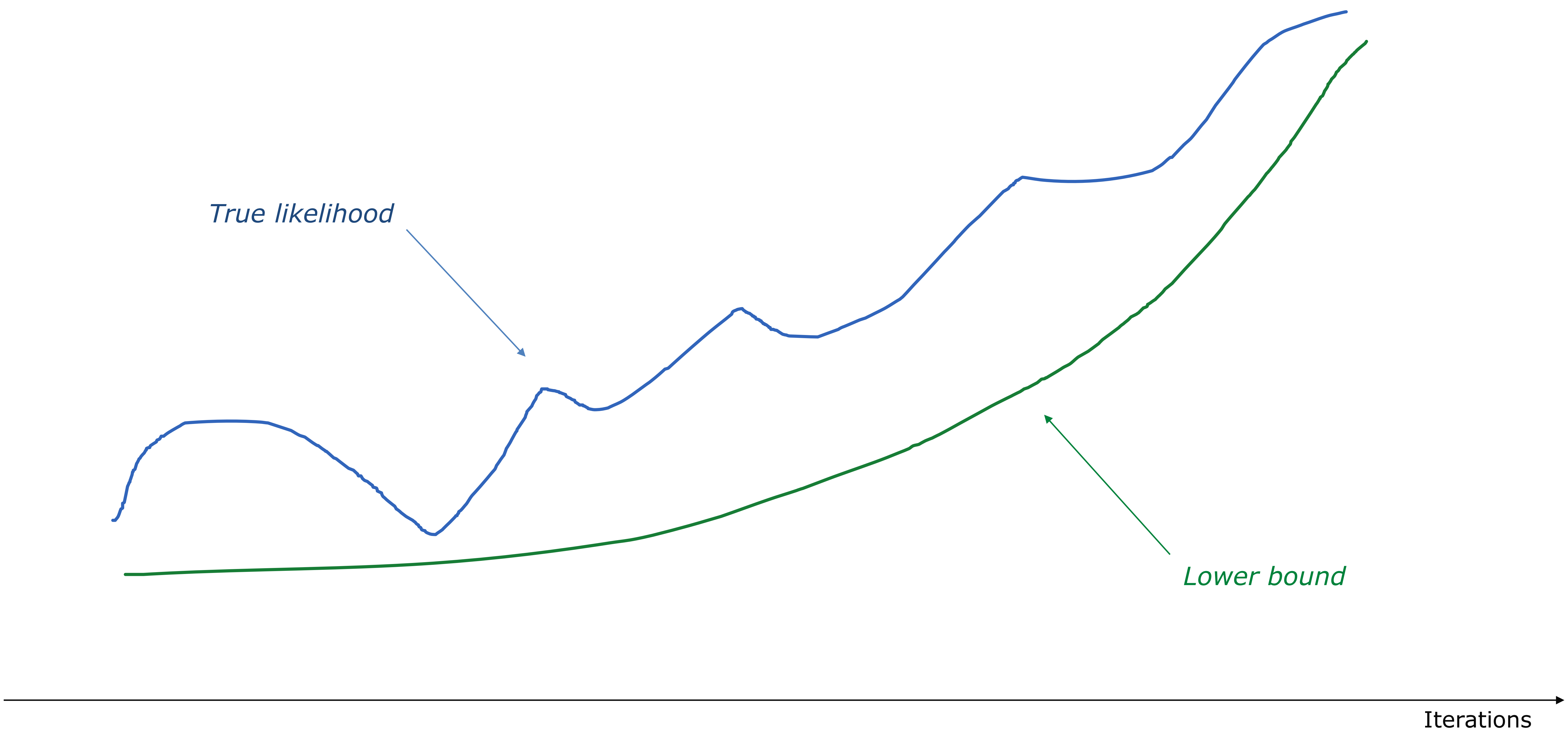


Variational autoencoder (VAE)



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

Maximize this instead!


True likelihood
Lower bound
Iterations

Variational autoencoders

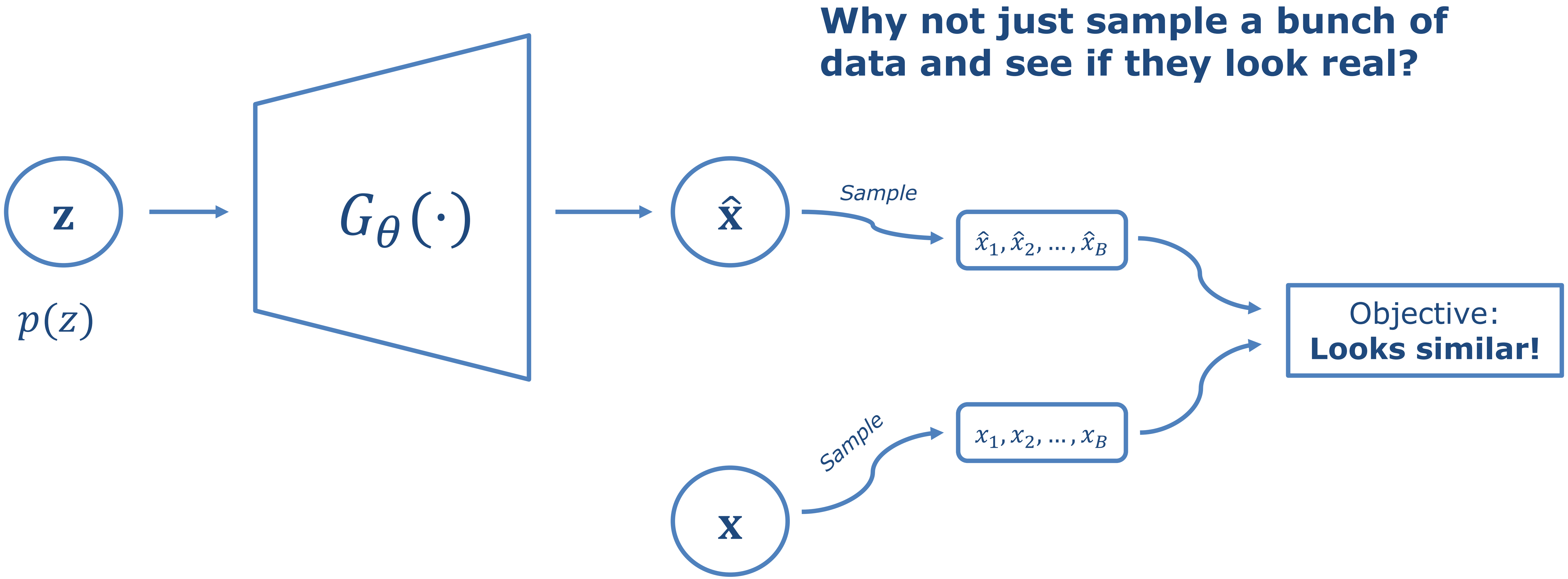
Pros:

- Efficient inference for free!
  - Great tool for modelling the hidden structure of data.
- Stable to train.
- Good theoretical ground.

Cons:

- Not very good samples.

Generative adversarial networks (GANs)



But… how do we measure similarity between groups of samples?

How to measure similarity of samples

One solution: **train a <u>classifier</u>** $D_\phi(x)$ **to discriminate!**

- If the classifier can not tell if a sample is real or fake, both distributions are close.
- Trained with the standard *cross-entropy loss*:

$$\max_{\phi} L_d(\phi) = \max_{\phi}\left(\mathbb{E}_{x_r \sim p_{real}} \log\left(D_\phi(x_r)\right) + \mathbb{E}_{x_f \sim p_{fake}} \log\left(1 - D_\phi\left(x_f\right)\right)\right)$$

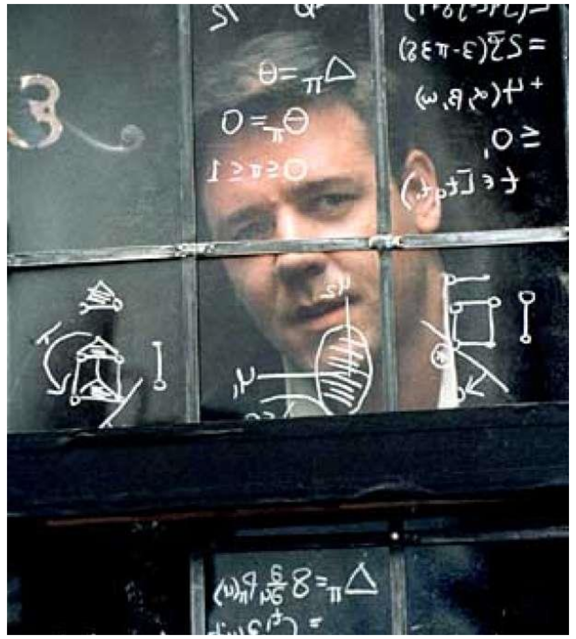
It can be shown that the *optimal* classifier performance $L_d(\phi^*)$ is related to the *closeness* between both distributions (JS divergence).

The GAN game

We want to minimize “closeness” between the generated and real samples, as measured by the discriminator loss:

$$\min_{\theta} \text{"closeness"}$$

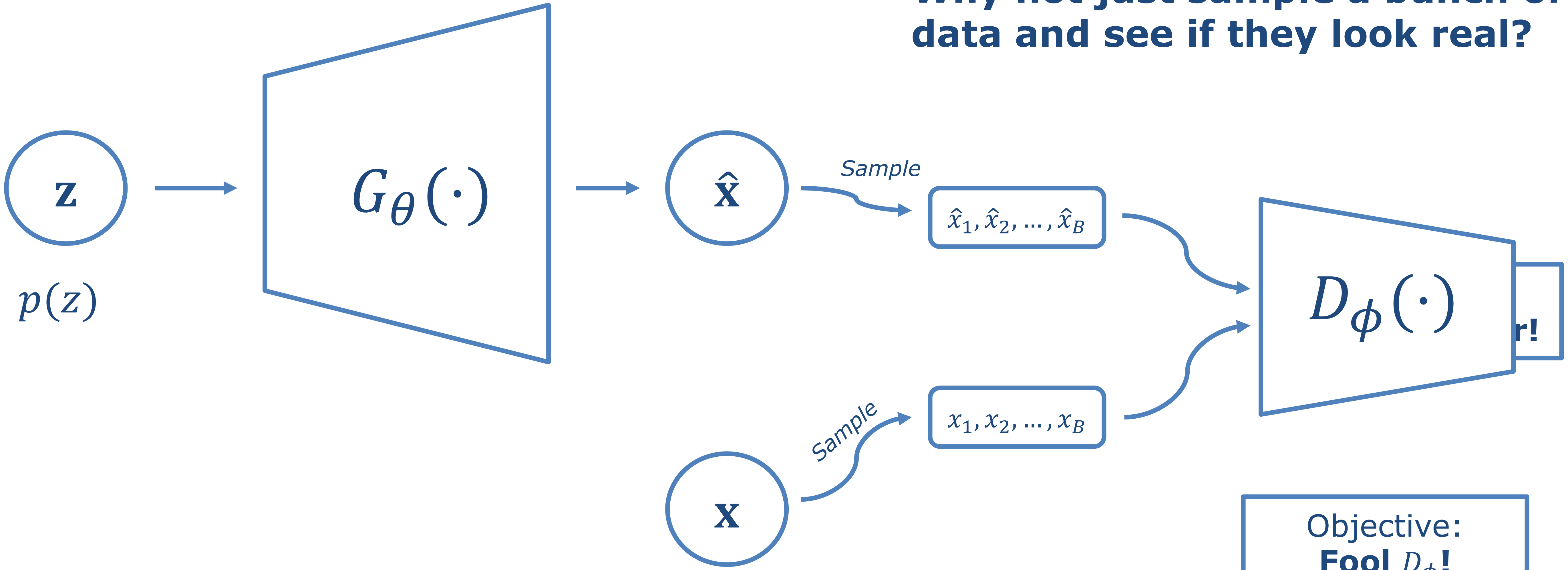
$$= \min_{\theta}\left(\max_{\phi}\left(\mathbb{E}_{x_r\sim p_{real}} \log\left(D_{\phi}(x_r)\right) + \mathbb{E}_{x_f\sim p_{fake}} \log\left(1 - D_{\phi}(x_f)\right)\right)\right)$$



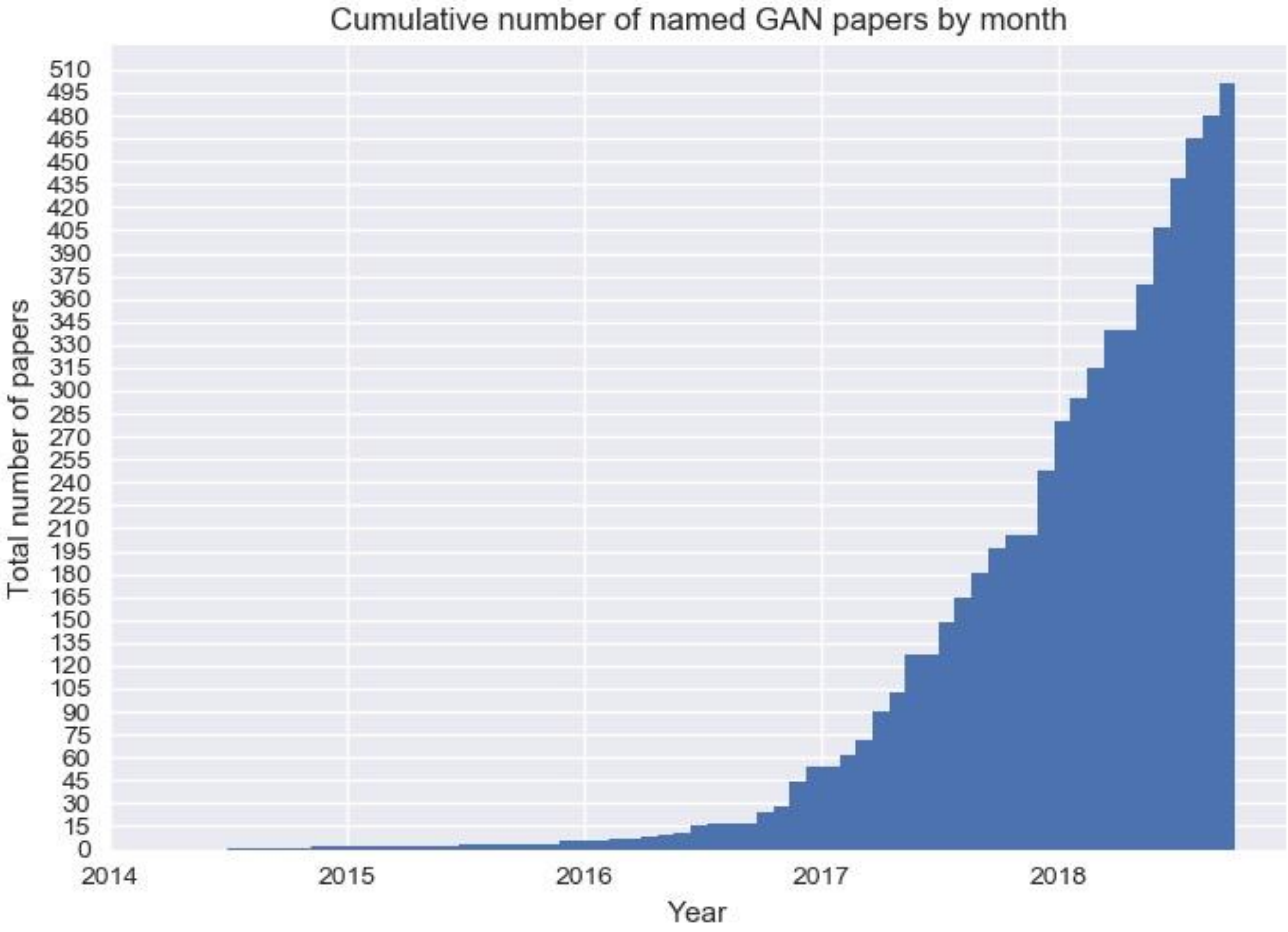
<u>It’s formally a two-player minimax game!!</u>

Generative adversarial networks

Why not just sample a bunch of data and see if they look real?



Objective:
Fool D_ϕ**!**

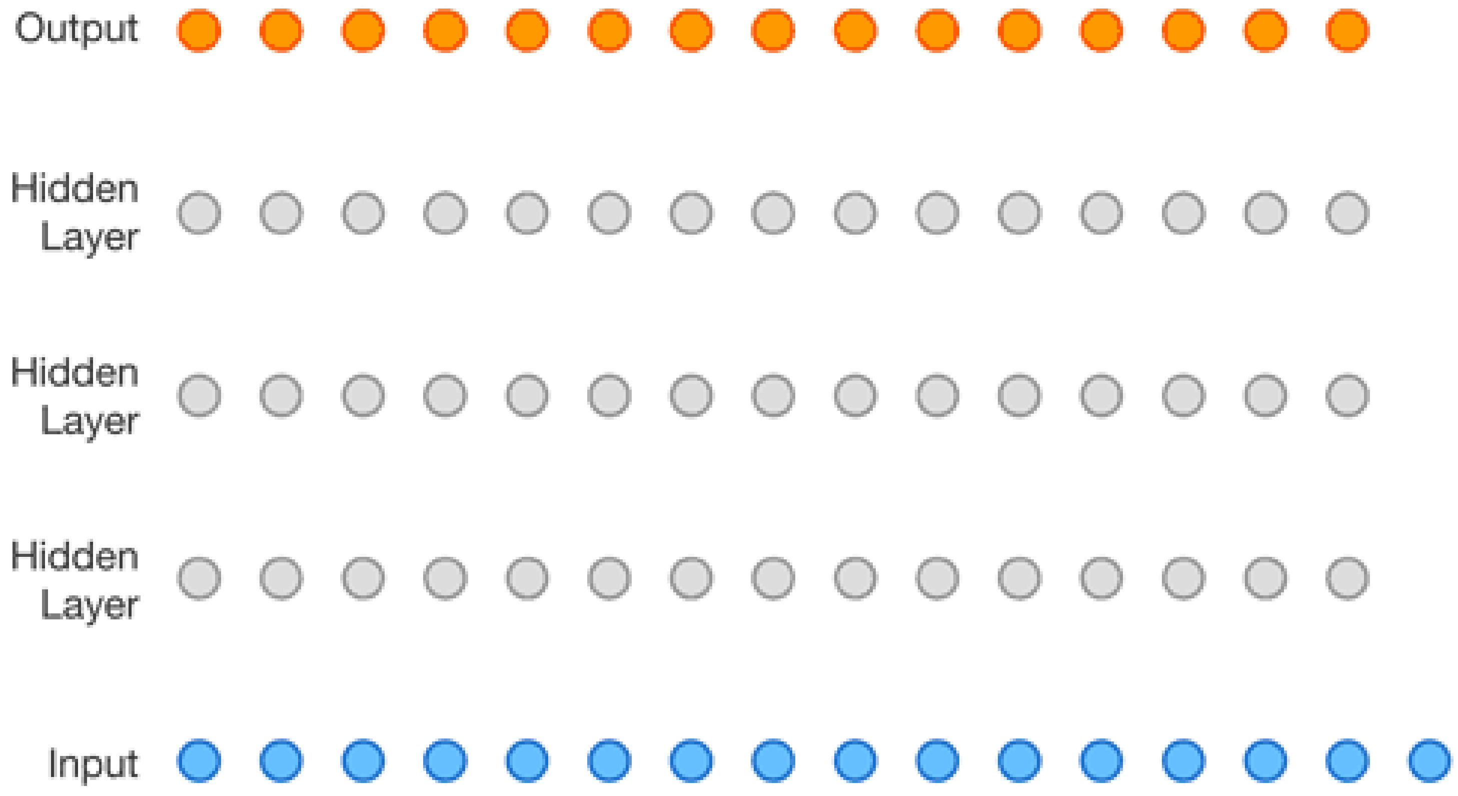

Cumulative number of named GAN papers by month
Total number of papers
510
495
480
465
450
435
420
405
390
375
360
345
330
315
300
285
270
255
240
225
210
195
180
165
150
135
120
105
90
75
60
45
30
15
0
2014
2015
2016
2017
2018
Year

GANs

- Pros:
  - Awesome samples
- Cons:
  - Unstable training
  - No explicit probability density
  - No direct inference

Large Scale GAN Training for High Fidelity Natural Image Synthesis. Brock et al. 2018

Bonus: autoregressive methods



$$p_\theta(x) = \prod_{t=1}^{T} p_\theta(x_t | x_1, \ldots, x_{t-1})$$

Generate little by little!

Wavenet: A Generative Model for Raw Audio. Van den Oord et al. 2016.

OK!

OK! I can generate stuff.

OK! I can generate stuff.

But how do I influence what I generate?

OK! I can generate stuff.

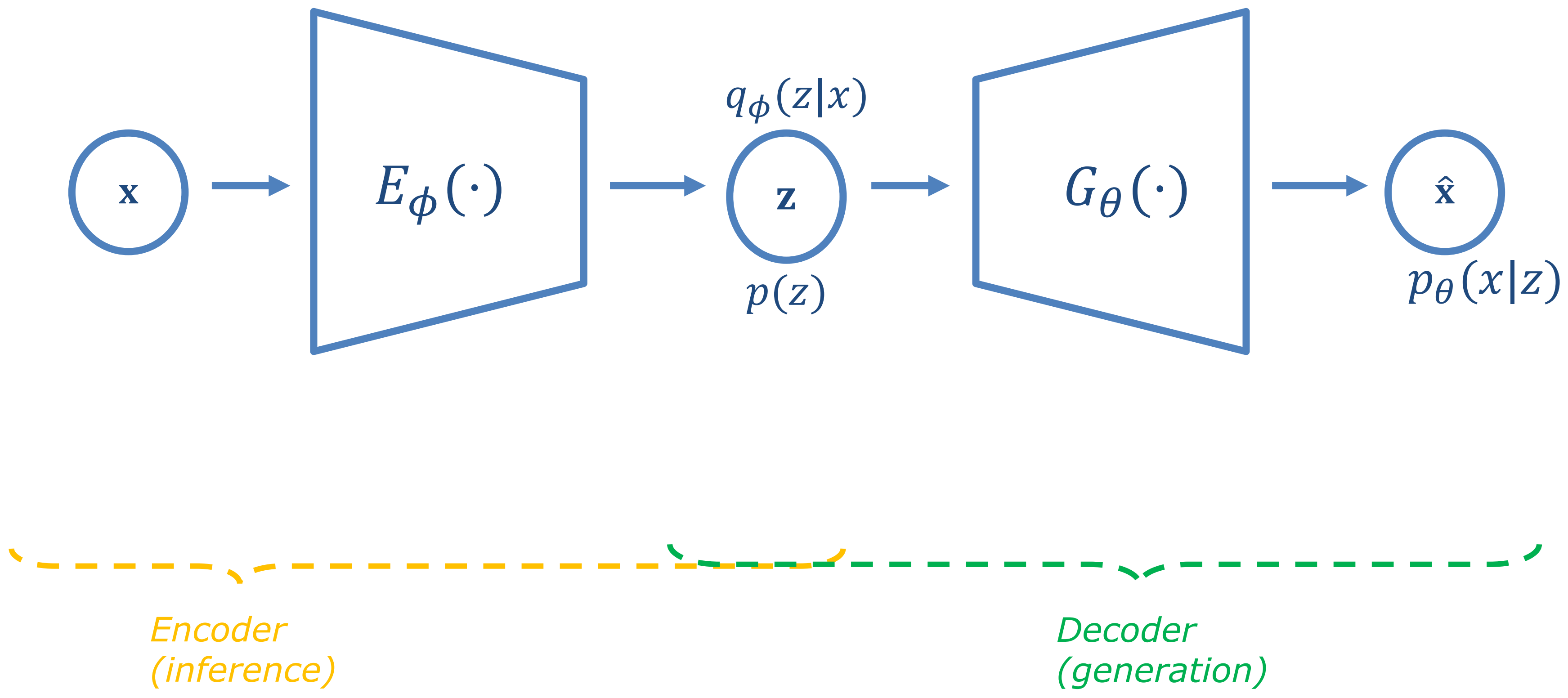
How do I remix existing stuff??

Conditional generative models

What if I have information $\mathbf{c}$ to *condition* the generation/inference, e.g., class labels?

- Just introduce them in the networks!

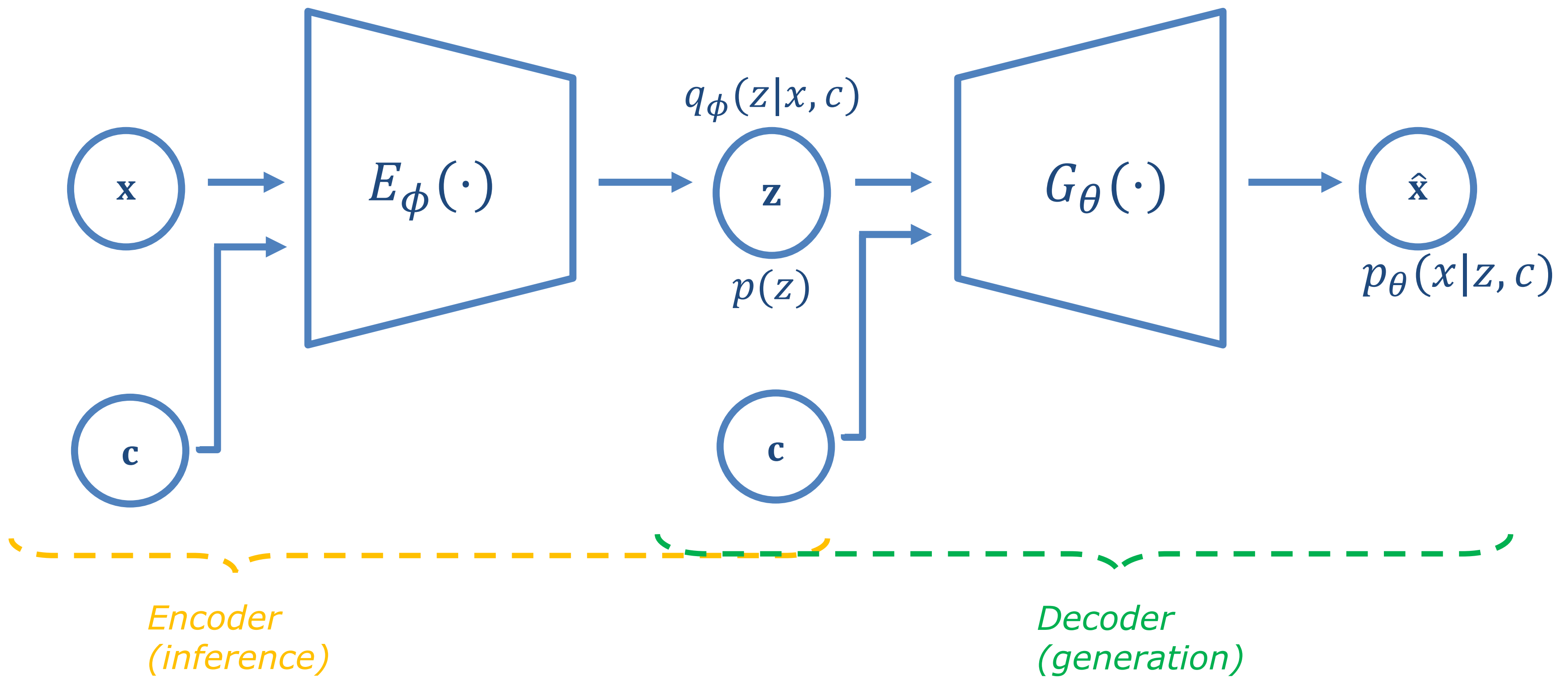
Variational Autoencoder



Conditional generative models

What if I have information $\mathbf{c}$ to *condition* the generation/inference, e.g., class labels?

- Just introduce them in the networks!



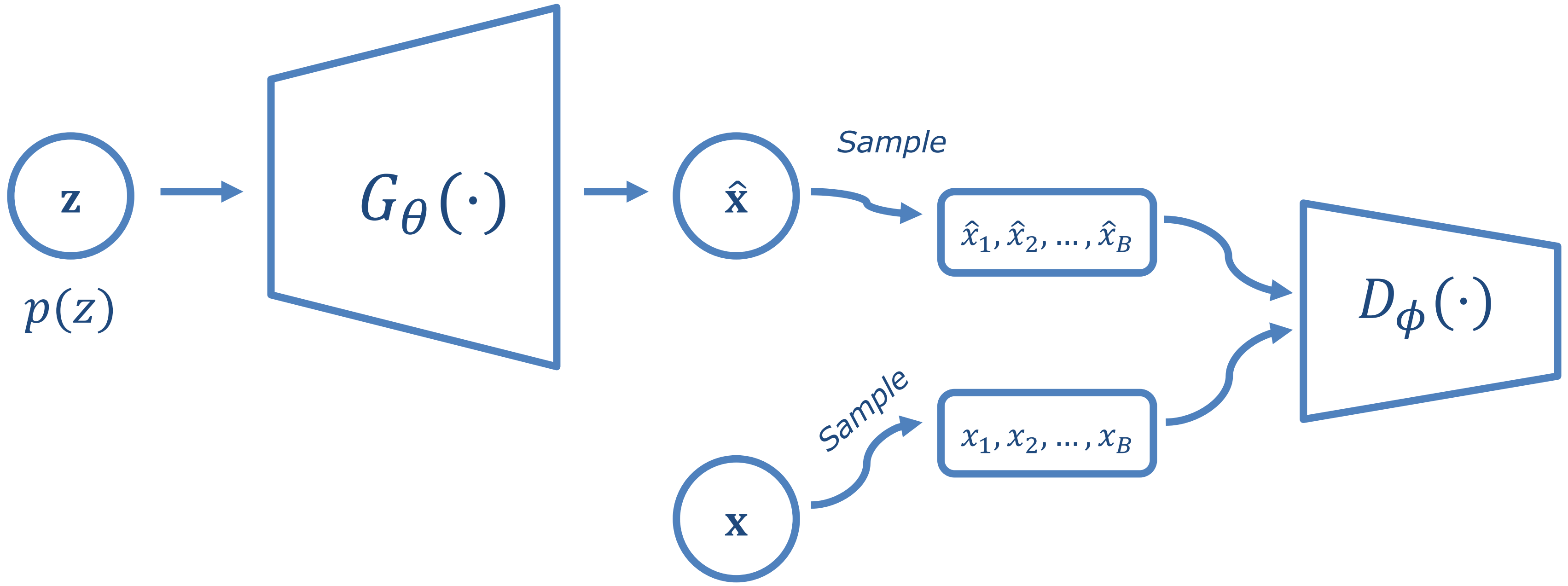
Conditional Variational Autoencoder

Conditional generative models

What if I have information $\mathbf{c}$ to *condition* the generation/inference, e.g., class labels?

- Just introduce them in the networks!

GAN

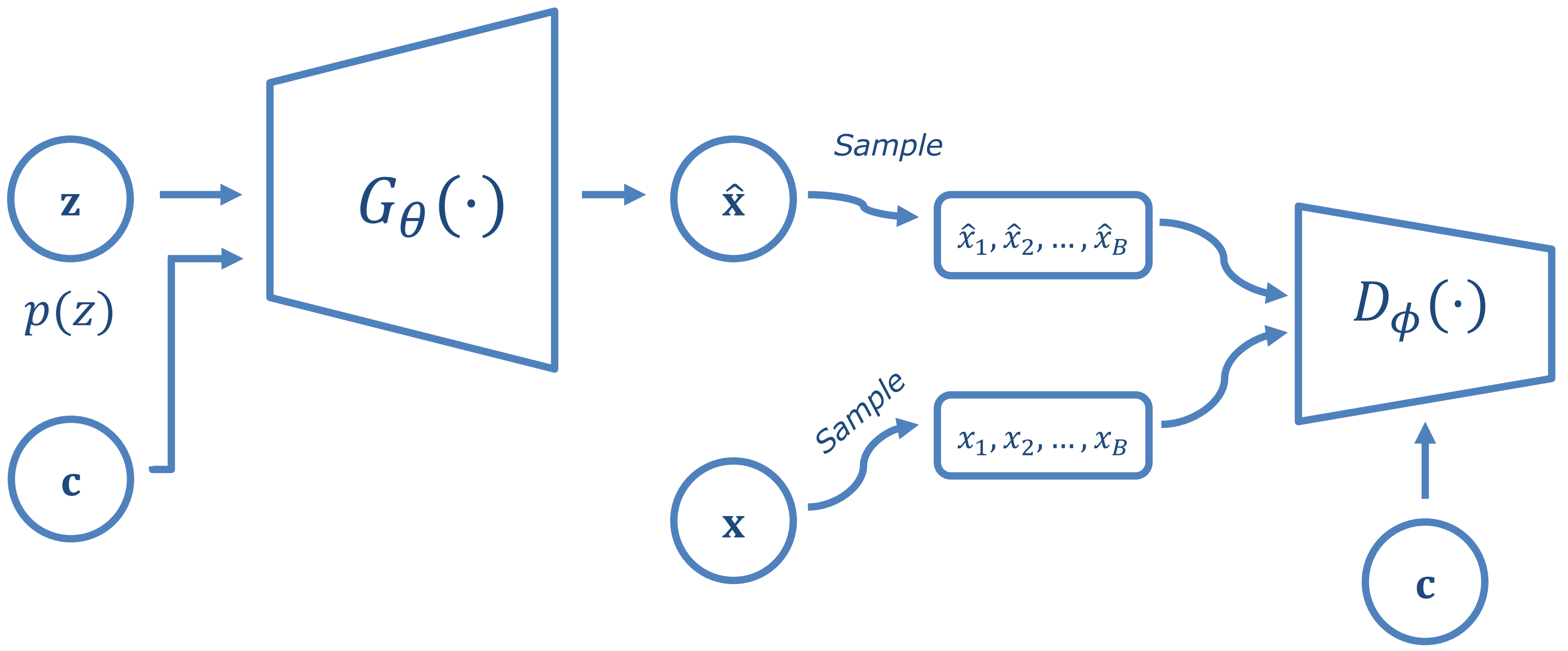


Conditional generative models

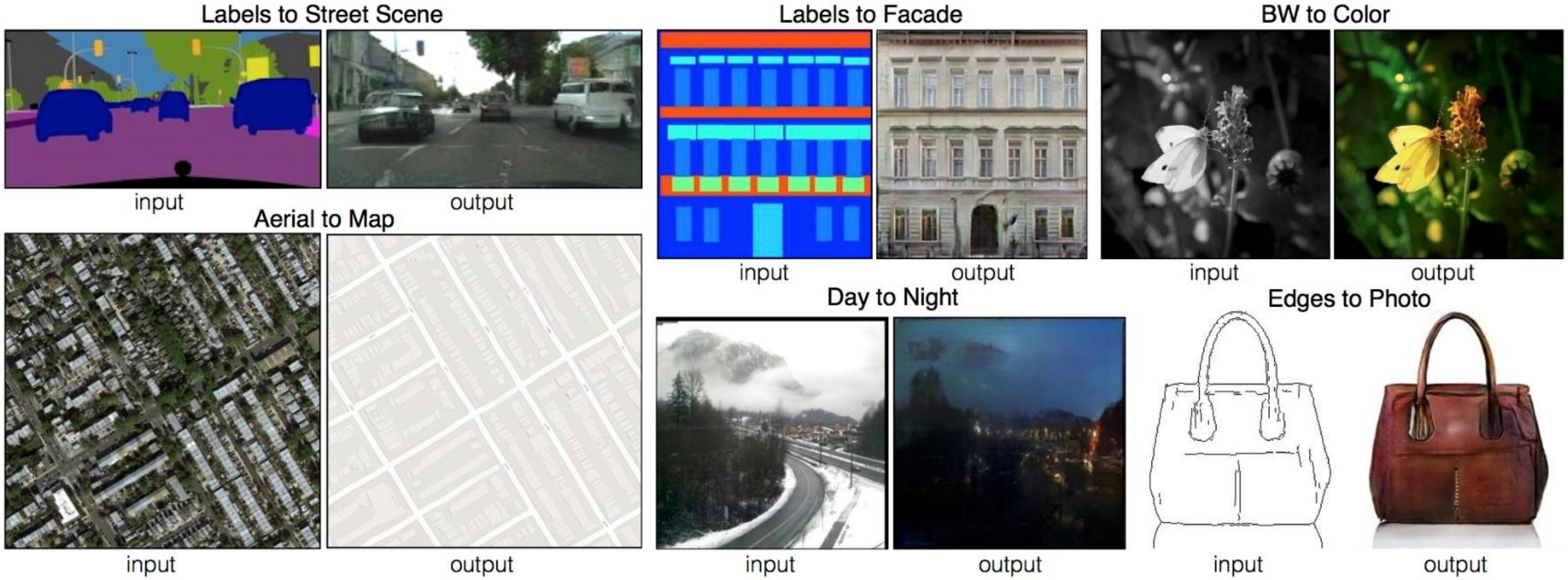
What if I have information $\mathbf{c}$ to *condition* the generation/inference, e.g., class labels?

- Just introduce them in the networks!

Conditional GAN

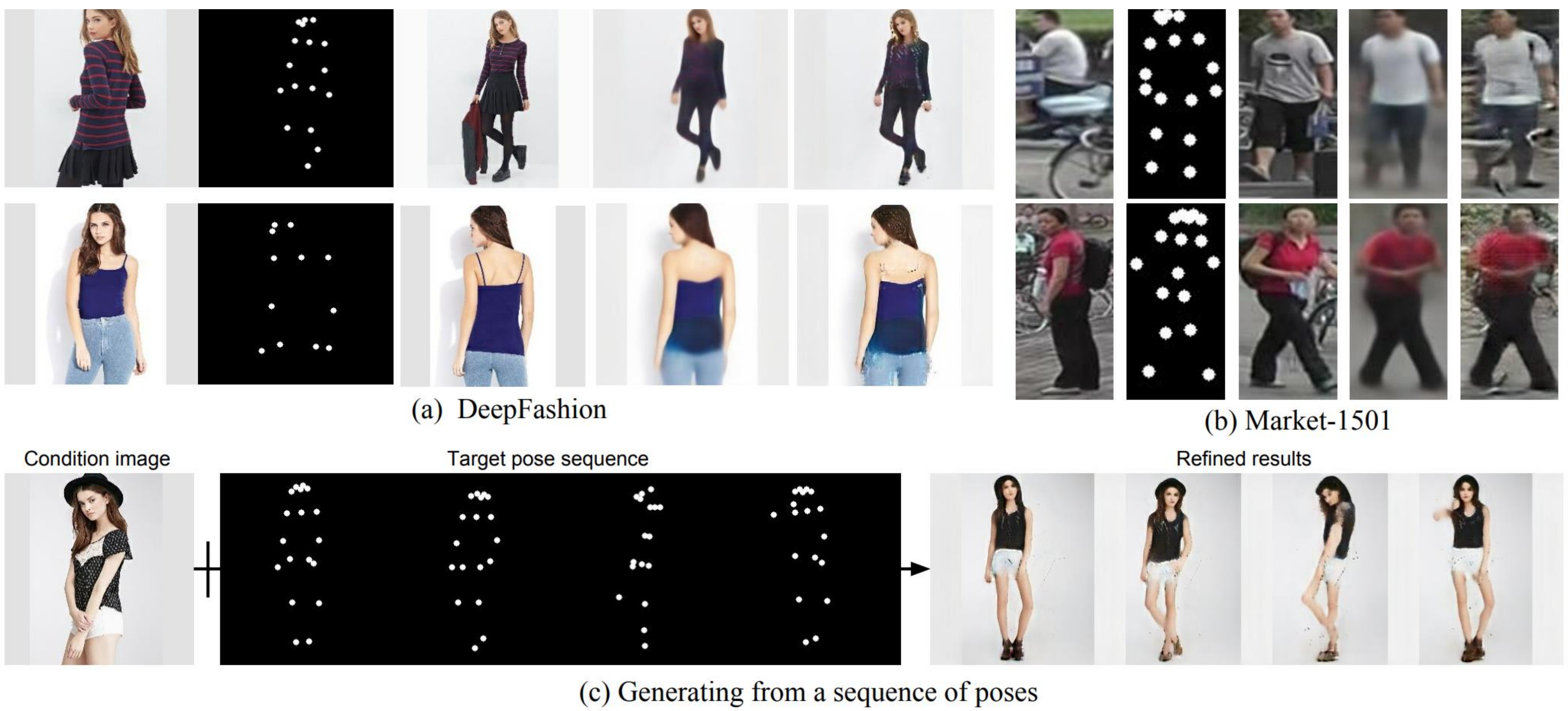


Conditional GMs are very important!



Pix2Pix: Image-to-Image Translation with Conditional Adversarial Networks. Isola et al.

Conditional GMs are very important!



Pose Guided Person Image Generation. Ma et al. 2017.

Some applications to game dev so far?

Generation of terrain

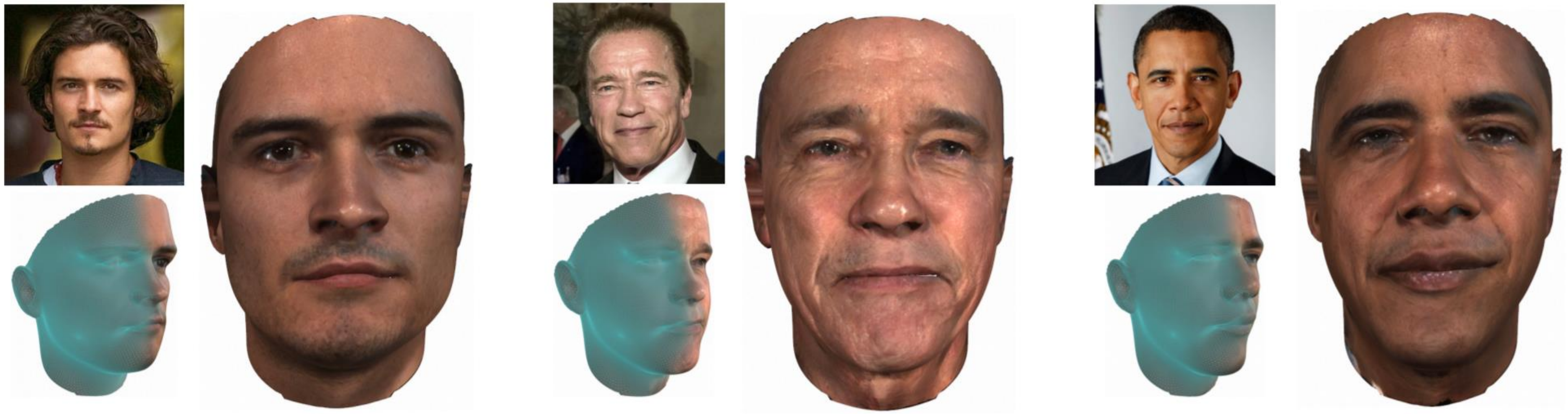
Interactive Example-Based Terrain Authoring with Conditional Generative Adversarial Networks. Guérin et al. 2017.

3D Content Generation

Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling. Wu et al. 2016

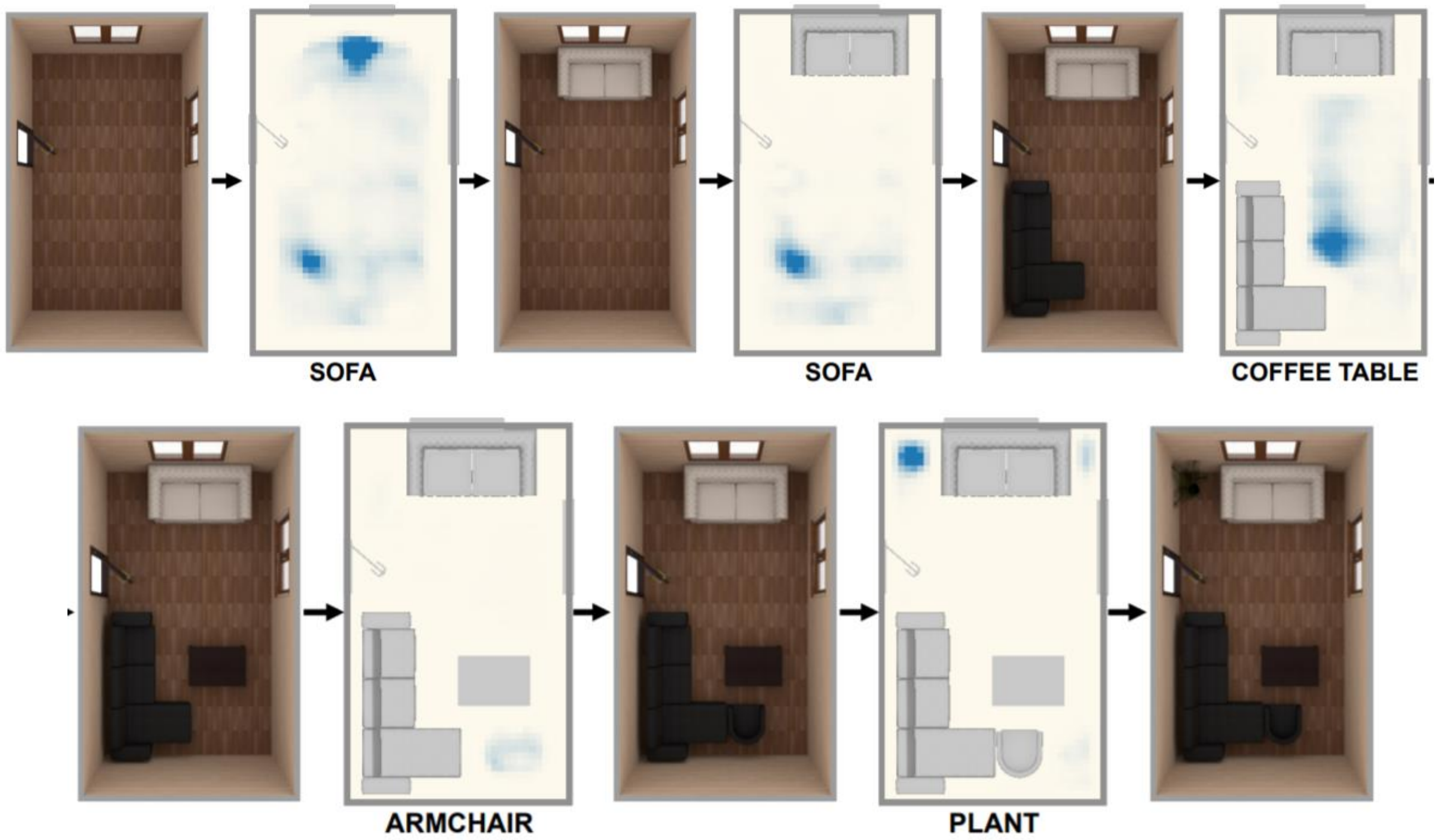
DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. Park et al. 2019.

Face generation



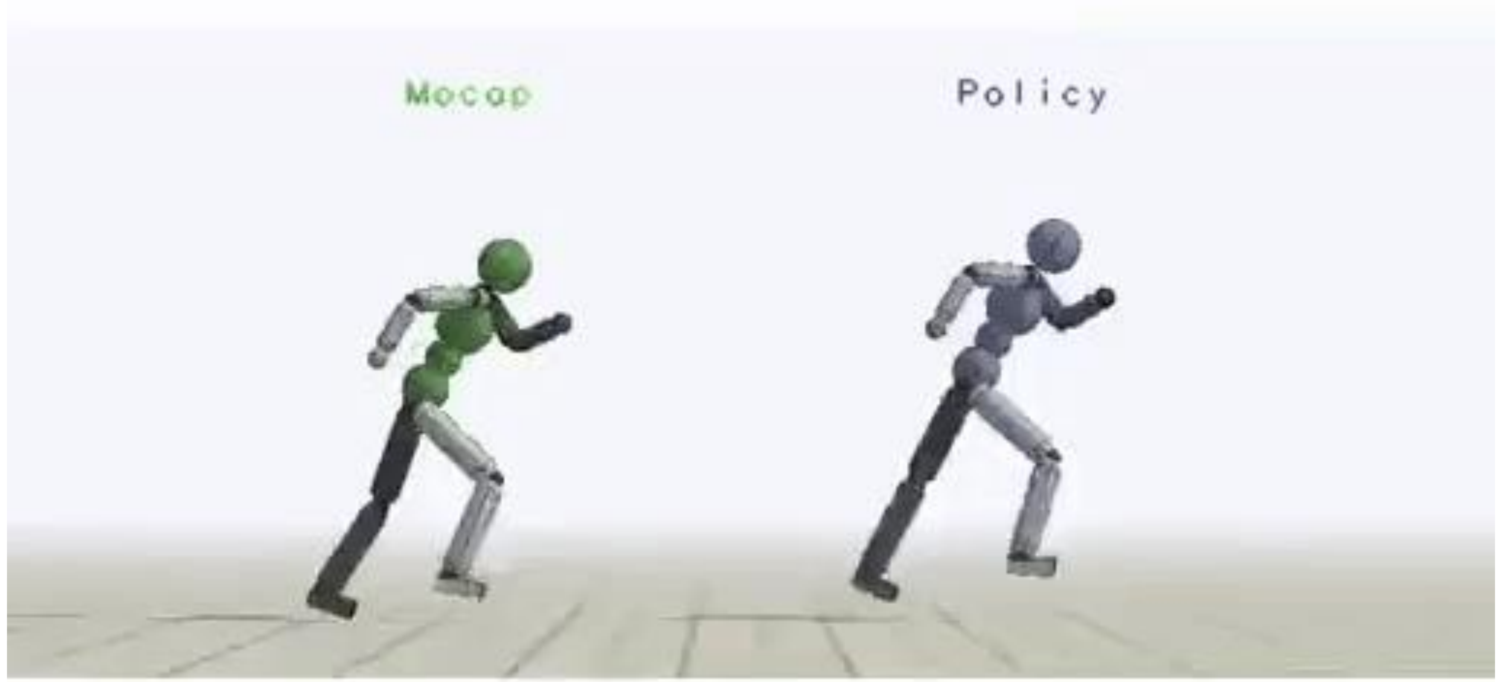
GANFIT: Generative Adversarial Network Fitting for High Fidelity 3D Face Reconstruction. Gecer et al. 2019 (FaceSoft.io)

Procedural placement



Deep Convolutional Priors for Indoor Scene Synthesis. Wang et al. 2018.

Generation of behaviour policies

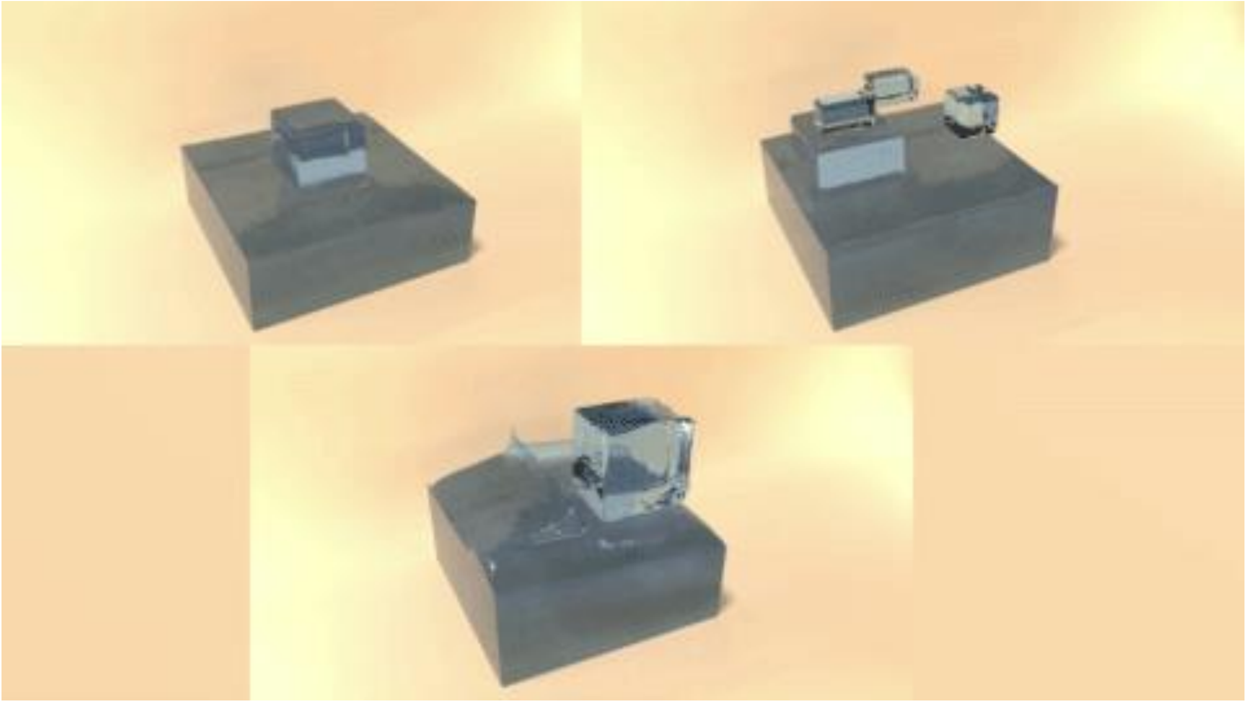


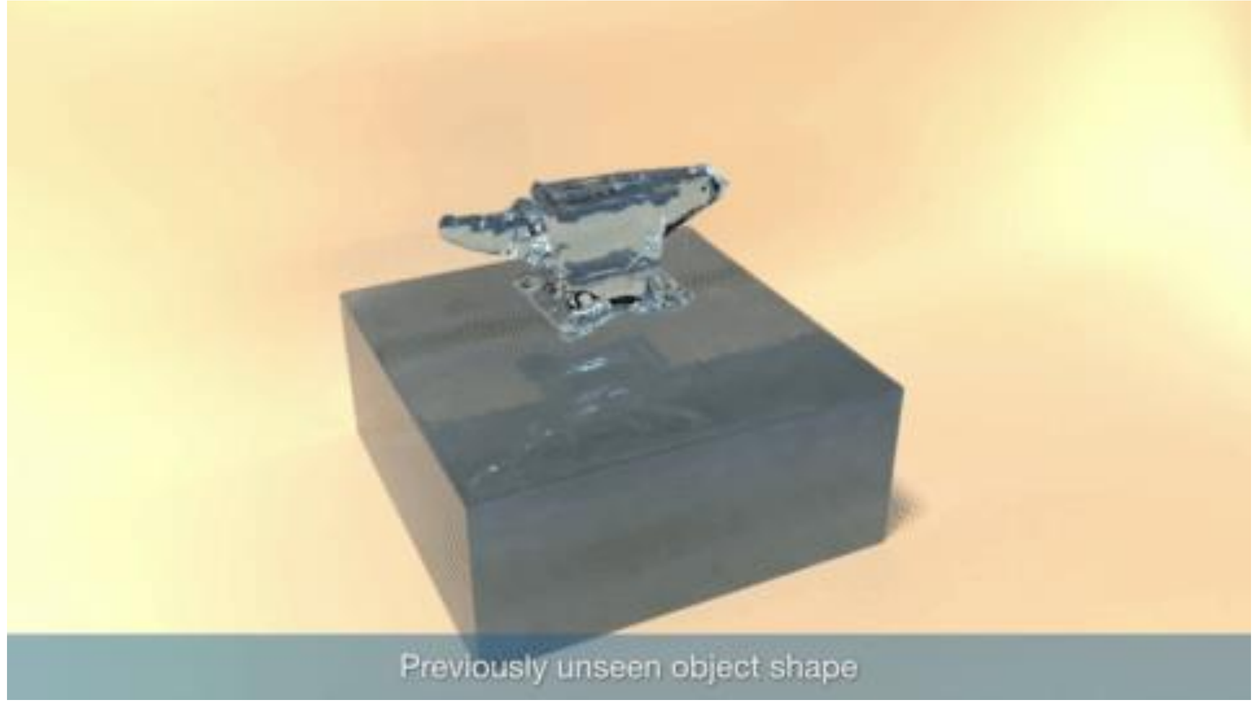
Variational Discriminator Bottleneck: Improving Imitation Learning, Inverse RL, and GANs by Constraining Information Flow. Peng et al. 2018.

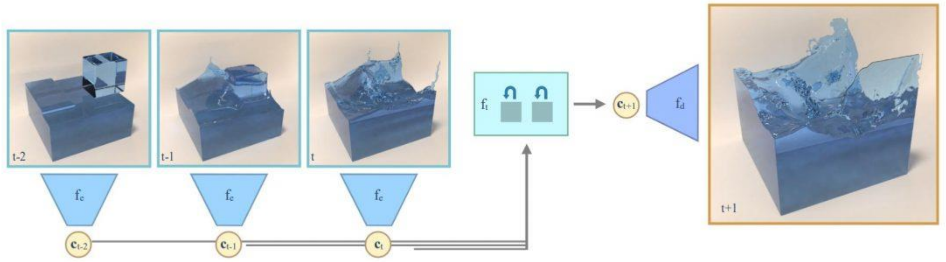
Generation of behaviour policies

Imitation Learning with Concurrent Actions in 3D Games. Harmer et al. 2018 (SEED)

Learn and accelerate physics







Latent-space Physics: Towards Learning the Temporal Evolution of Fluid Flow. Wiewel et al. 2018

Thanks for inspiration and insightful conversations ☺

Anastasia Opara
Camilo Gordillo
Colin Barré-Brisebois
Hector Anadón León
Henrik Johansson
Jack Harmer
Joakim Bergdahl
Johan Andersson
Kristoffer Sjöö
Ken Brown
Linus Gisslen
Martin Singh-Blom
Mattias Teye
Mark Cerny
Mónica Villanueva Aylagas
Magnus Nordin
Olivier Pomarez
Paul Greveson
Roy Harvey
Tomasz Stachowiak

SEED // SEARCH FOR EXTRAORDINARY EXPERIENCES DIVISION

STOCKHOLM – LOS ANGELES – MONTRÉAL – REMOTE

SEED.EA.COM

WE'RE HIRING!

Jorge del Val Santos
jdelvalsantos@ea.com