Lecture 11 - Deep generative models

DD2424

April 26, 2021

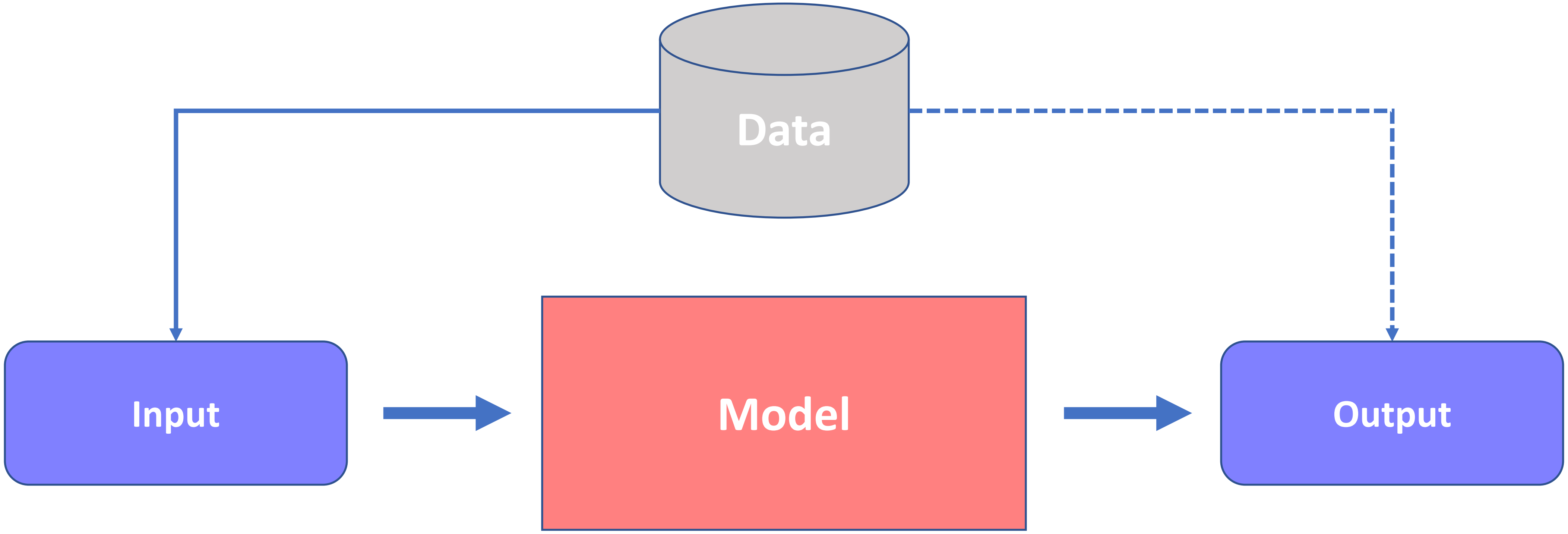
Outline

- Generative Modelling
- Variational Auto Encoders
- Generative Adversarial Networks
- Other methods

Outline

- Generative Modeling
- Variational Auto Encoders
- Generative Adversarial Training
- Other methods

Machine Learning

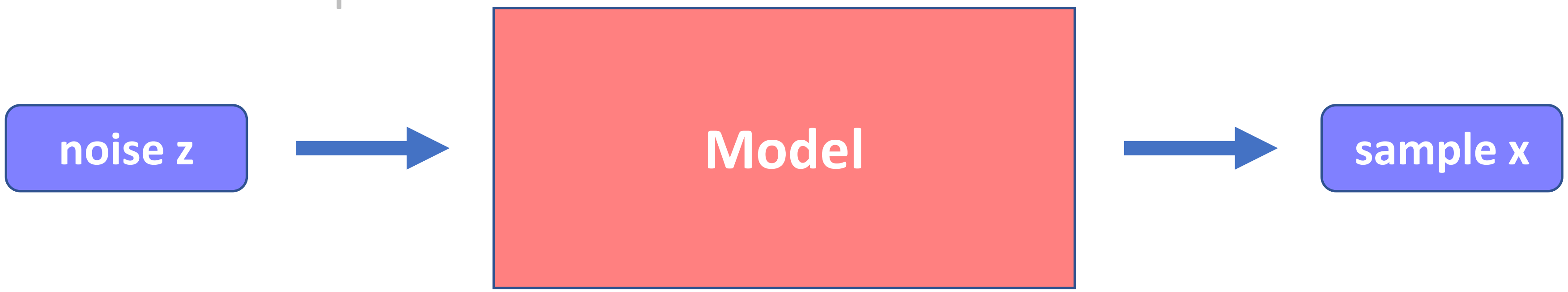


Most Common Models

- Most common setup: Supervised Discriminative Learning
  - Supervised: Correct output (labels) are provided for a set of input examples
  - Discriminative: Directly model the correct output given the input
  - objects given image, pixel-level depth given image, sentiment given a text,
  - Most famous architectures are designed for a supervised discriminative task: AlexNet, Inception, ResNet, FCN, U-Net, LSTM, etc.

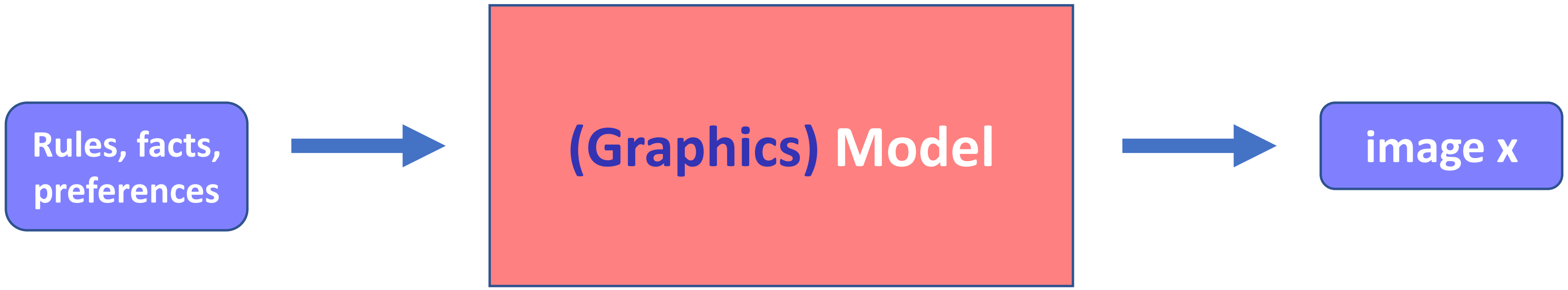
Goals

- Generate realistic samples (from the same distribution as training data, *i.e.*, P(**x**))
  - latent variable models
  - fully-observable models
- Assign likelihood to samples



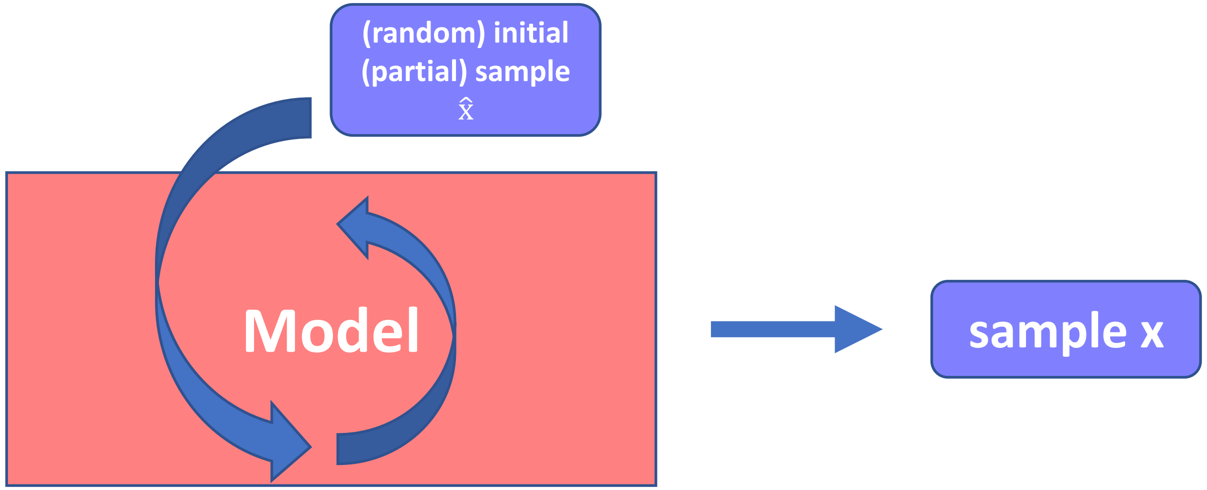
Goals

- Generate realistic samples a concrete case
- Assign likelihood to samples



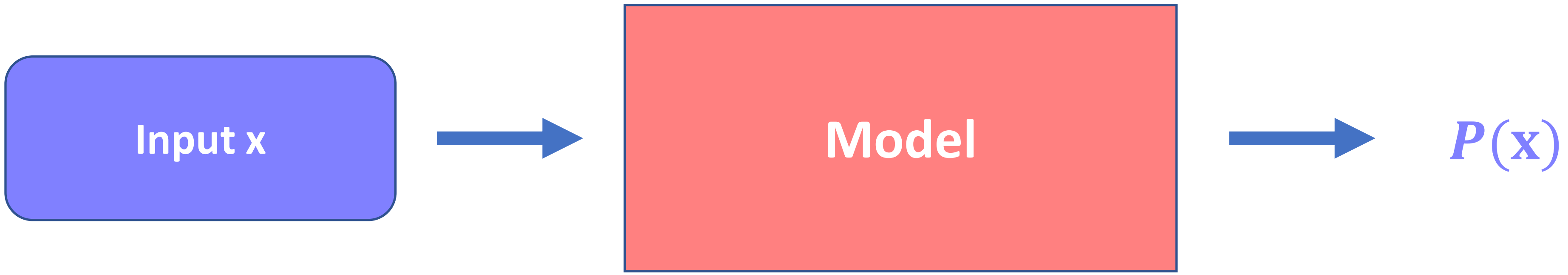
Goals

- Generate realistic samples (from the same distribution as training data, *i.e.*, P(**x**))
  - latent variable models
  - fully-observable models
- Assign likelihood to samples



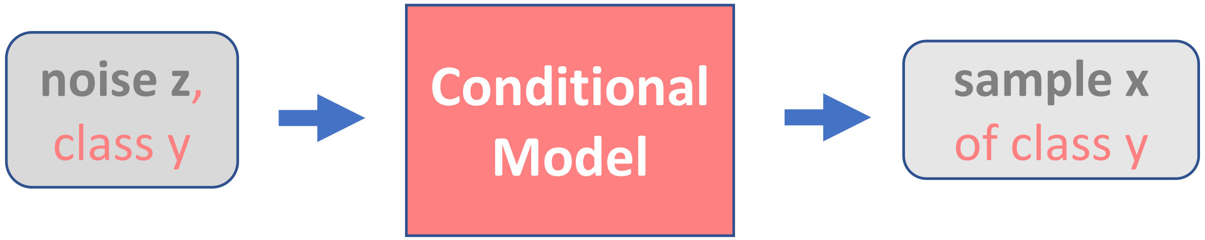
Goals

- Generate realistic samples
- Assign likelihood to samples (density Estimation)

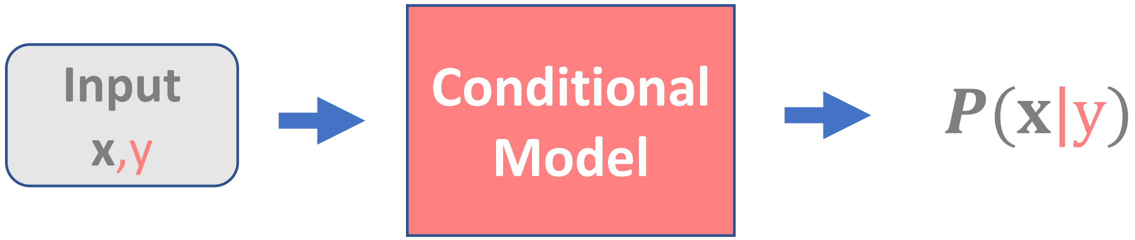


Goals

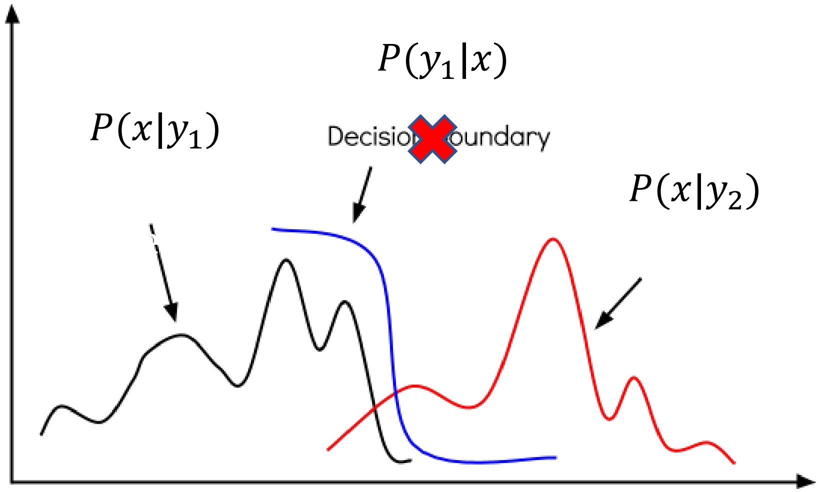
- Generate realistic samples



- Assign likelihood to samples (Density Estimation)

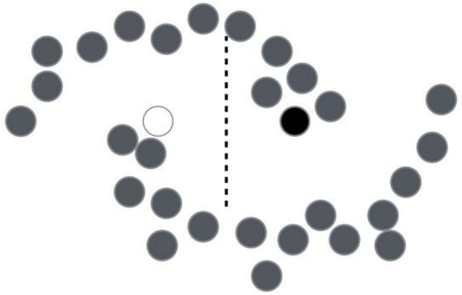


- Machine learning techniques as probabilistic models:
  - Generative Models: $P(\mathbf{x})$ or $P(\mathbf{x}|y)$ or maybe $P(\mathbf{x}, y)$
  - Discriminative models: $P(y|\mathbf{x})$
    - we can get $P(y|\mathbf{x}) = P(\mathbf{x}, y)/P(\mathbf{x})$ in generative models, especially if we are not interested in normalized probabilities: $P(y|\mathbf{x}) \propto P(\mathbf{x}, y) = P(\mathbf{x}|y)P(y)$

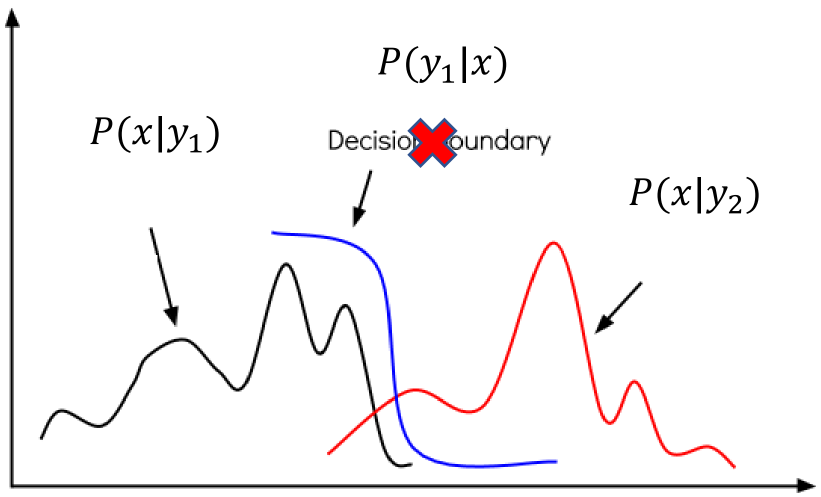


Generative Models of $P(y|\mathbf{x})$

- Pros:
  - Out of distribution samples
  - Missing data
  - Missing dimensions
  - Semi-supervised learning
  - Synthetic sample generation
  - …
- Cons
  - Number of data points
  - Number of assumptions



Despite the dimensionality of the input space ($\mathcal{X}$), why can we have a chance to train generative models in lack of enough data?



Generative Models – Latent Variable Models

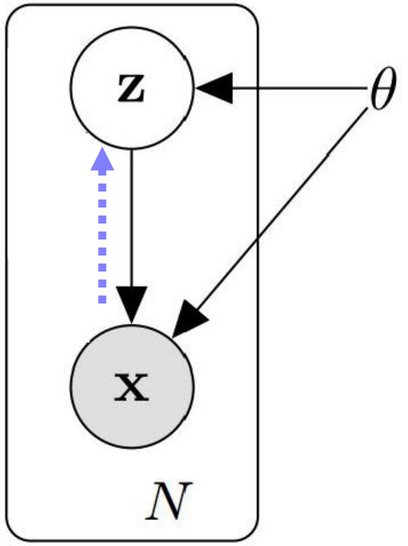
- Underlying factors of variations, using simpler lower dimensional hidden variables, $\mathbf{z}$

$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$

$\mathbf{x} \in \mathbb{R}^{\mathbf{32*32}}$

$z^{(1)}$: horizontal location
$z^{(2)}$: vertical location
$z^{(3)}$: rotation

$\mathbf{z} \in \mathbb{R}^{\mathbf{3}}$



- $P(\mathbf{z})$ and $P(\mathbf{x}|\mathbf{z})$

  and *maybe* $P(\mathbf{z}|\mathbf{x})$

- $P(\mathbf{x}) = \int P(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})P(\mathbf{z}|\boldsymbol{\theta})d\mathbf{z}$

Goals

- Generate realistic samples

- Assign likelihood to samples (Density Estimation)

- $P(\mathbf{z}|\mathbf{x})$: (Compressed) Representation Learning
  - Understanding underlying generative factors (e.g. similar to what PCA or matrix factorization do)
  - Data compression
  - Semi-supervised learning
  - ....

Why are they called Generative models?

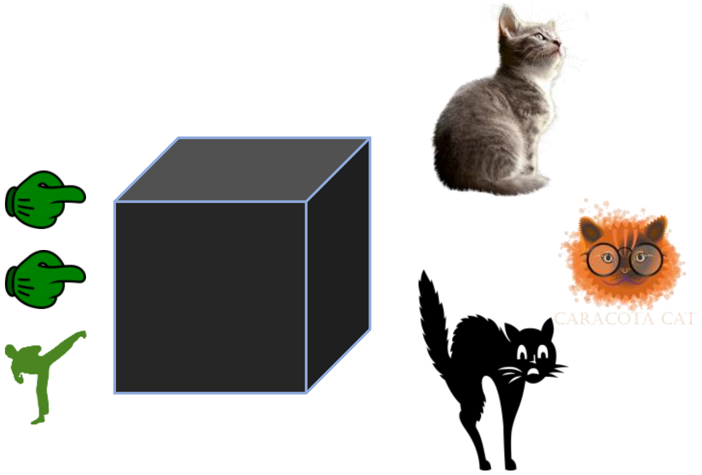
Sample from $P(\mathbf{x})$

Sample from $P(\mathbf{x}, y)$

- Sample y from $P(y)$
- Then sample $\mathbf{x}$ from $P(\mathbf{x}|y)$

That was the probabilistic approach, do all models do that though?

- Not always, sometimes:
  - Discriminative model: $\underset{y}{\operatorname{argmax}}\ score(\mathbf{x}, y)$
  - Generative model: $\mathrm{g}(\mathbf{z}) \rightarrow \hat{\mathbf{x}} \sim P(\mathbf{x})\ \ or\ \ \hat{\mathbf{x}}, \hat{y} \sim P(\mathbf{x}, y)$



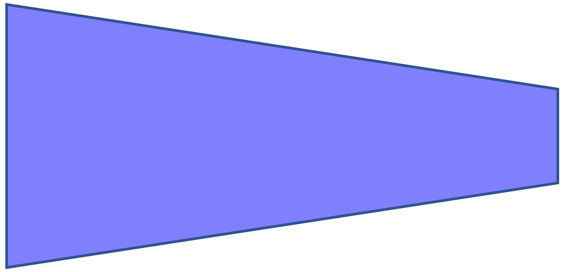
Another way to look at generative models

$\mathbf{x} = g(\mathbf{z})$

Information generation

Generative Modeling

$\mathbf{x}$



$\mathbf{z}$ / y

$\mathbf{z} = f(\mathbf{x})$

Information extraction

Discriminative Modeling

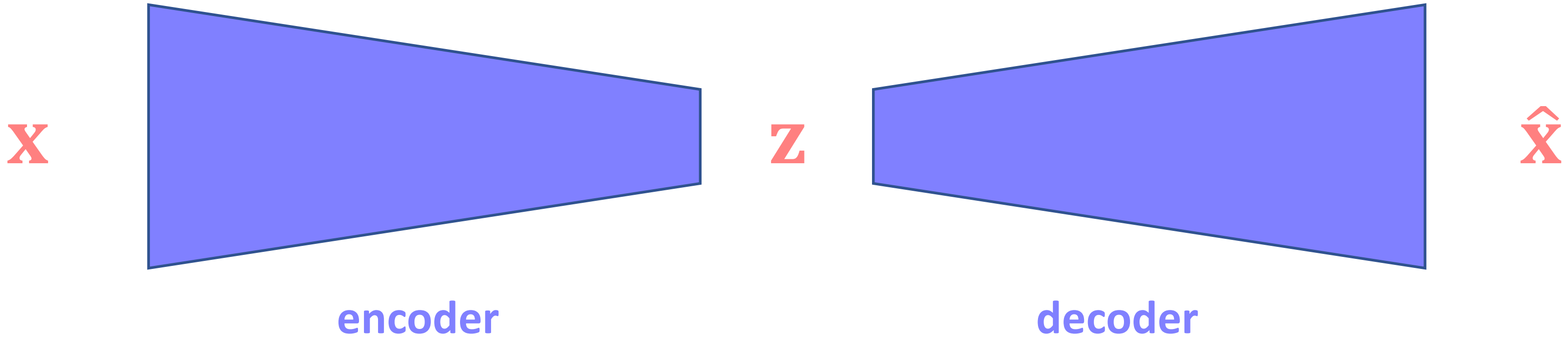
Deep Generative Models

Outline

- Generative Modelling
- Variational Auto Encoders
  - Auto Encoders
  - Steps to create Variational Auto Encoders
  - Some Discussions
- Autoregressive Models
- Normalizing Flow Models
- Energy-based Models

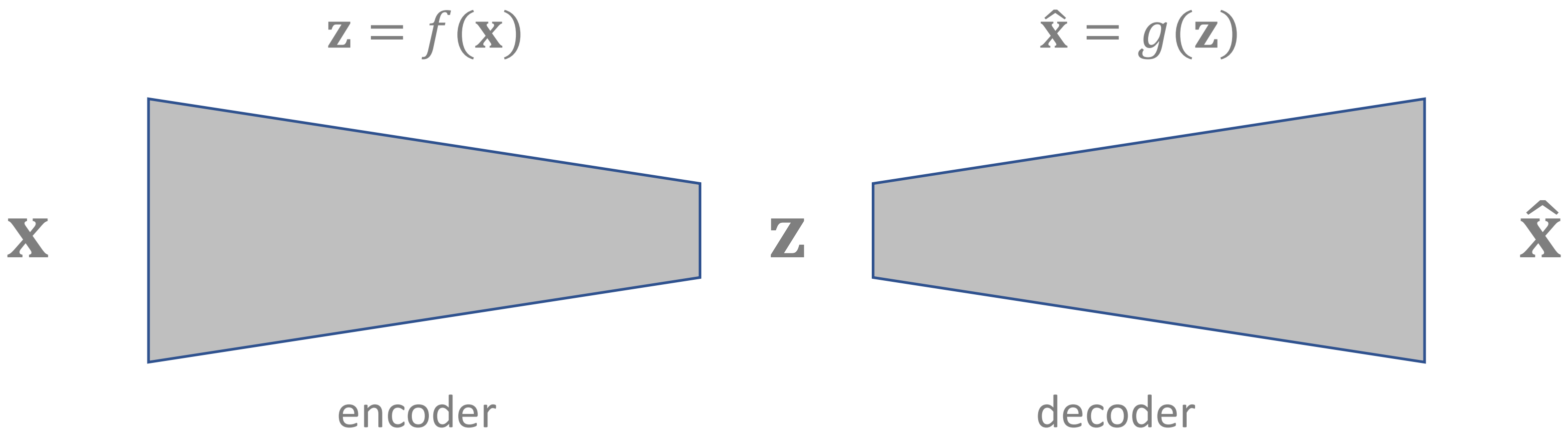
Bottleneck Auto-Encoder Networks

$\mathbf{z} = f(\mathbf{x})$ $\hat{\mathbf{x}} = g(\mathbf{z})$



$\mathbf{x}, \hat{\mathbf{x}} \in \mathbb{R}^d$ $\mathbf{z} \in \mathbb{R}^p$ $d \gg p$

Bottleneck Auto-Encoder Networks

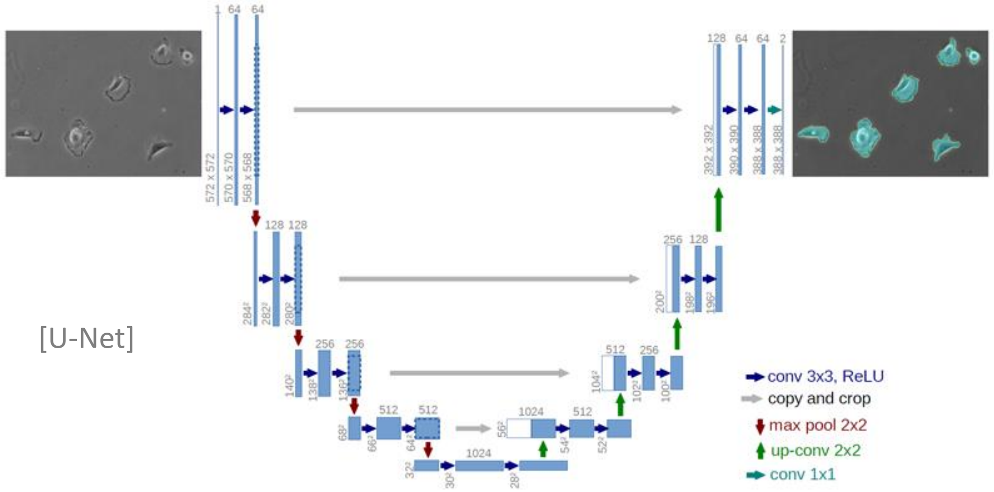


Loss function

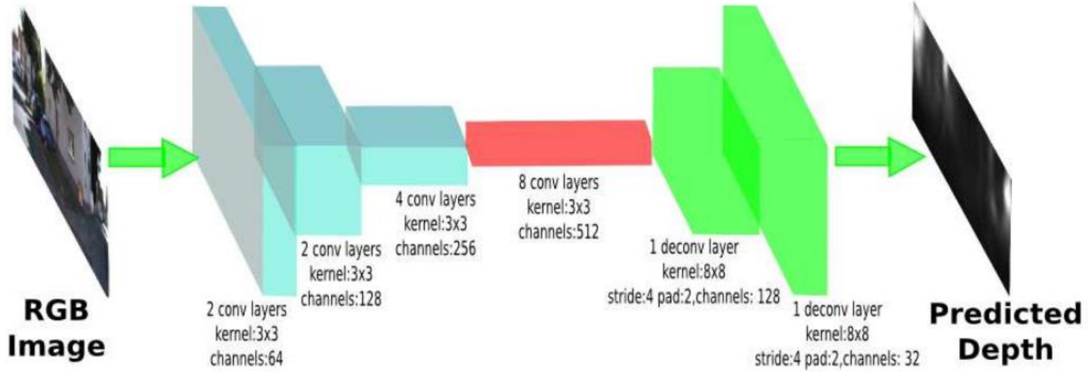
$$L = \sum ||\mathbf{x} - \hat{\mathbf{x}}||^2$$

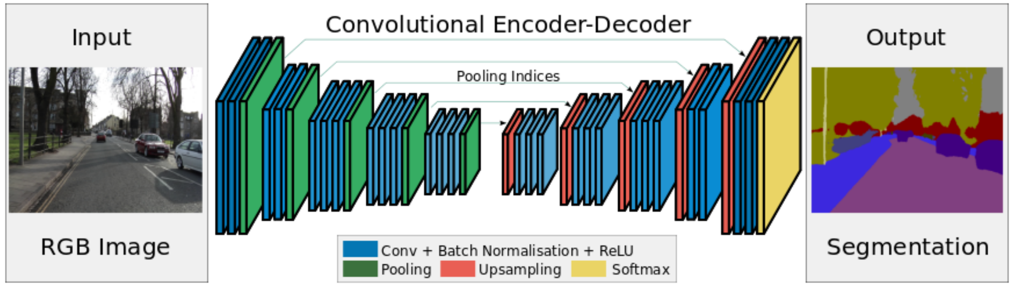
$\mathbf{x}, \hat{\mathbf{x}} \in \mathbb{R}^d$ $\quad$ $\mathbf{z} \in \mathbb{R}^p$ $\quad$ $d \gg p$

Have you seen that kind of bottleneck networks before?



[U-Net]

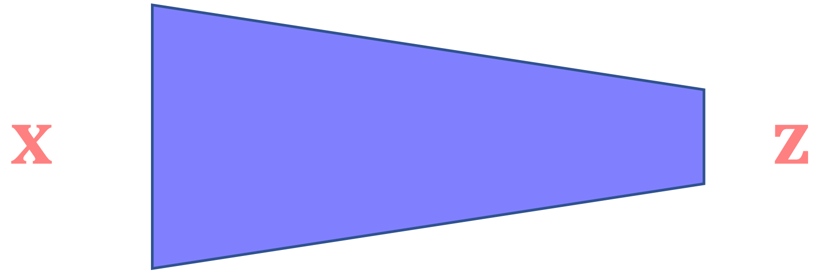




[SegNet]

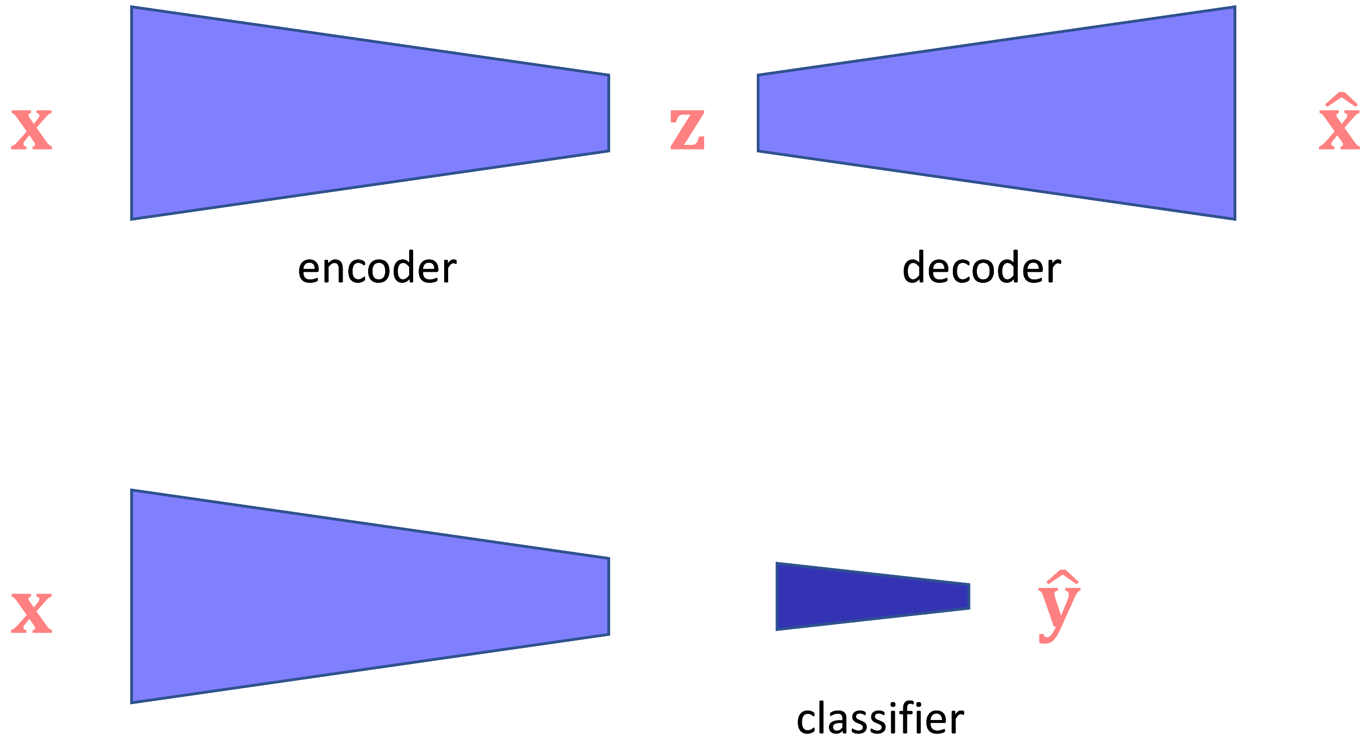
What can AEs be used for?

- Dimensionality Reduction
- Pretraining
- Denoising AutoEncoder
- Item imputation



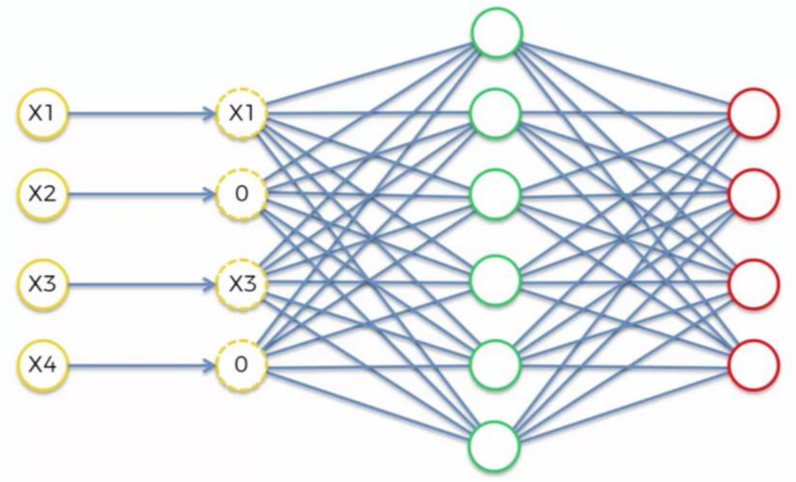
$\mathbf{x} \in \mathbb{R}^d$ $\mathbf{z} \in \mathbb{R}^p$ $d \gg p$

- **What can AEs be used for?**
- Dimensionality Reduction
- Pretraining
- Denoising AutoEncoder
- Item imputation

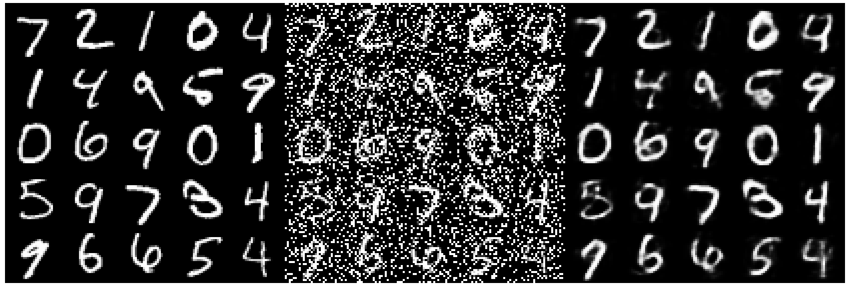


What can AEs be used for?

- Dimensionality Reduction
- Pretraining
- Denoising AutoEncoder
- Item imputation



[source: Kirill Eremenko]

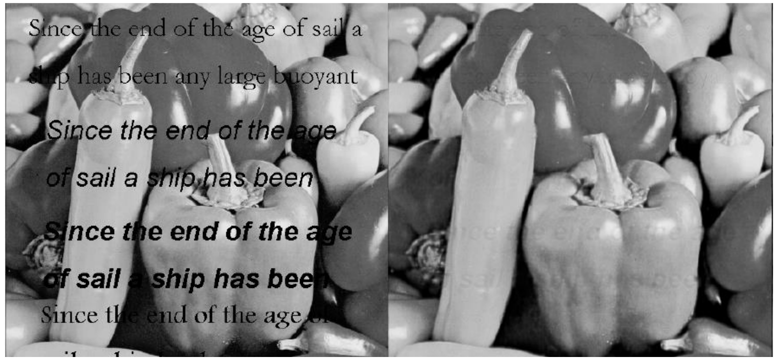


Vincent, H. Larochelle Y. Bengio and P.A. Manzagol, "**Extracting and Composing Robust Features with Denoising Autoencoders**", ICML 2008

What can AEs be used for?

Dimensionality Reduction

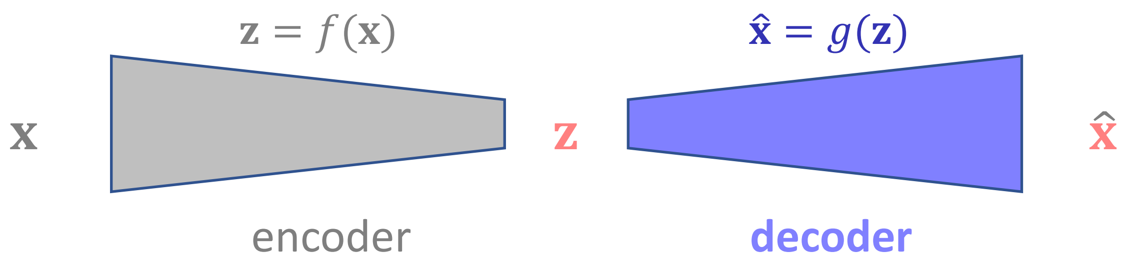
- Pretraining
- Denoising AutoEncoder
- Item imputation (e.g. Image Inpainting)



Auto-Encoder

Are Auto-Encoders generative models?

not in principle!



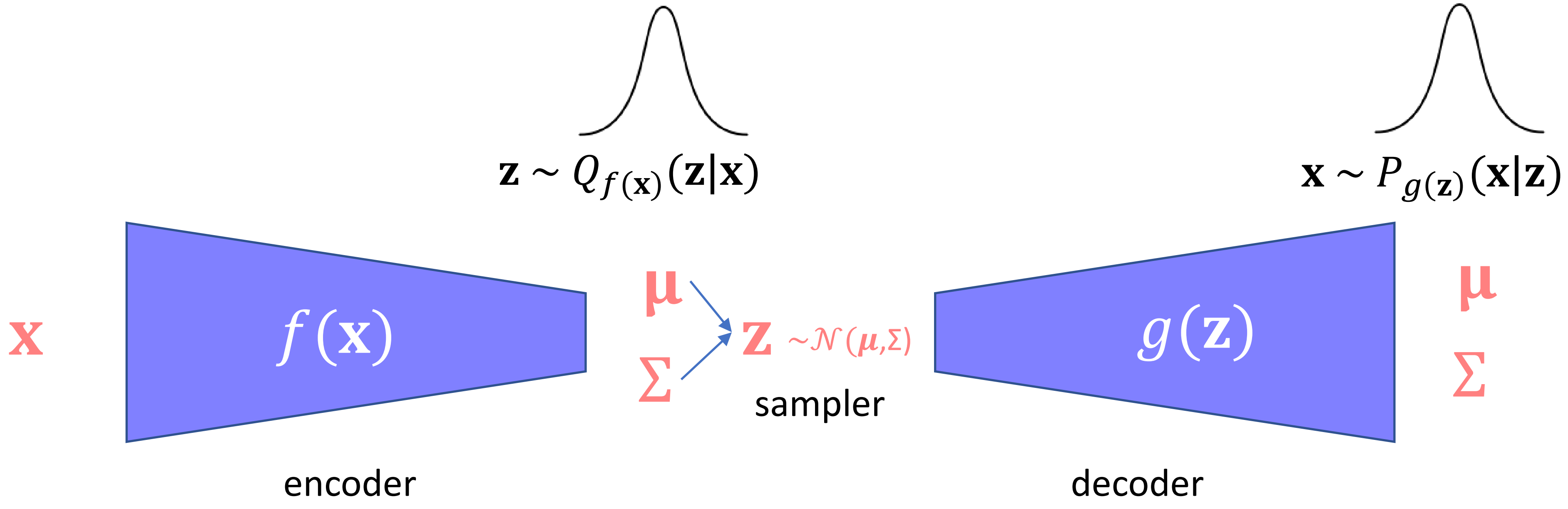
Bengio, Yao, Alain, Vincent, "**Generalized Denoising Auto-Encoders as Generative Models**", NIPS 2013

We can make it probabilistic and truly generative!

- While the outcome is deceivingly simple, several steps are needed to understand every choice!

Kingma, Welling "**Auto-Encoding Variational Bayes“,** ICLR 2013

implementation perspective



AutoEncoder - Variational Inference

$$\max_{\boldsymbol{\omega}=\{\boldsymbol{\omega}_i\}_{i=1:n}} \mathcal{L}_{VAE} = \sum_{i=1}^{n} \left(\mathbb{E}_{Q_{\boldsymbol{\omega}_i}(\mathbf{z}|\mathbf{x}_i)} \log P(\mathbf{x}_i|\mathbf{z}) - D_{KL}(Q_{\boldsymbol{\omega}_i}(\mathbf{z}|\mathbf{x}_i)||P(\mathbf{z})) \right)$$

VAE training implementation

$$\max_{\theta_f,\theta_g} \frac{1}{s}\sum_{i=1}^{n}\sum_{k=1}^{s}\left(\log\mathcal{N}(\mathbf{x}_i;\{\boldsymbol{\mu},\Sigma\} = g(\widehat{\mathbf{z}_i^k})) - D_{KL}(\mathcal{N}(\widehat{\mathbf{z}_i^k};\{\boldsymbol{\mu},\Sigma\} = f(\mathbf{x}_i))\,||\mathcal{N}(\boldsymbol{\mu} = \mathbf{0},\Sigma = \mathbb{I})\,)\right)$$

$$\left\{\widehat{\mathbf{z}_i^k}\right\}_{k:1..s} \sim \mathcal{N}(\mathbf{z};\{\boldsymbol{\mu},\Sigma\} = f(\mathbf{x}_i))$$

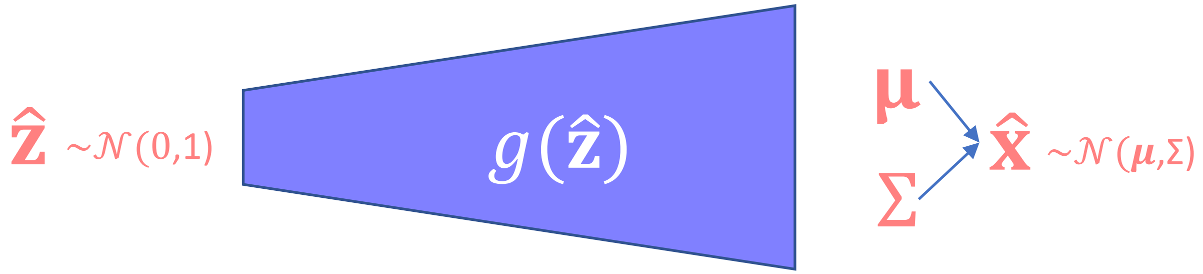
- Common case in practice
  - $P(\mathbf{z}) \quad = \quad \mathcal{N}(\boldsymbol{\mu} = \mathbf{0},\Sigma = \mathbb{I})$
  - $P(\mathbf{x}|\mathbf{z}) \; = \; \mathcal{N}(\mathbf{z};\{\boldsymbol{\mu}_\mathbf{x} = g(\mathbf{z}),\mathbb{I}\}$
  - $Q(\mathbf{z}|\mathbf{x}) \quad = \; \mathcal{N}(\mathbf{x};\{\boldsymbol{\mu}_\mathbf{z},\boldsymbol{\sigma}_\mathbf{z}\mathbb{I}\} = f(\mathbf{x}))$
  - Single-sample monte carlo estimation ($s = 1$)

$$\min_{\theta_f,\theta_g} \mathcal{L}(\theta_f,\theta_g,X) \triangleq \sum_{i=1}^{n}\sum_{j=1}^{p}\left(\frac{\left(\mu_{\mathbf{x},j} - x_{i,j}\right)^2}{2} + \log\sigma_{\mathbf{z},j} + \frac{\sigma_{\mathbf{z},j}^2 + \mu_{\mathbf{z},j}^2}{2}\right)$$

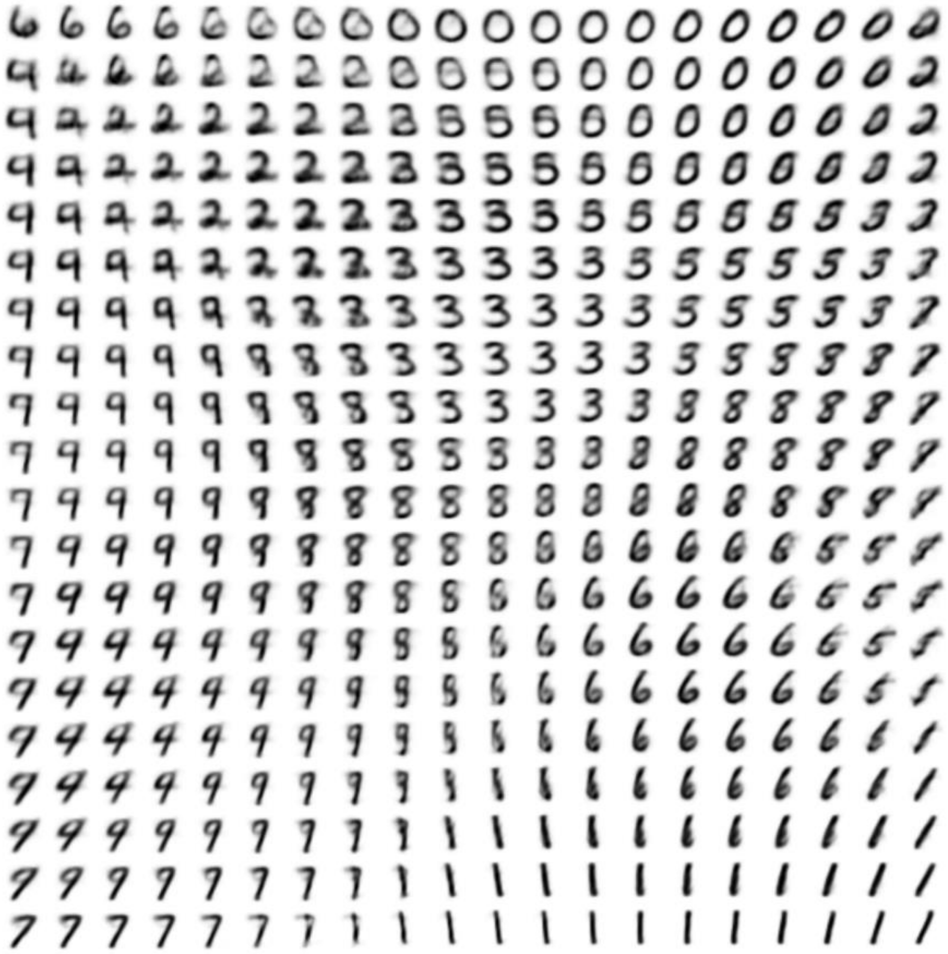
How to sample from $P(\mathbf{x})$ using the trained VAE?

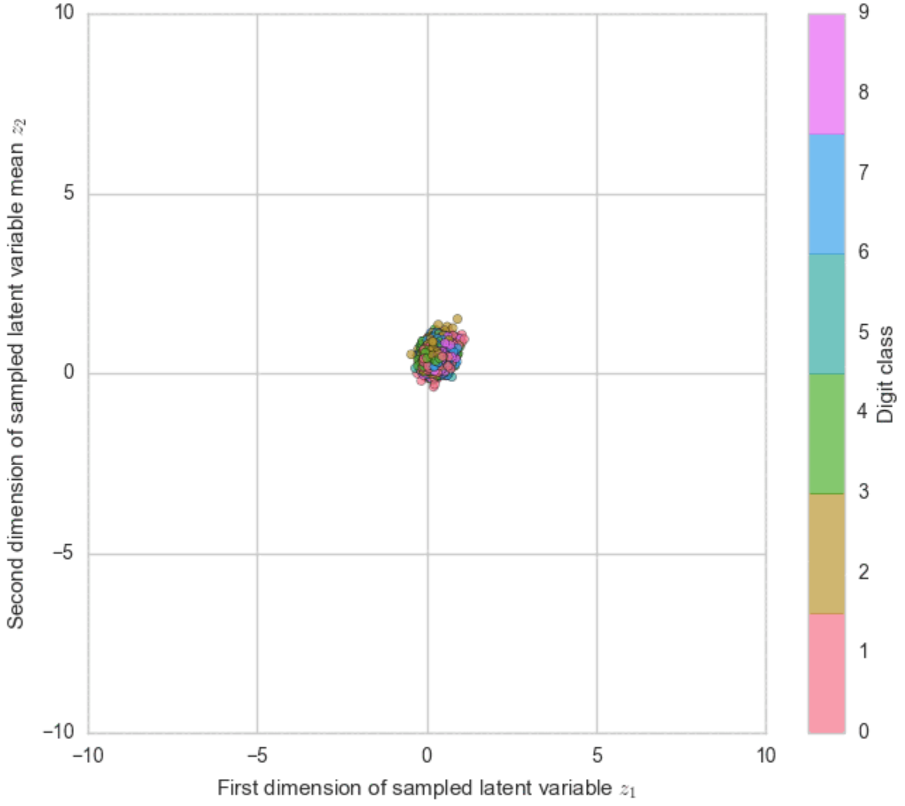
VAE as generative model

- Sample $\mathbf{z}$ from $\mathcal{N}(\mathbf{0}, \mathbf{1})$
- Then sample $\mathbf{x}$ using from $P(\mathbf{x}|\mathbf{z})$



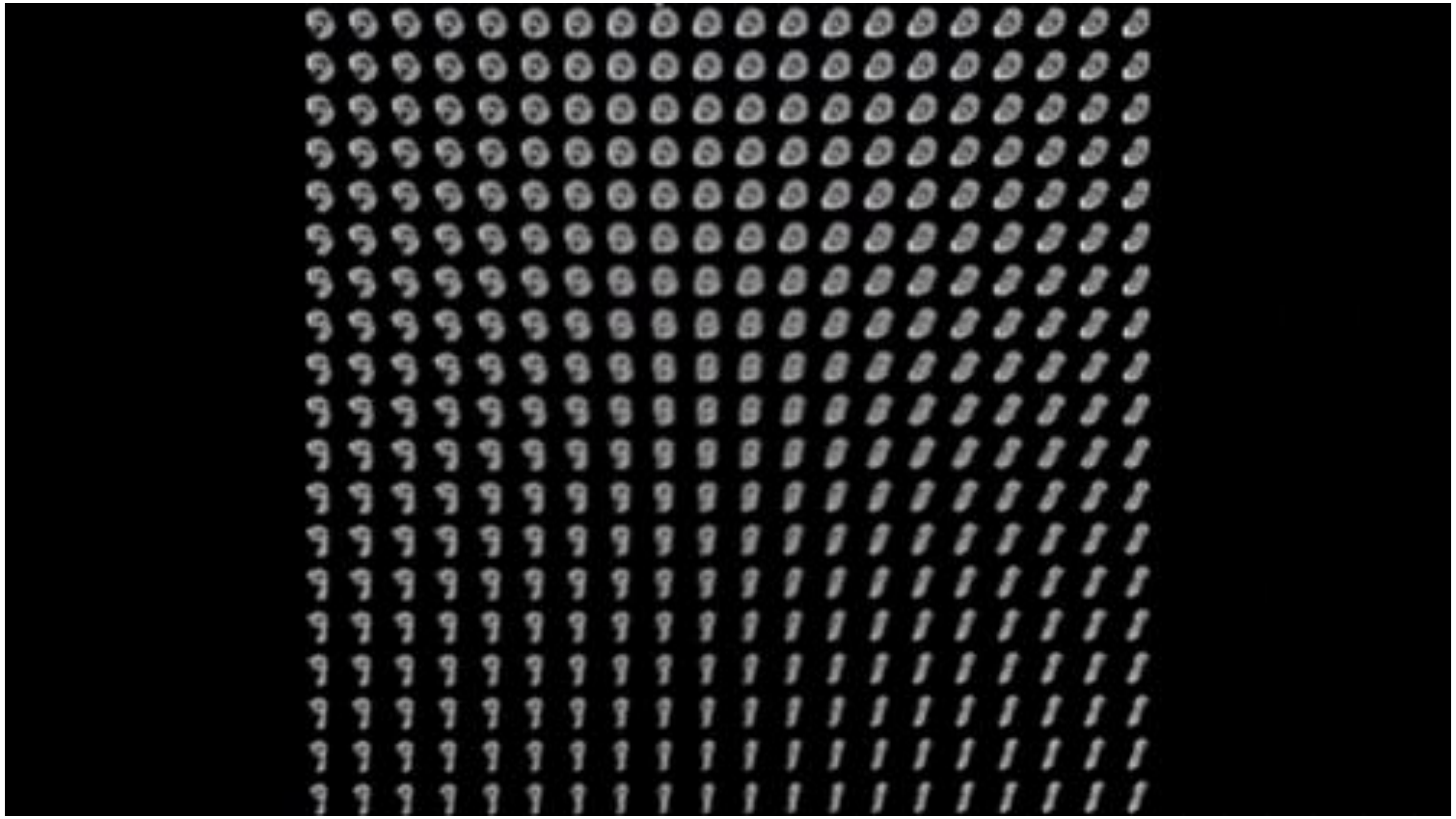
VAE as generative model





https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

VAE (example)



- Tries to disentangle the latent representations

- What does it mean?
  - Each factor $z^{(i)}$ is uncorrelated with other factors

- Why is this good?
  - Better interpretability of the latent space

- How to do this?
  - Remember the prior P(**z**) had a diagonal covariance
  - We can enforce that more strongly

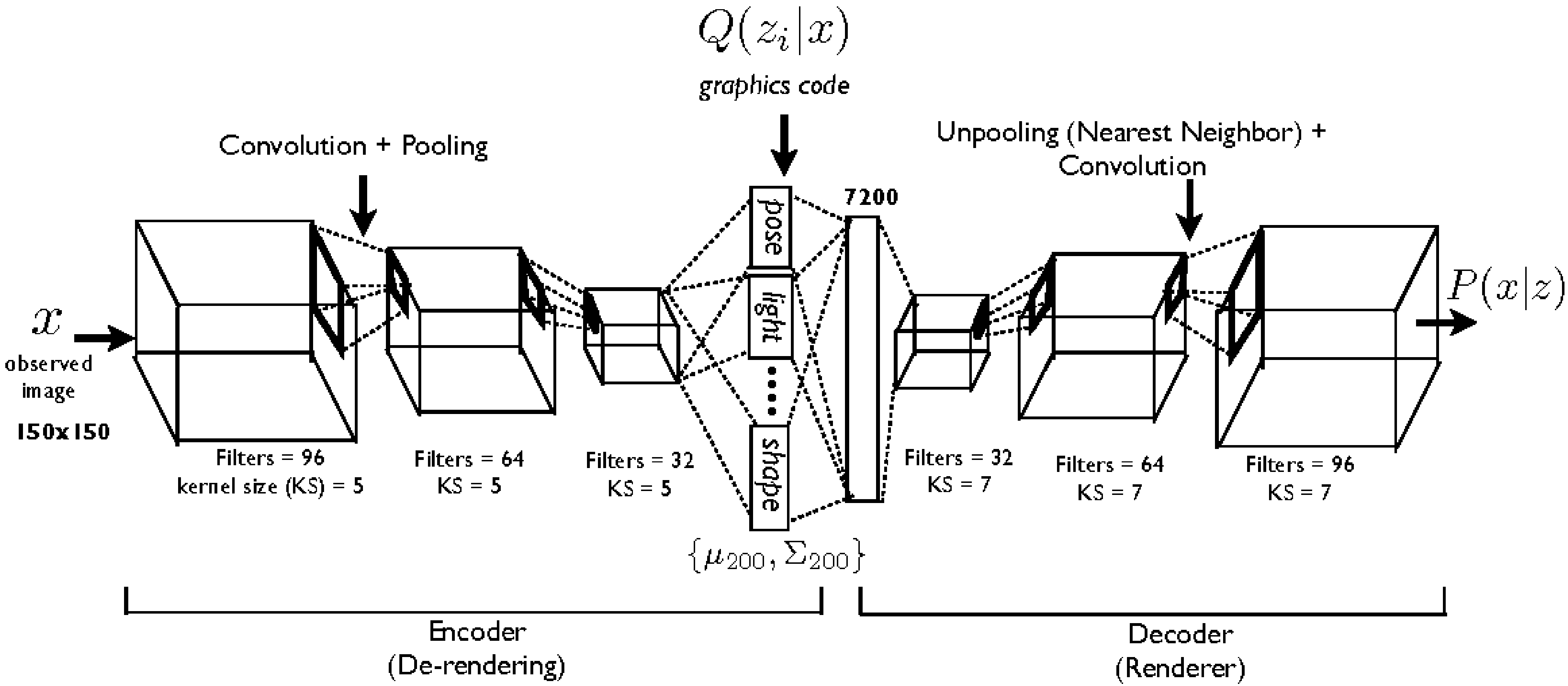
Higgins et al. "**β-VAE: LEARNING BASIC VISUAL CONCEPTS WITH A CONSTRAINED VARIATIONAL FRAMEWORK**", ICLR 2017

VAE - βVAE

$$\max_{\omega} \sum_{i=1}^{n} \left(\mathbb{E}_{Q_{\omega_i}(\mathbf{z}|\mathbf{x}_i)} \log P(\mathbf{x}_i|\mathbf{z}) - \beta D_{KL}(Q_{\omega_i}(\mathbf{z}|\mathbf{x}_i)||P(\mathbf{z})) \right)$$

Higgins et al. “**β-VAE: LEARNING BASIC VISUAL CONCEPTS WITH A CONSTRAINED VARIATIONAL FRAMEWORK**”, ICLR 2017

VAE - βVAE

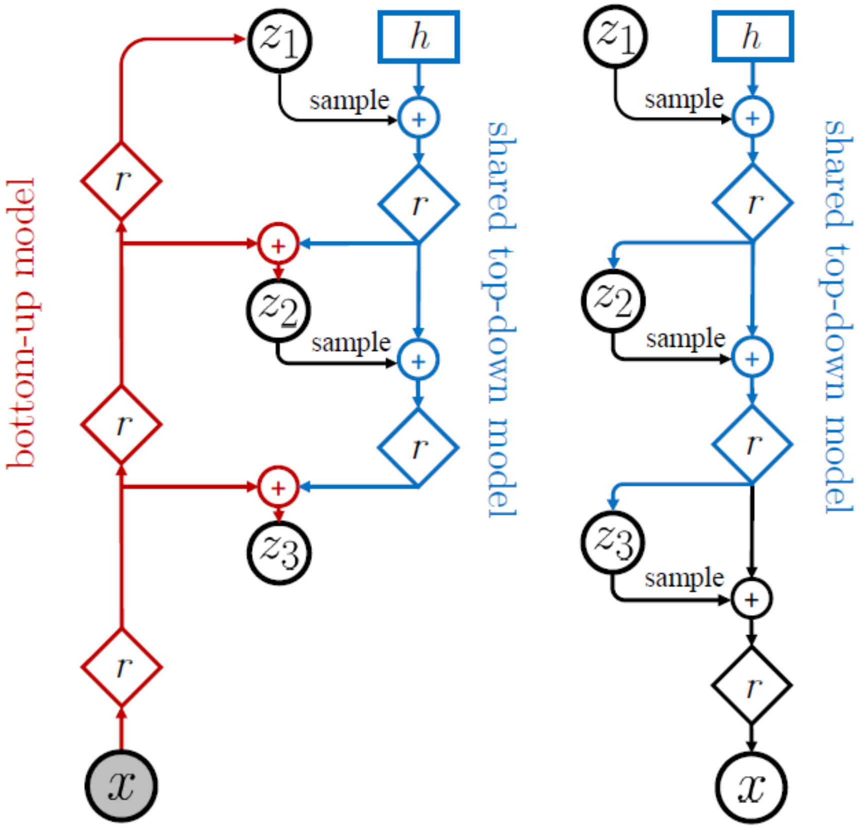


Higgins et al. "**β-VAE: LEARNING BASIC VISUAL CONCEPTS WITH A CONSTRAINED VARIATIONAL FRAMEWORK**", ICLR 2017

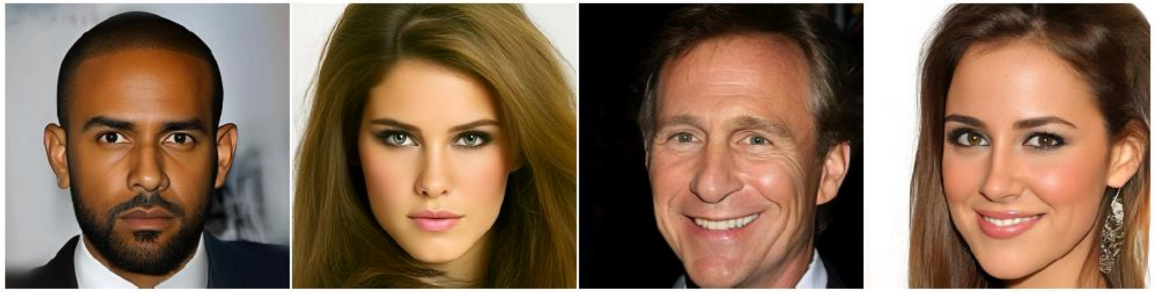
Vanilla VAE – summary

- Probabilistic inference with simple interpretations, and simple and efficient implementation
- (Slightly) Blurry images
- Generally good for latent representation learning (dimensionality reduction)
- Many other notable works
  - Check out [Tschannen, Bachem, Lucic, "Recent Advances in Autoencoder-Based Representation Learning", NIPS 2018]

Hierarchical VAE



(a) Bidirectional Encoder (b) Generative Model



[Vahdat&Kautz “NVAE: A Deep Hierarchical Variational Autoencoder” NeurIPS 2020]

Outline

- Generative Modeling
- Variational Auto Encoders
- Generative Adversarial Training
- Other methods

Outline

- Generative Adversarial Networks
- Issues with GANs
- Some important GAN variants

Outline

- Generative Adversarial Networks
- Issues with GANs
- Some important GAN variants

Remember the **Goals** of generative models

- Generate realistic samples
- Assign likelihood to samples (Density Estimation)
- (Compressed) Representation Learning

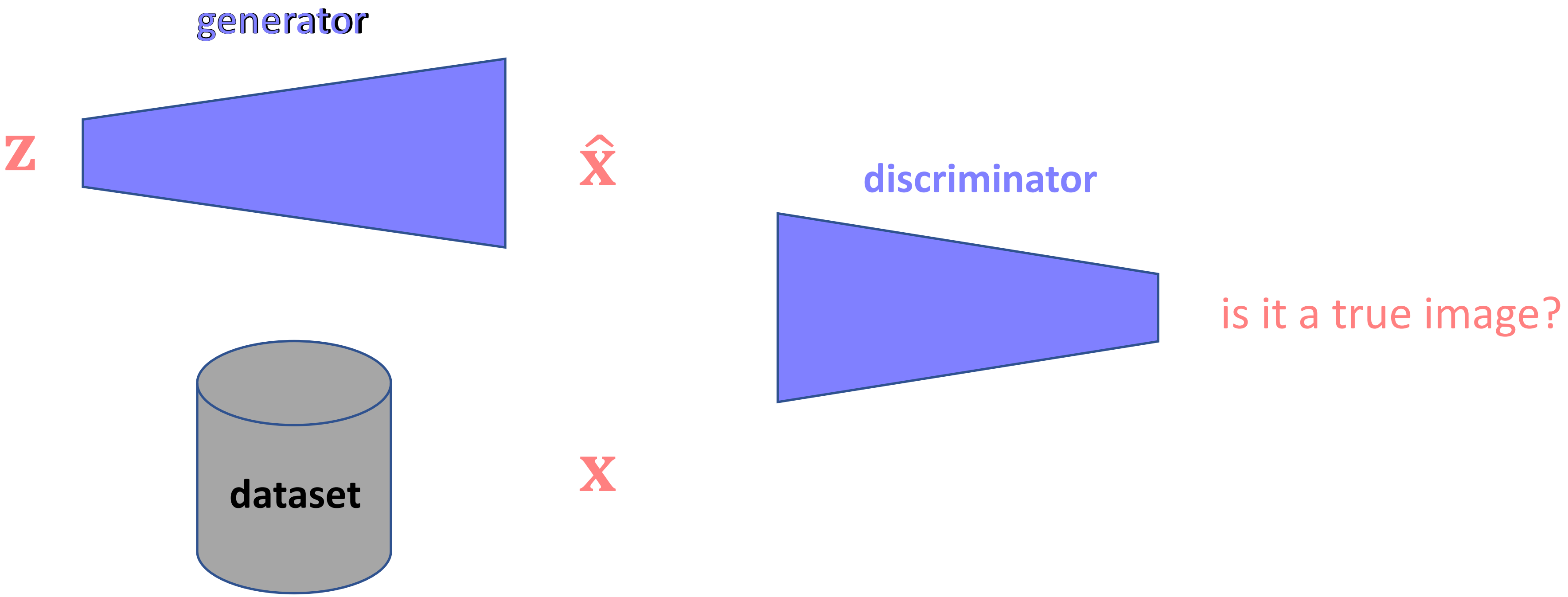
Goals (for GAN as an *implicit generative model*)

- Generate realistic samples
- Assign likelihood to samples (Density Estimation)
- (Compressed) Representation Learning

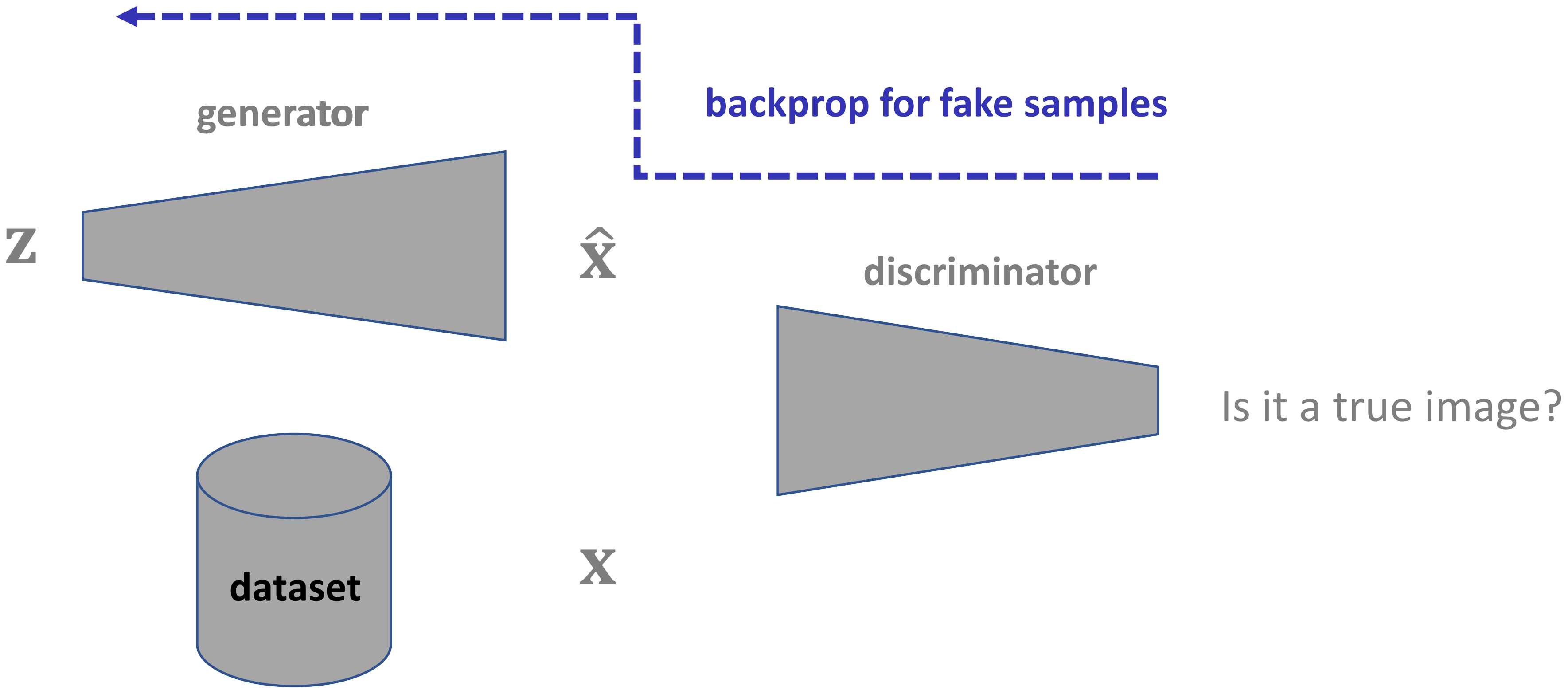
Generative Adversarial Networks

- Generative *Adversarial* Networks (GAN)
- Generative: we want to generate samples
- Networks: we use deep networks for parametrization of our model
- Adversarial: Two adversary networks – one that generates (fake) samples, one that discriminates between fake and true samples

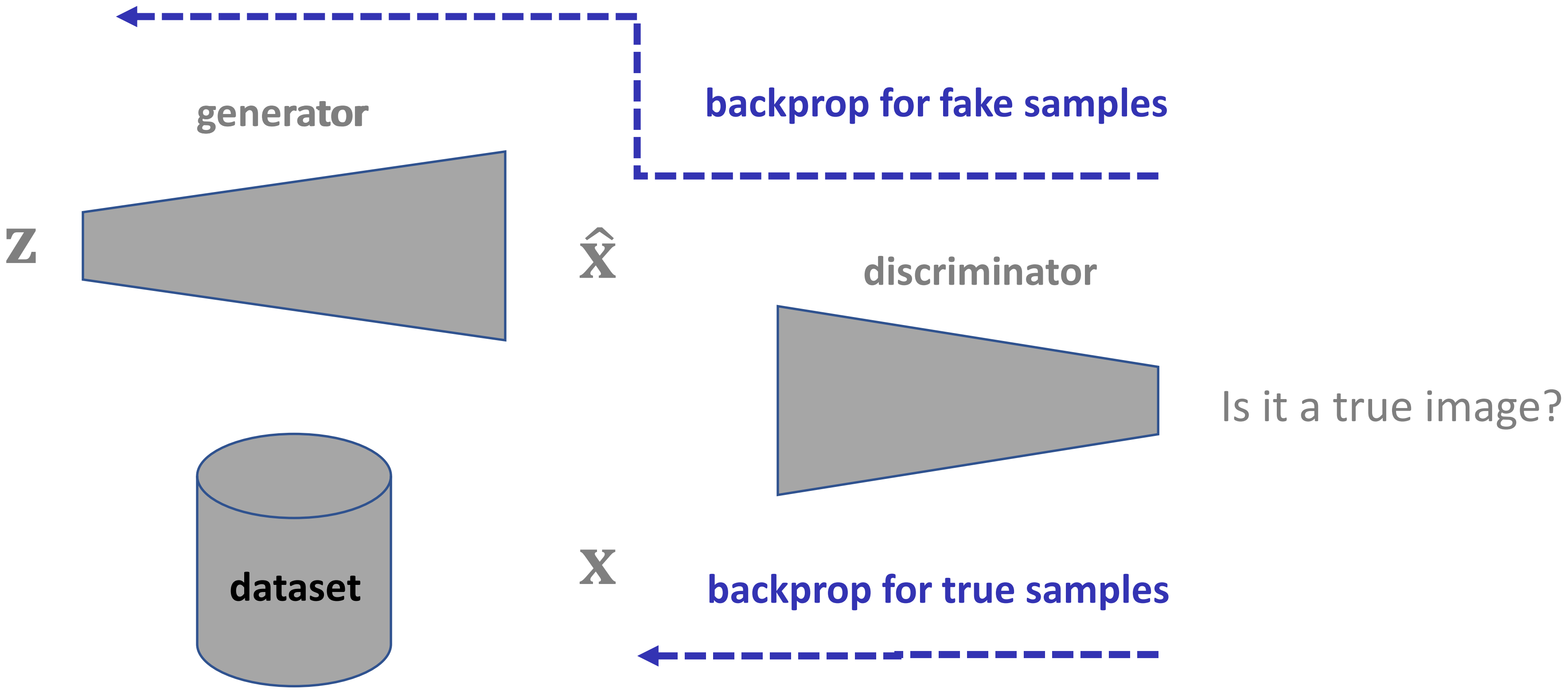
- Adversarial:



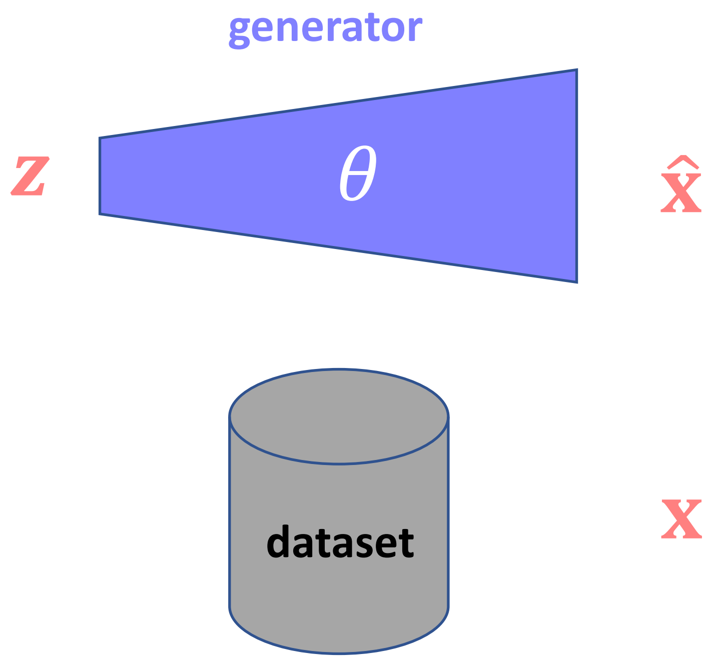
- Key is end-to-end learning (backprop):



- Key is end-to-end learning (backprop):

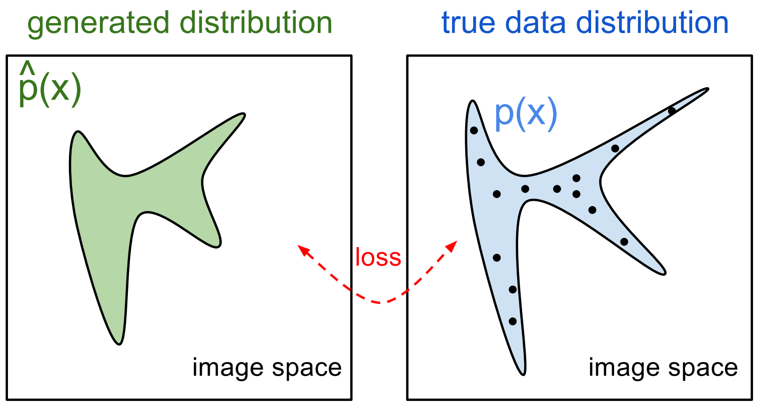


- Loss function



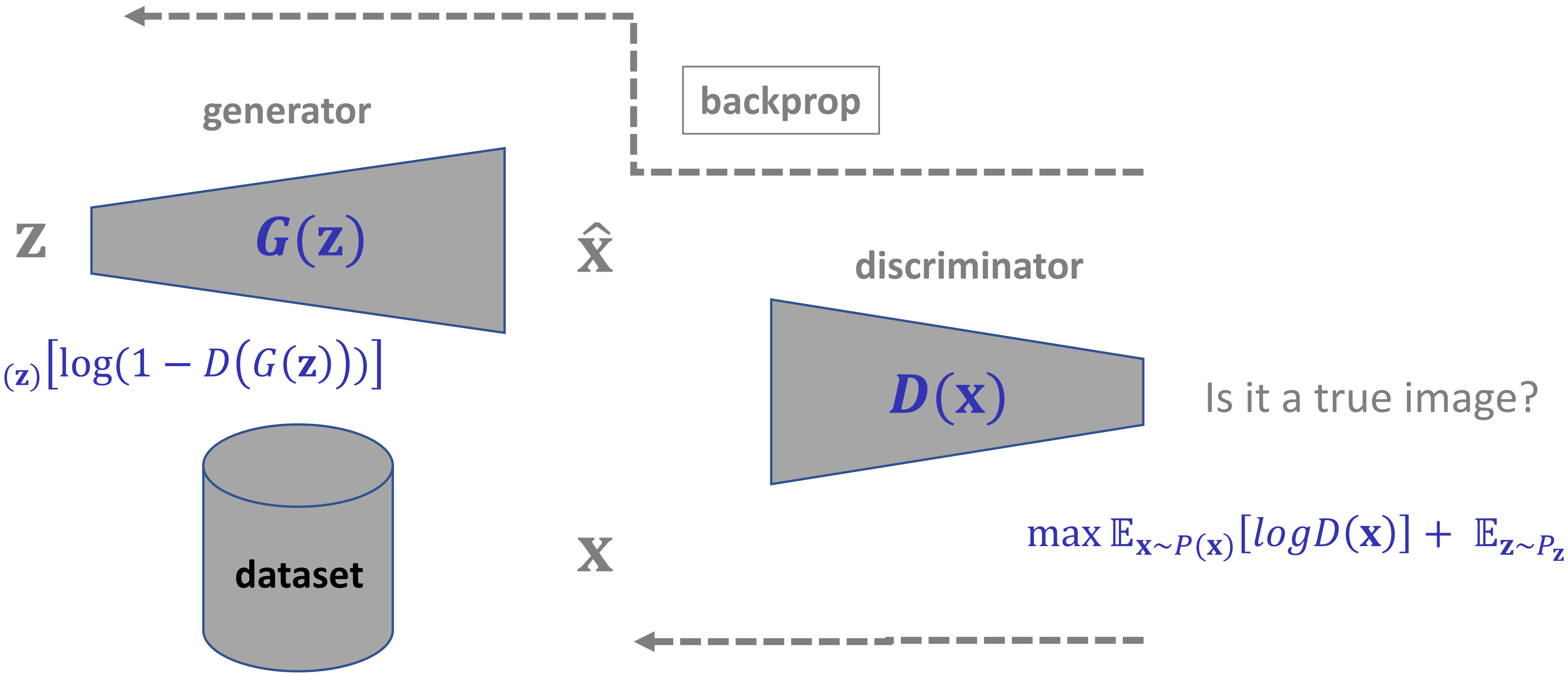
$\hat{P}_\theta(\mathbf{x})$

$P(\mathbf{x})$



[openAI]

Let's see what objective we optimize in GANs

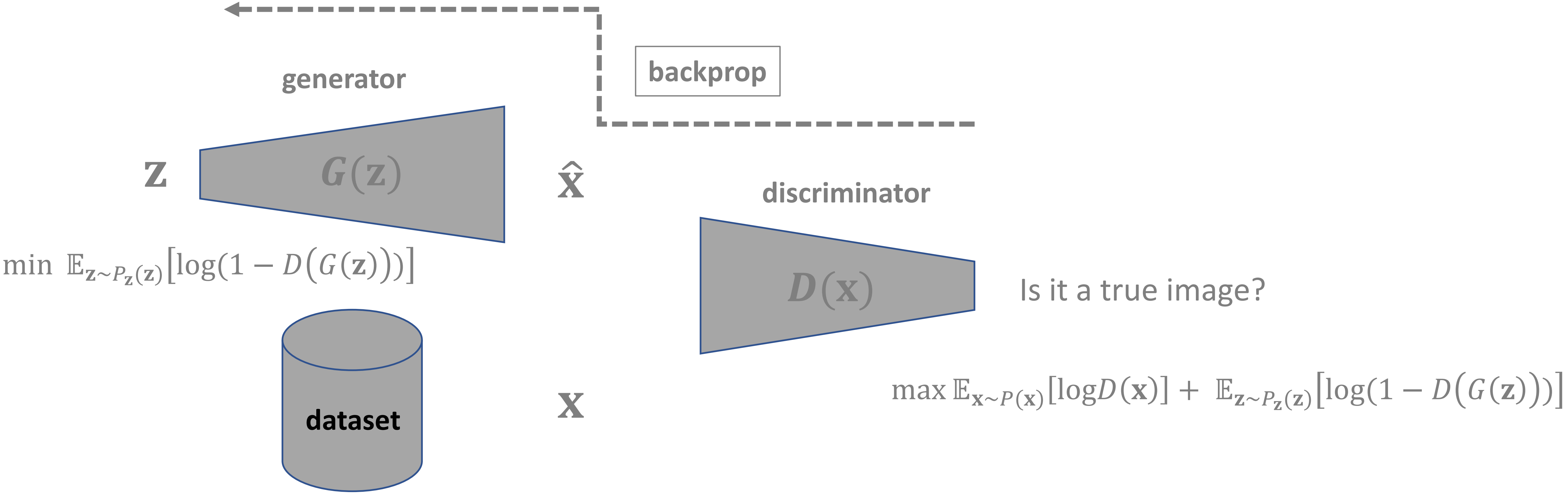


$$\min \ \mathbb{E}_{\mathbf{z}\sim P_{\mathbf{z}}(\mathbf{z})}\left[\log\left(1 - D\left(G(\mathbf{z})\right)\right)\right]$$

$$\max \mathbb{E}_{\mathbf{x}\sim P(\mathbf{x})}[log D(\mathbf{x})] + \ \mathbb{E}_{\mathbf{z}\sim P_{\mathbf{z}}(\mathbf{z})}\left[\log\left(1 - D\left(G(\mathbf{z})\right)\right)\right]$$

Let's see what objective we optimize in GANs

$$\min_{G} \quad \max_{D} \quad \mathbb{E}_{\mathrm{x}\sim P(\mathrm{x})}[\log D(\mathrm{x})] + \mathbb{E}_{\mathrm{z}\sim P_z(\mathrm{z})}[\log(1 - D(G(\mathrm{z})))]$$



Training GAN

- Iterative Alternating Fashion

  - Let both discriminator and generator fiddle against a static version of their adversaries

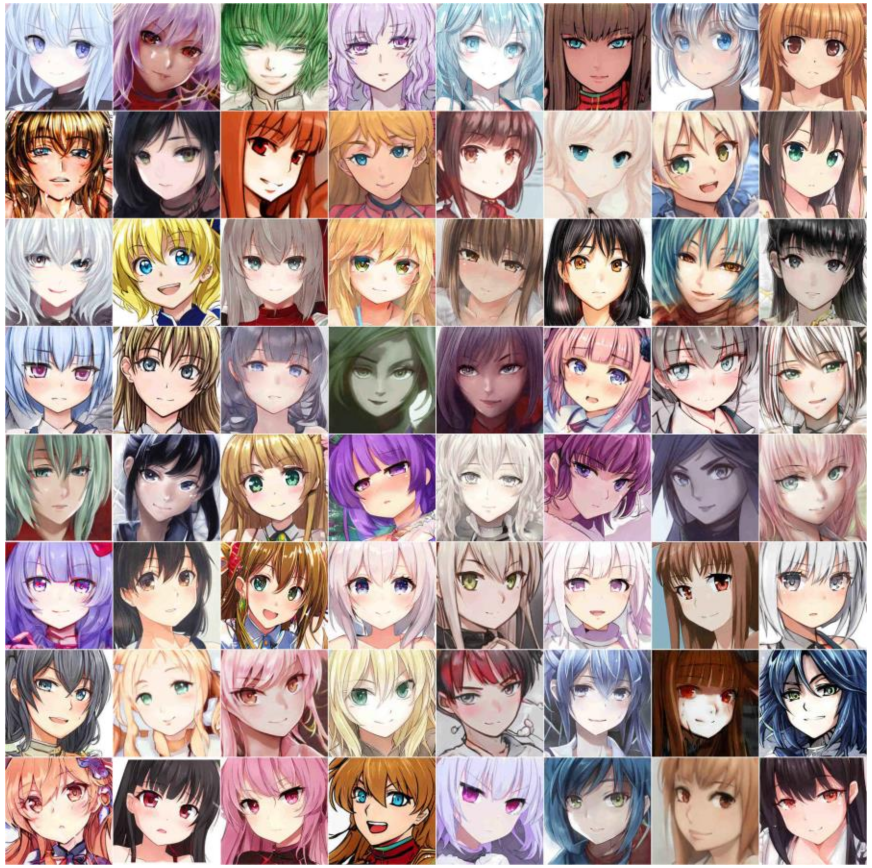
  - For D: use SGD-like algorithm of choice (Adam) on two minibatches simultaneously:
    - A minibatch of training examples
    - A minibatch of generated samples
    - Loss: $-\mathbb{E}_{\mathbf{x}\sim\boldsymbol{P}(\mathbf{x})}[\mathbf{log}\boldsymbol{D}(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim\boldsymbol{P}_{\mathbf{z}}(\mathbf{z})}\left[\mathbf{log}\left(\mathbf{1}-\boldsymbol{D}\left(\boldsymbol{G}(\mathbf{z})\right)\right)\right]$

  - For G: use SGD-like algorithm of choice (Adam) on one minibatch
    - A minibatch of generated samples
    - Loss: $\mathbb{E}_{\mathbf{z}\sim\boldsymbol{P}_{\mathbf{z}}(\mathbf{z})}\left[\mathbf{log}\left(\mathbf{1}-\boldsymbol{D}\left(\boldsymbol{G}(\mathbf{z})\right)\right)\right]$
    - Non-saturating Loss: : $-\mathbb{E}_{\mathbf{z}\sim\boldsymbol{P}_{\mathbf{z}}(\mathbf{z})}\left[\mathbf{log}\left(\boldsymbol{D}\left(\boldsymbol{G}(\mathbf{z})\right)\right)\right]$

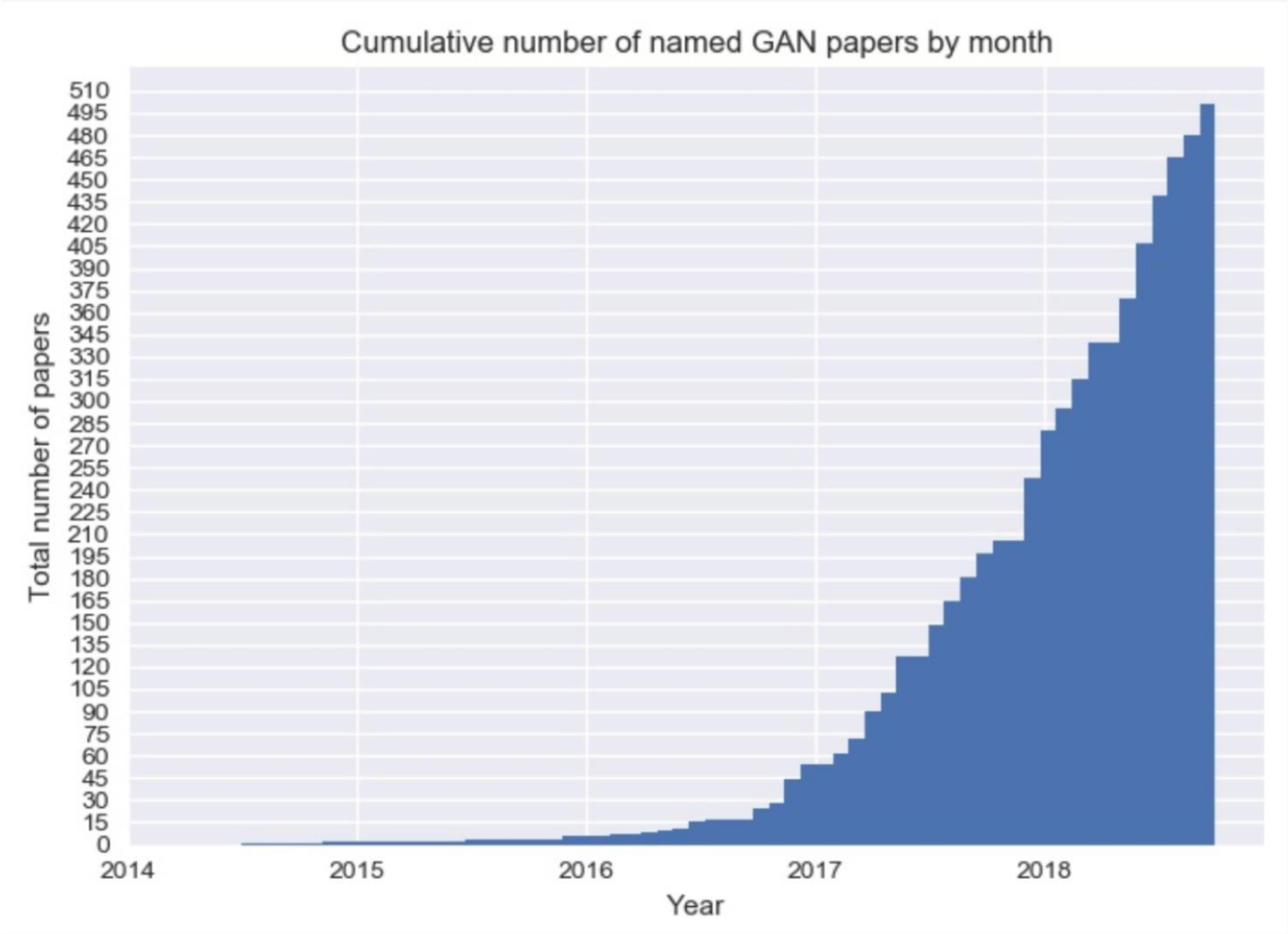
  - Optional: run *k* steps of one player for every step of the other player.

GANs achieve great results!

[StyleGAN v2]



GANs achieve great results



Source: Avinash Hindupur

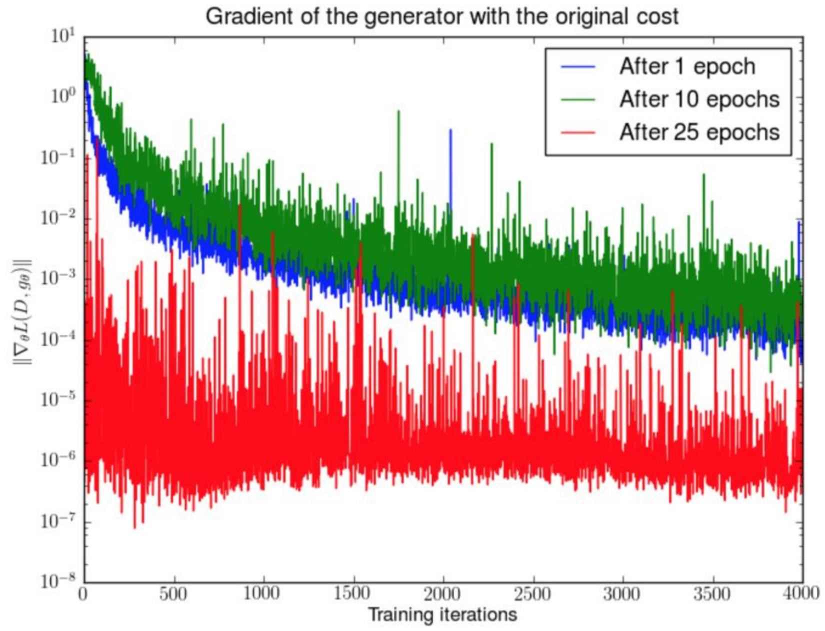
Do GANs have problems though?

Ohhh, YES!

Outline

- Likelihood-free modelling
- Generative Adversarial Networks
- Issues with GANs
- Some important GAN variants

Issue #1: Balance of power



[Arjovsky and Bottou, 2017]

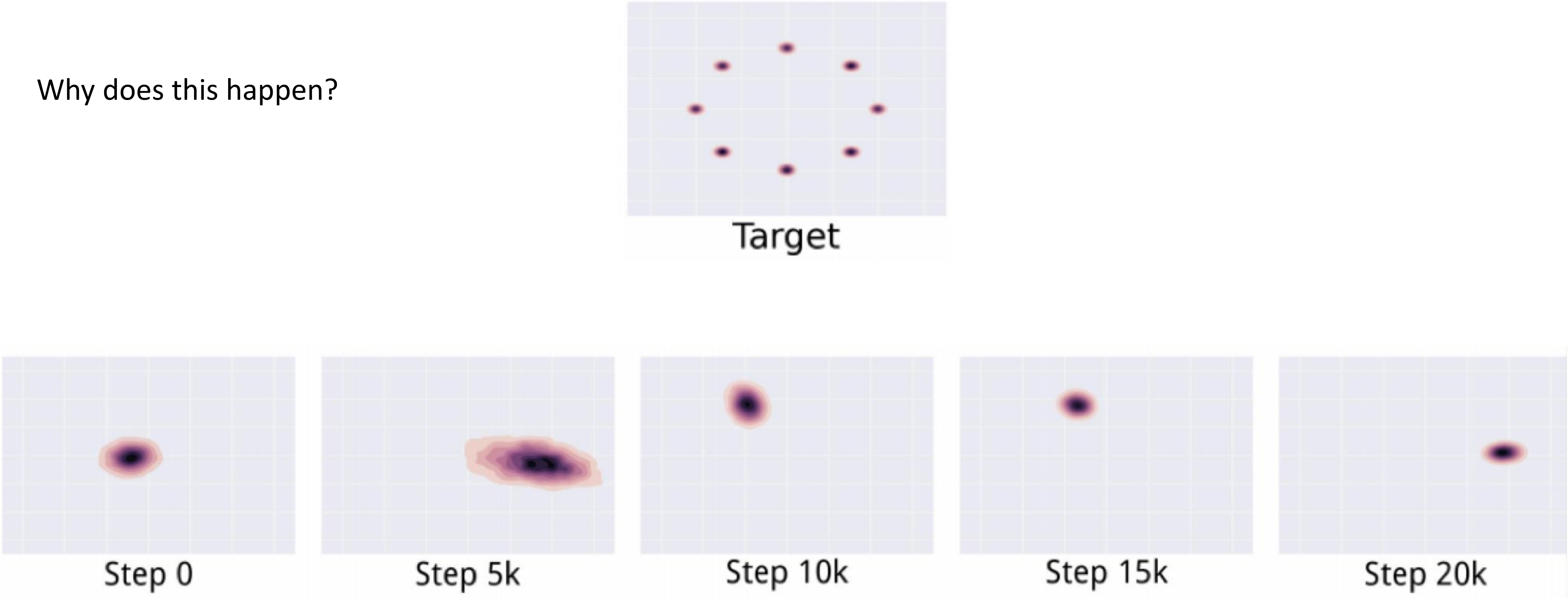
GAN – Mode collapse

Issue #2: Mode collapse

Generator only generates one or few samples/class of samples

GAN – Mode collapse

Why does this happen?

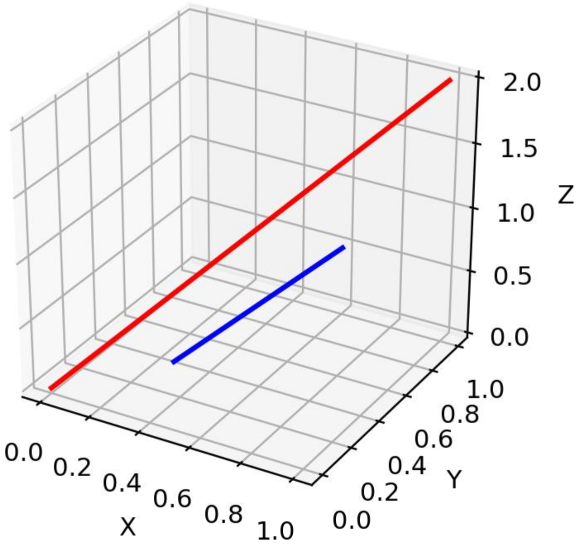


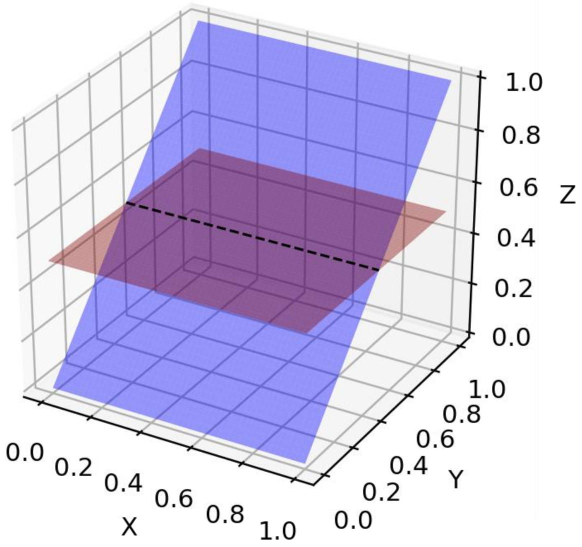
[Metz, Poole, Pfau, Dickstein, “Unrolled Generative Adversarial Networks”, ICLR 2017

GANs: (Almost) Disjoint support

Issue #3: Low-dimensional support of both $P(\mathbf{x})$ and $\hat{P}_\theta(\mathbf{x})$

- D can discriminate all the time!





[Arjovsky&Bottou "TOWARDS PRINCIPLED METHODS FOR TRAINING GENERATIVE ADVERSARIAL NETWORKS"]

GAN: No latent representation

“issue” #4: no latent representation

Vanilla GAN does not have a standard way of steering the generated samples, or producing latent representations for an input x.

Outline

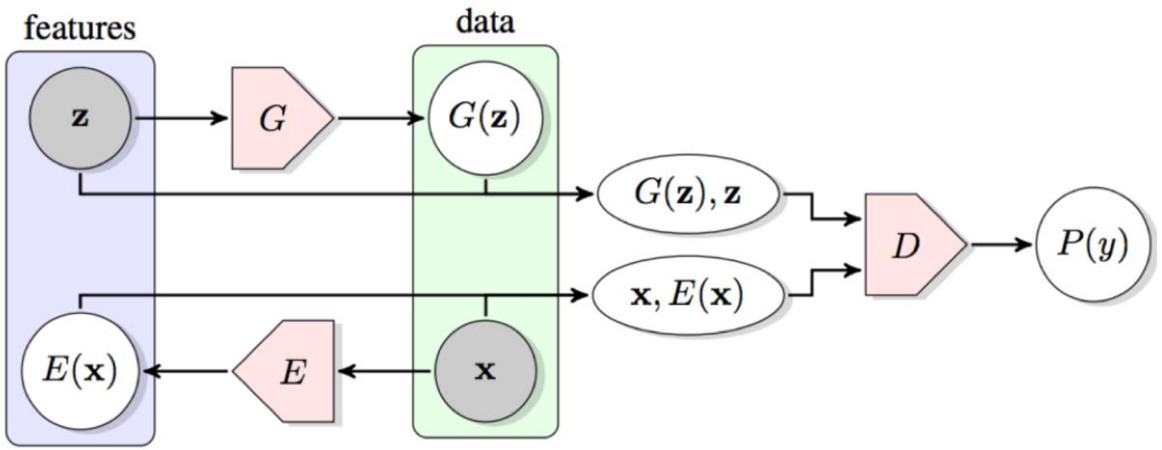
- Likelihood-free modelling
- Generative Adversarial Networks
- Issues with GANs
- Some important GAN variants

BiGAN

- How to encode using GAN?

  - Naïve: One can take the middle representations from the Discriminator

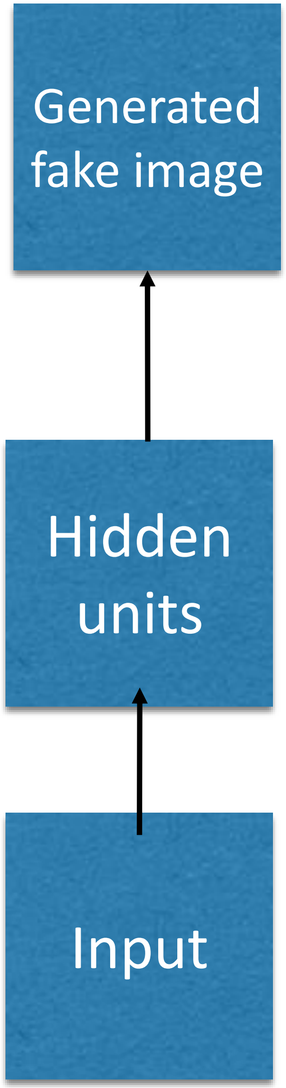
  - BiGAN
    - we have an encoder network $E: \mathbf{x} \rightarrow \mathbf{z}$
    - Discriminator $\big(x, E(x)\big)\ vs\ (G(z), z)$

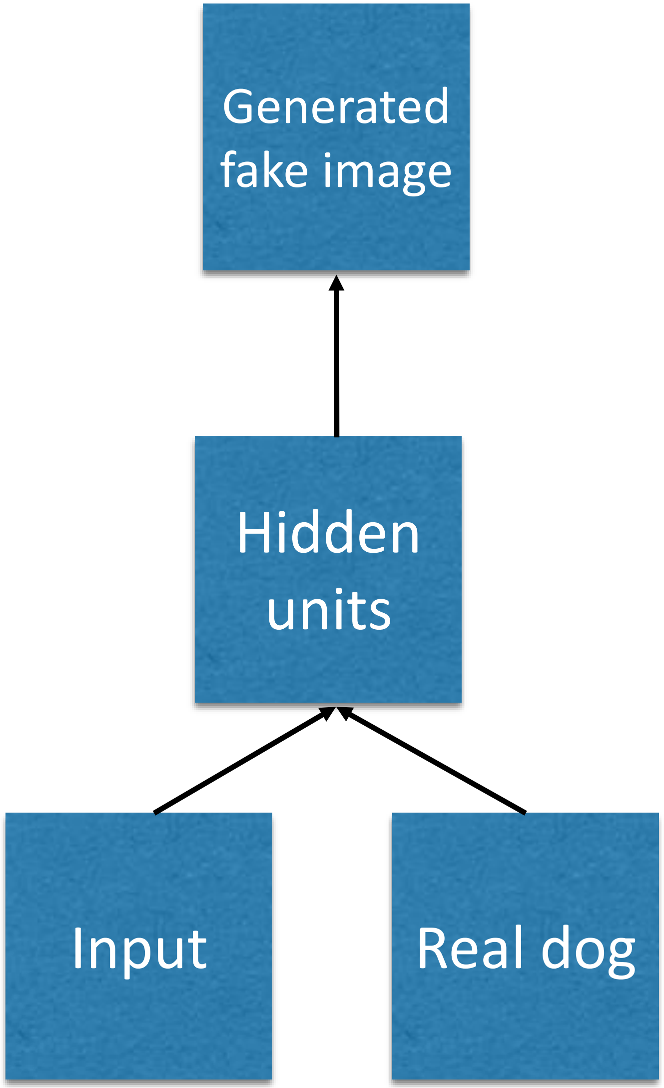


[Jeff Donahue, Philipp Krähenbühl, Trevor Darrell, "Adversarial Feature Learning", ICLR 2017]

But also many other works: InfoGAN, VAEGAN, …

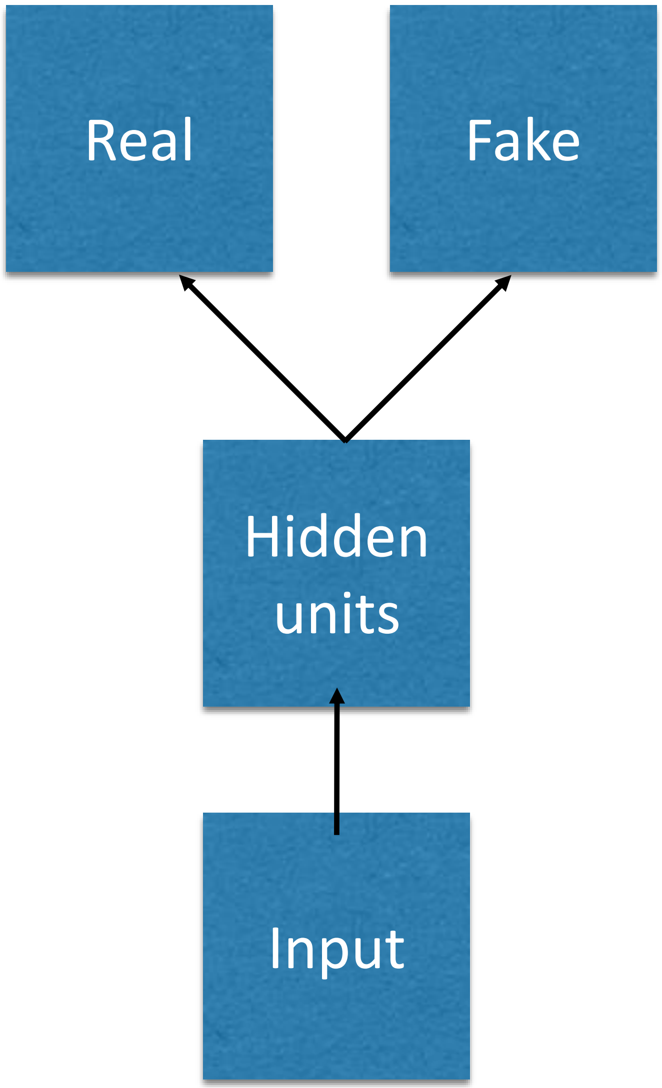
Conditional GAN

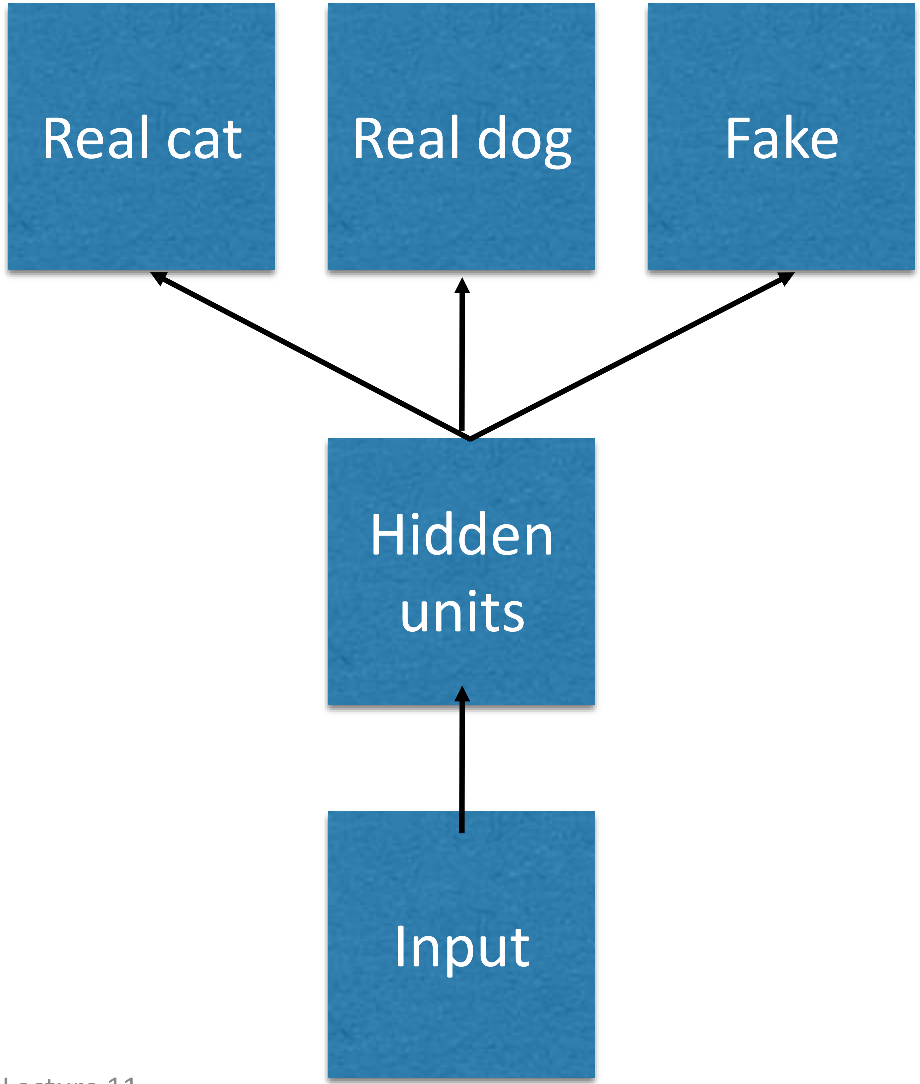




[Odena et al 2016, Salimans et al. 2106)

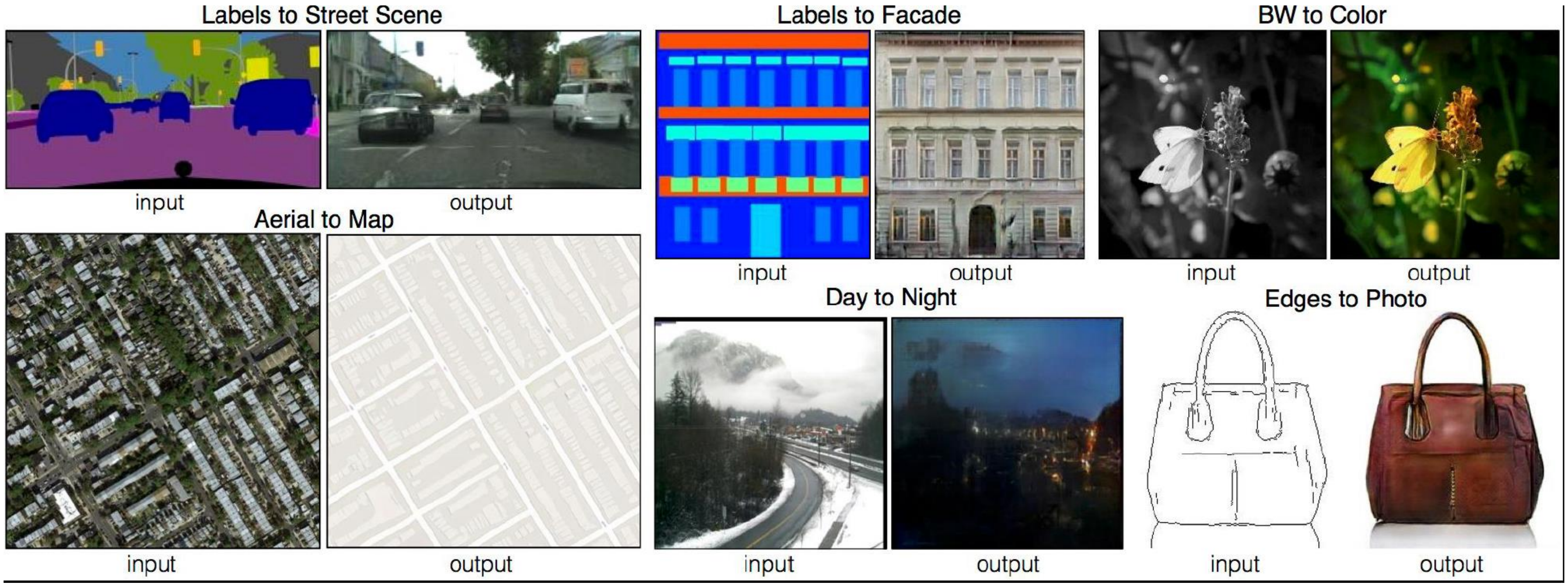
Conditional GAN





[Odena et al 2016, Salimans et al. 2106)

Conditional GAN



[Zhu et al. 2016]

Conditional Generative Models

[Zhu et al, 2017]

Outline

- Generative Modeling
- Variational Auto Encoders
  - AutoEncoders
  - Variational Approximation
  - Examples
- Generative Adversarial Training
- Other methods

Now let's look at a fully-observable case

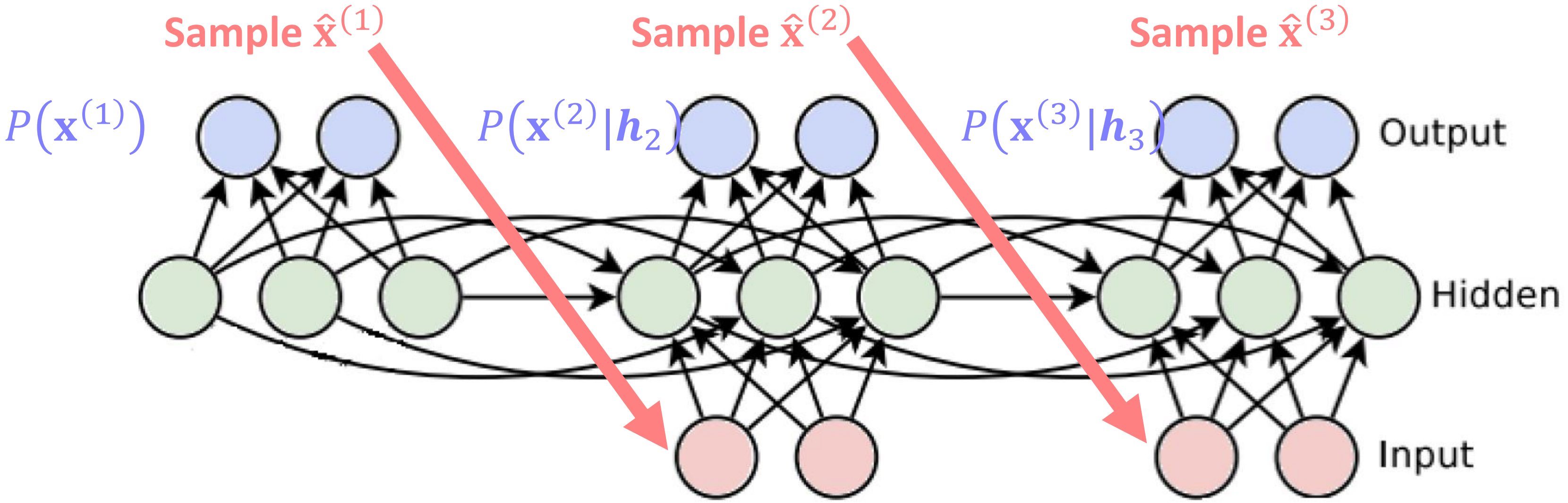
Autoregressive Generative Models

Simple and general idea

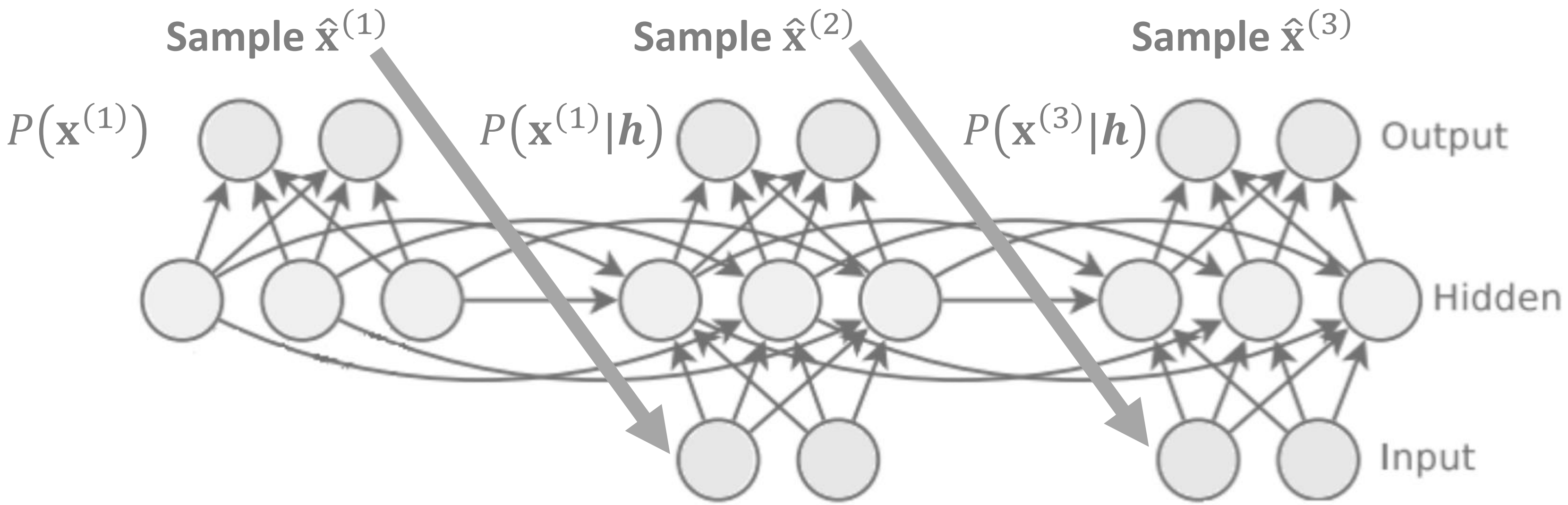
- Take the desired $P(\mathbf{x})$ (with $\mathbf{x}$ a discrete random variable)
- Factorize it using chain rule over dimensions of $\mathbf{x}$
  - $P(\mathbf{x}) = P\big(\mathbf{x}^{(1)}\big)P\big(\mathbf{x}^{(2)}|\mathbf{x}^{(1)}\big)P\big(\mathbf{x}^{(3)}|\mathbf{x}^{(2)},\mathbf{x}^{(1)}\big)\ldots P\big(\mathbf{x}^{(d)}|\mathbf{x}^{(d-1)},\ldots,\mathbf{x}^{(1)}\big)$
- Parametrized and learn each of the conditionals

In general, this still needs exponential number of parameters in d to model $P(\mathbf{x})$!

RNN can be used as an autoregressive generator!



RNN can be used as an autoregressive generator!



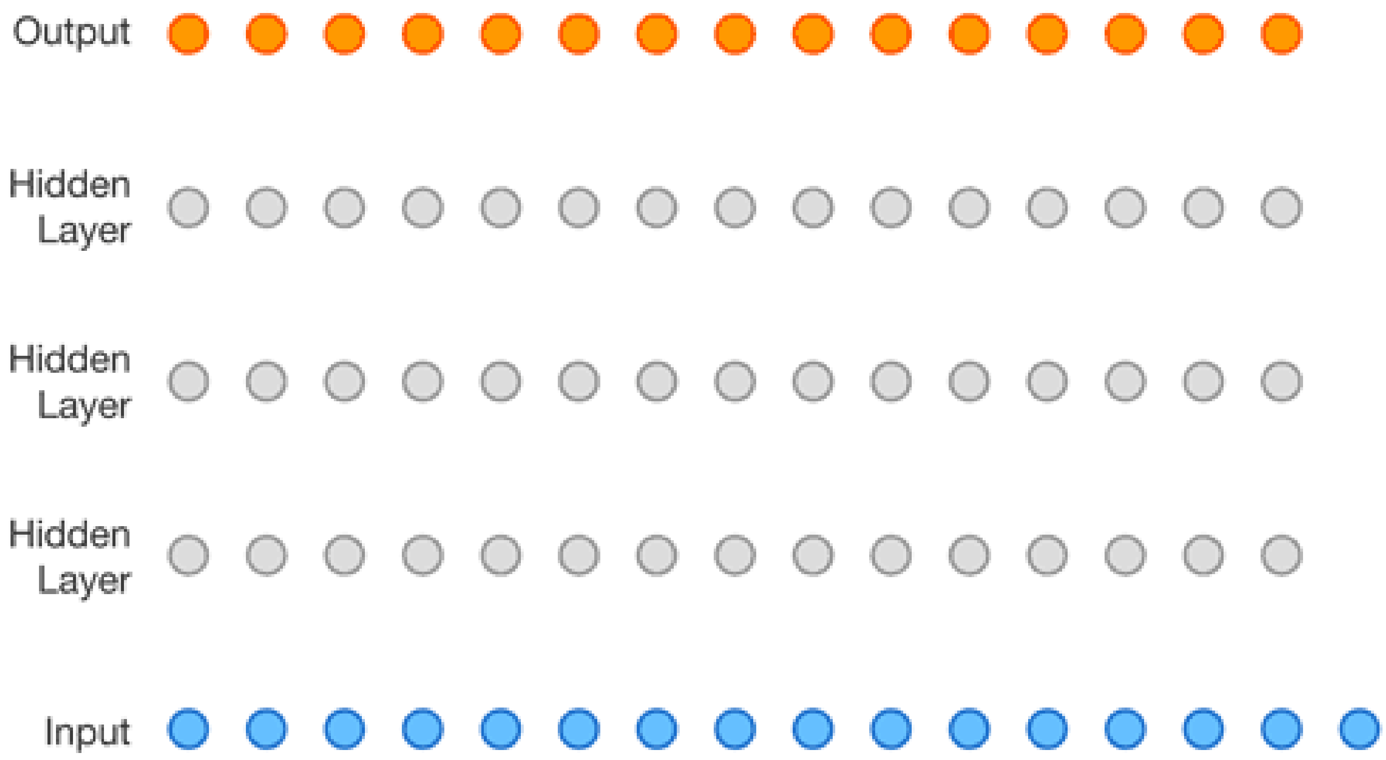
Pros

- ✓ Sequence of arbitrary length
- ✓ Can cast any generation process and increase the modeling capacity as we like

Cons

- ❖ Slow training due to sequential likelihood evaluation
- ❖ Vanishing/exploding gradients (but one can use LSTM/GRU)

Autoregressive deep models



[source: DeepMind: https://deepmind.com/blog/wavenet-generative-model-raw-audio/]

[Oord et al. "WaveNet: A Generative Model for Raw Audio", arXiv 2016]

Autoregressive - Applications

- WaveNet
  - https://deepmind.com/blog/wavenet-generative-model-raw-audio/

ENGLISH

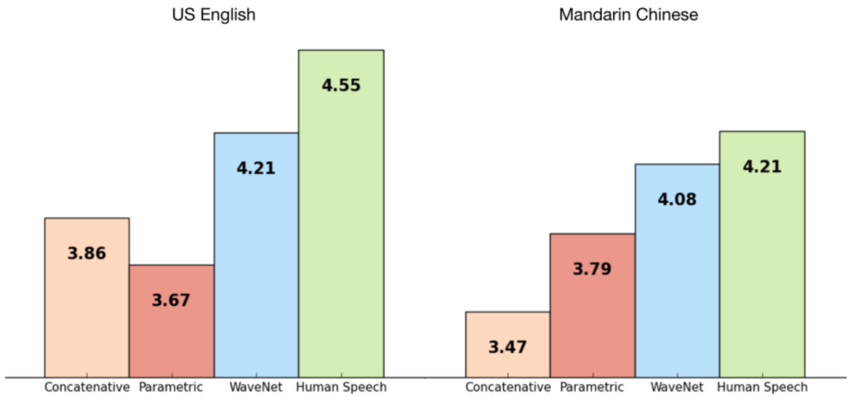
Google HMM Google RNN/LSTM Google WaveNet

MANDARIN

Google HMM Google RNN/LSTM Google WaveNet

MUSIC

Google WaveNet



[Oord et al. “WaveNet: A Generative Model for Raw Audio”, arXiv 2016]

Autoregressive - PixelCNN

Results

- data imputation (ImageNet 32x32)

Summary

- Simple generation process
- Exact and simple density estimation
- Very good for data imputation
- No encoding
- Slow training, sample generation, and density estimation (due to the sequential nature)

There is yet another important class of deep generative models

Normalizing Flow Models

Normalizing Flow

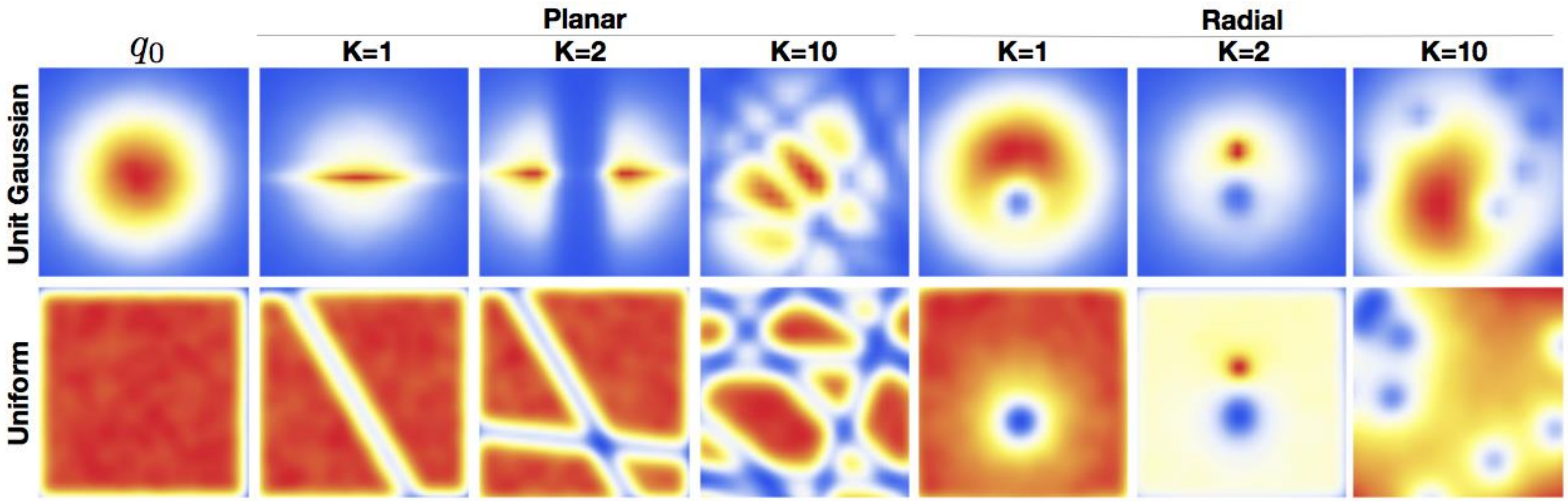
Key idea

- Start from simple distributions that are easy to generate sample from and enable simple density estimation

- Transform the distribution gradually complexify using change of variables.

[Rezende and Mohamed, "Variational Inference with Normalizing Flows", ICML 2016]

Results



[Rezende and Mohamed, "Variational Inference with Normalizing Flows", ICML 2016]

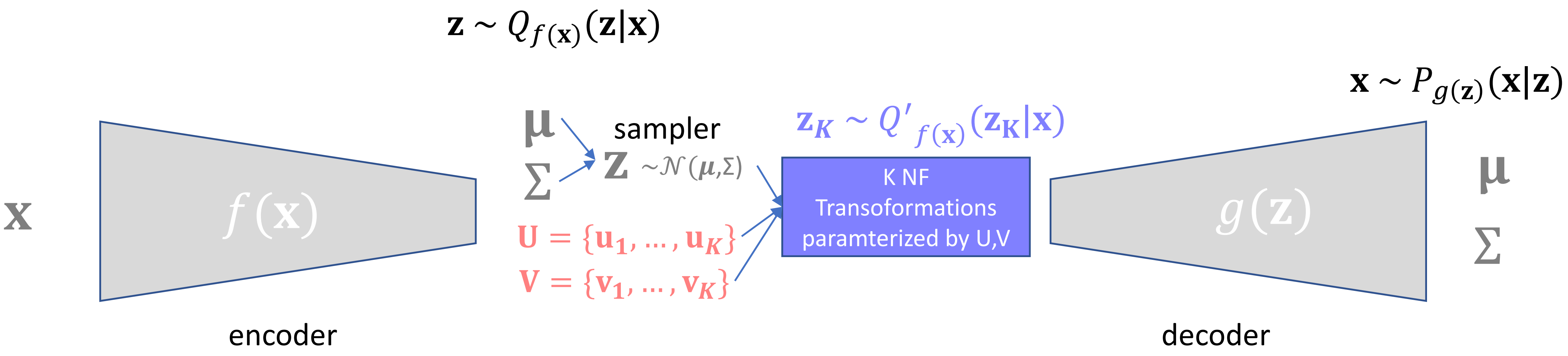
A general use-case

- Approximating posterior distributions are often too simplistic (e.g. Multivariate Gaussian with diagonal covariance)
  - This is usually to make the inference or ML training tractable

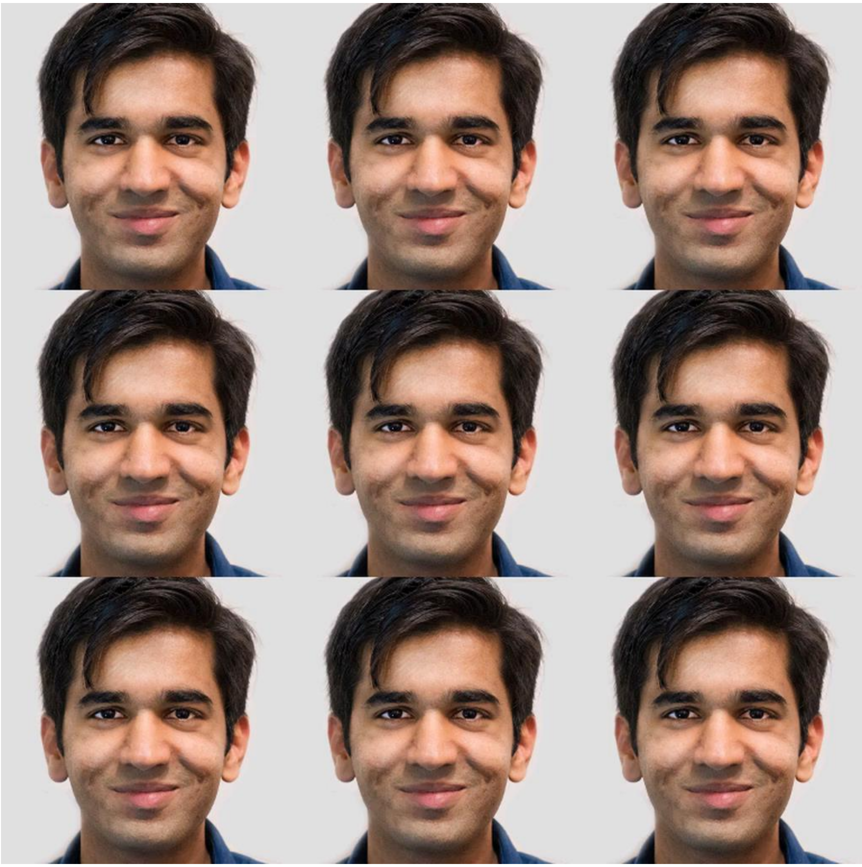
- Normalizing flow is a simple technique to enable modeling more complex approximate distributions

[Rezende and Mohamed, "Variational Inference with Normalizing Flows", ICML 2016]

A general use-case: e.g. VAE



$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}\,\sigma\left(\mathbf{v}^{\mathrm{T}}\mathbf{z} + b\right)$$

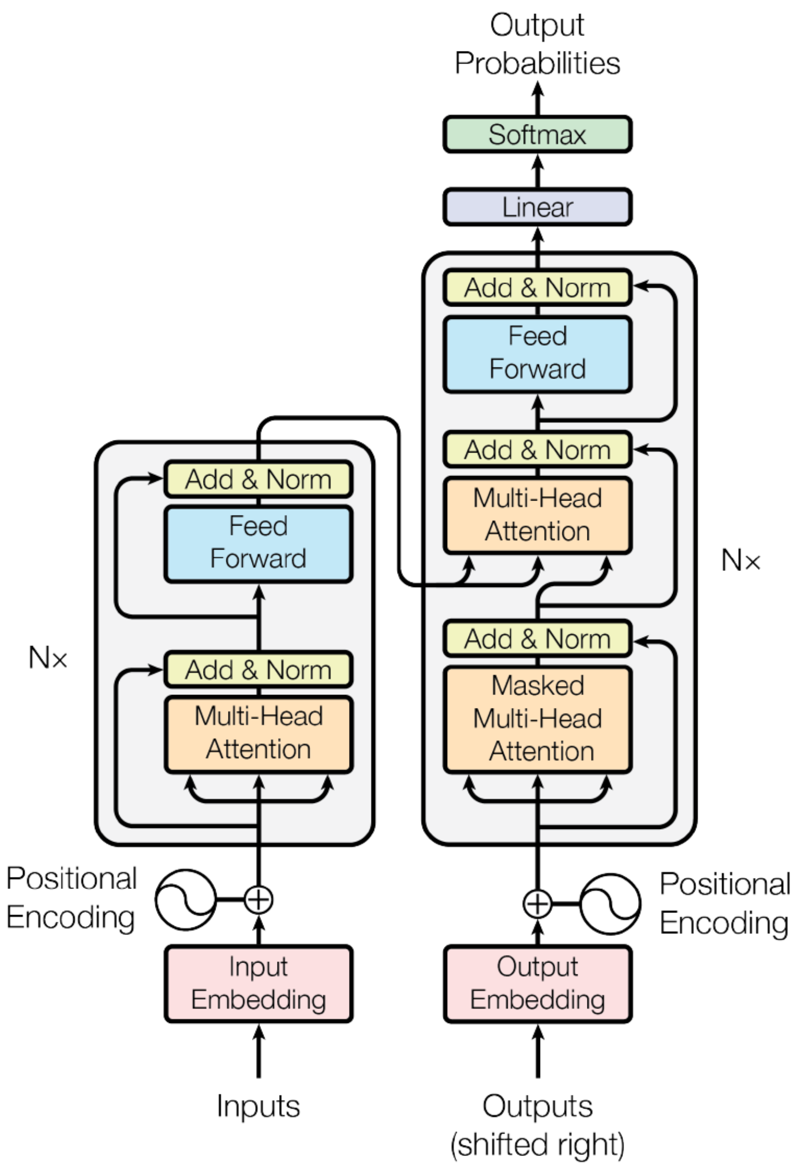


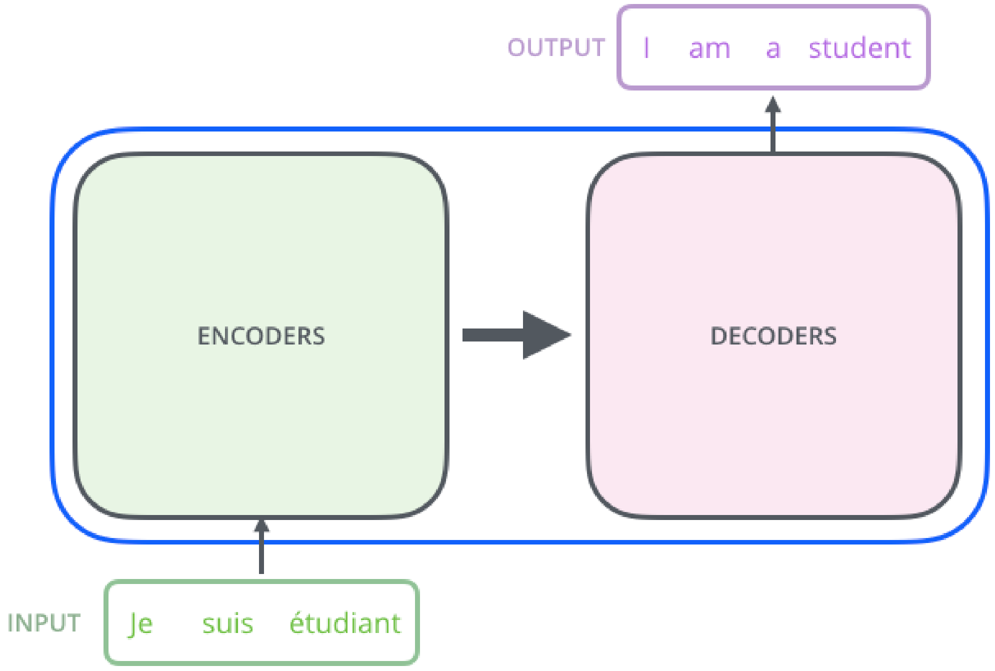
[Kingma, Dhariwal, “Glow: Better Reversible Generative Models”, NIPS 2018]

A recent trend

Cloze-test Image Transformers

Generative Transformers for Images

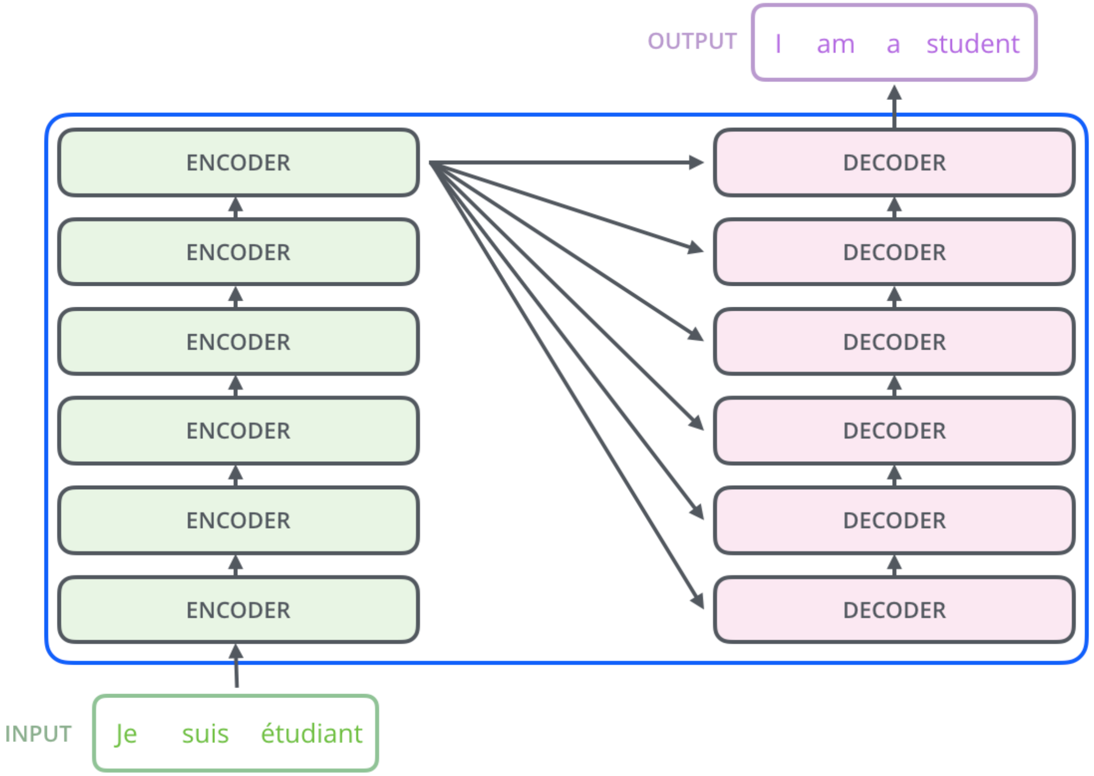


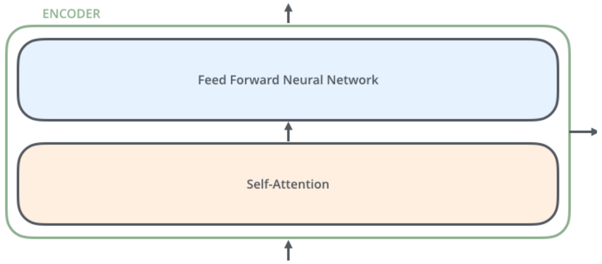


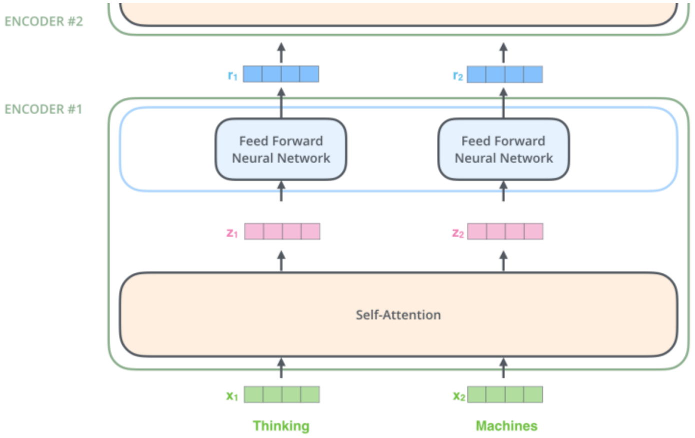
[https://jalammar.github.io/illustrated-transformer/]

Generative Transformers for Images

Encoder Decoder

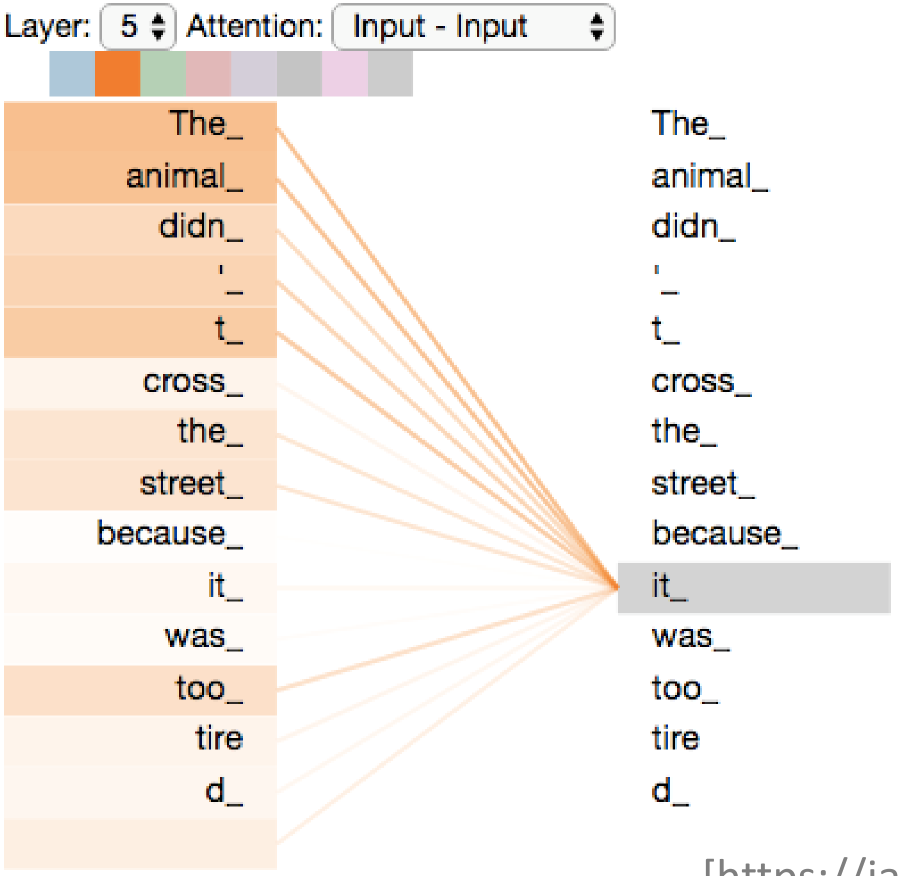






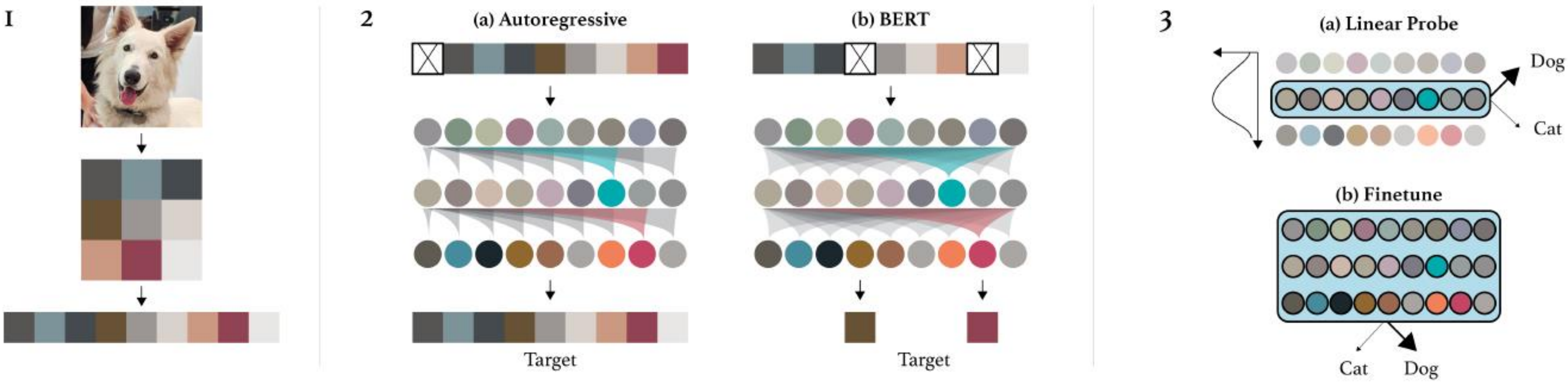
[https://jalammar.github.io/illustrated-transformer/]

Self-Attention



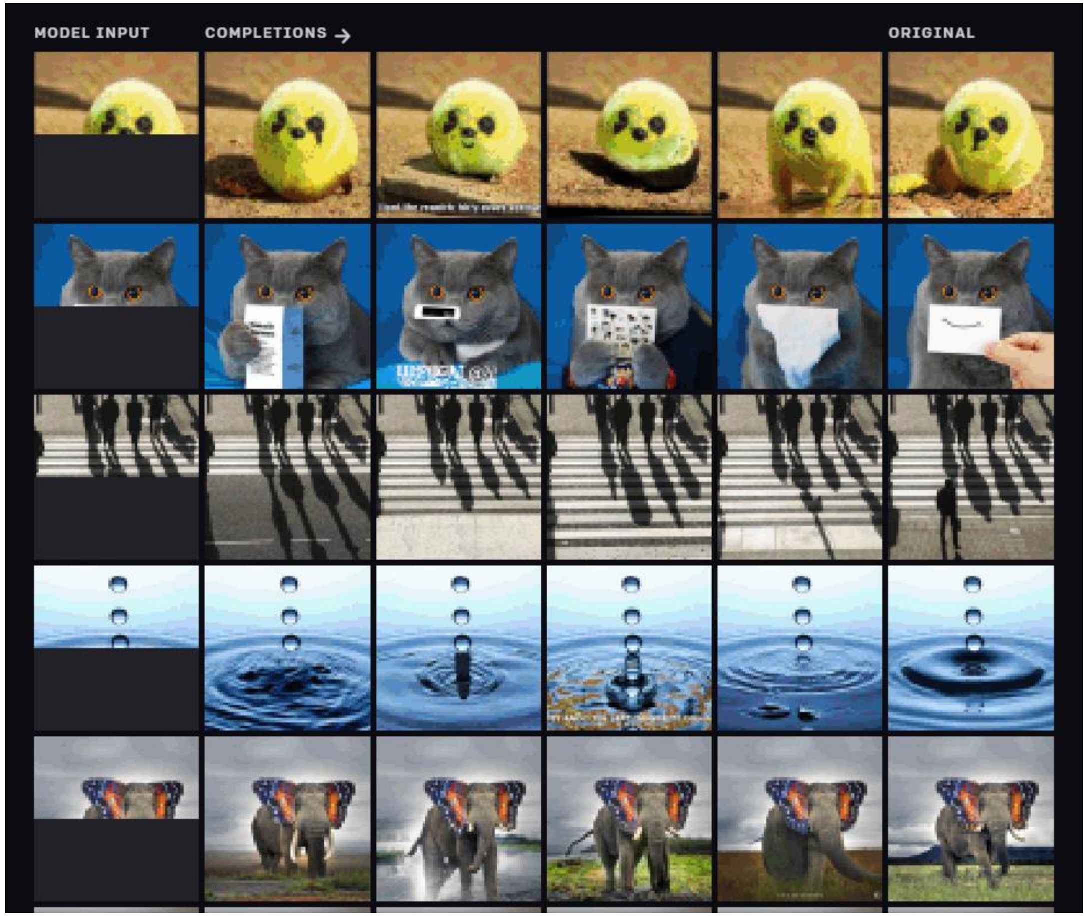
[https://jalammar.github.io/illustrated-transformer/]

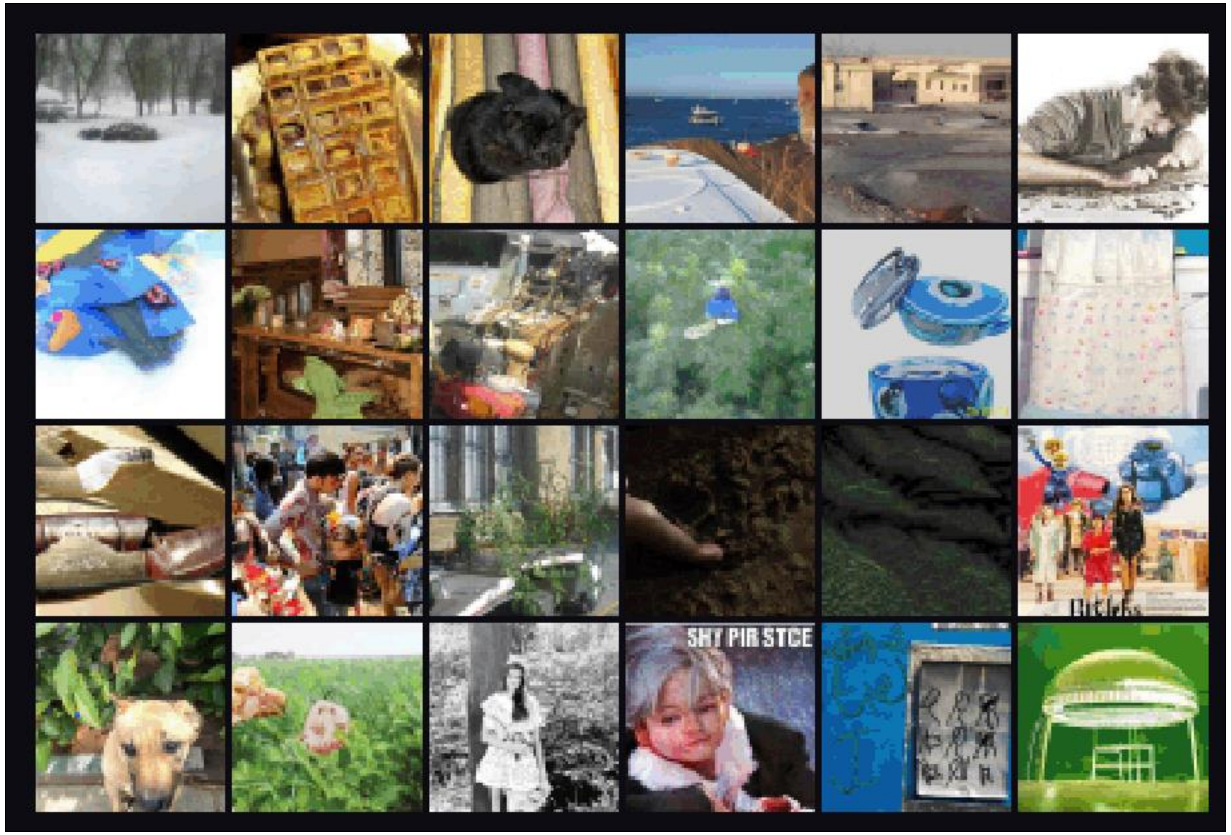
Generative Transformers for Images



[Chen et al., "Generative Pretraining from Pixels", ICML 2020]

Generative Transformers for Images





[https://openai.com/blog/image-gpt/]