Lecture 10 - Unsupervised learning deep generative models

DD2424

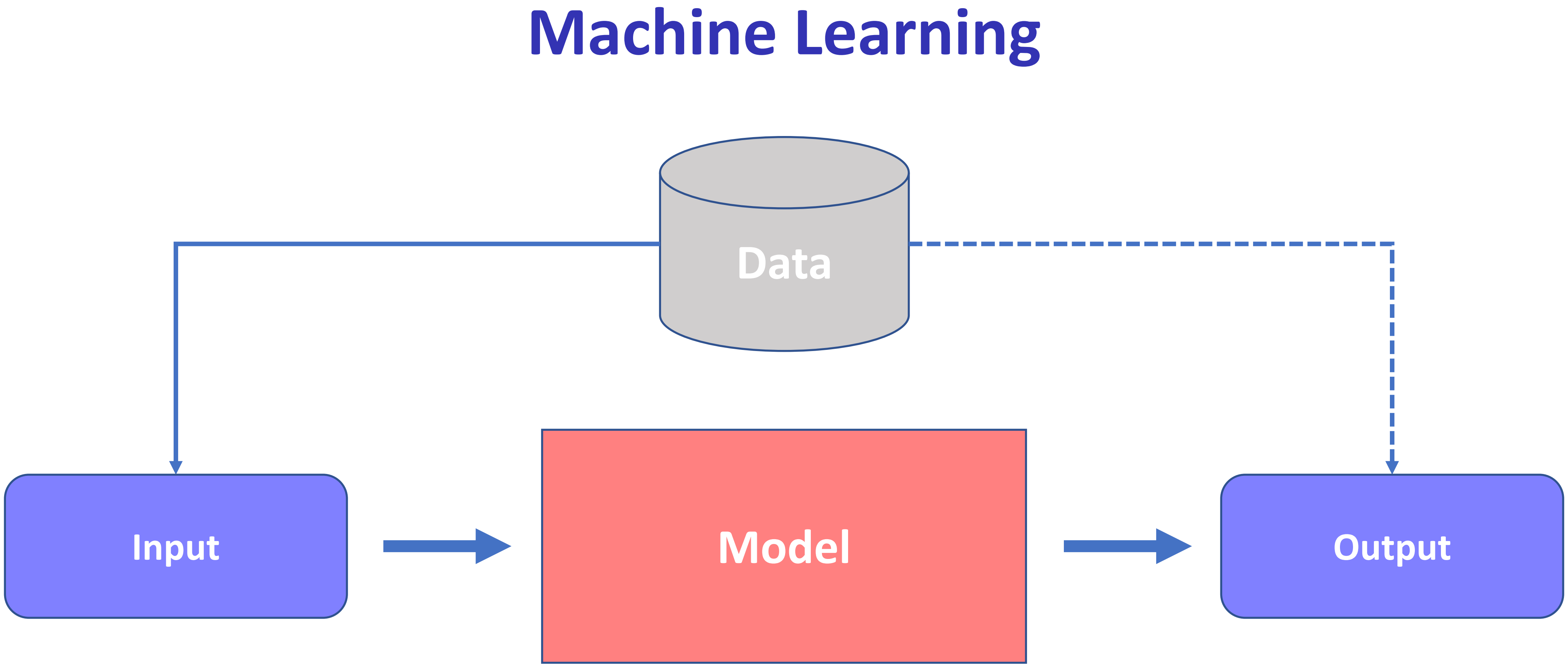
April 25, 2019

Outline

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

Outline

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

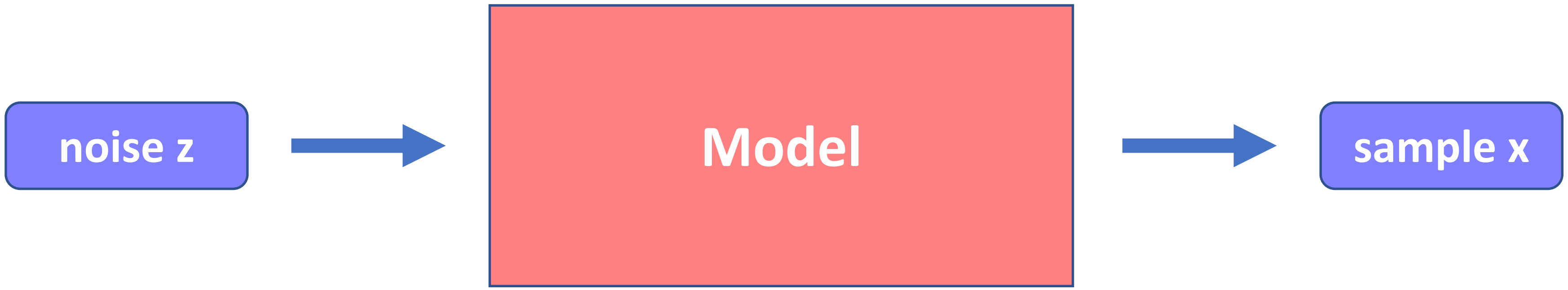

Machine Learning
Data
Input
Model
Output

Models so far

- 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.

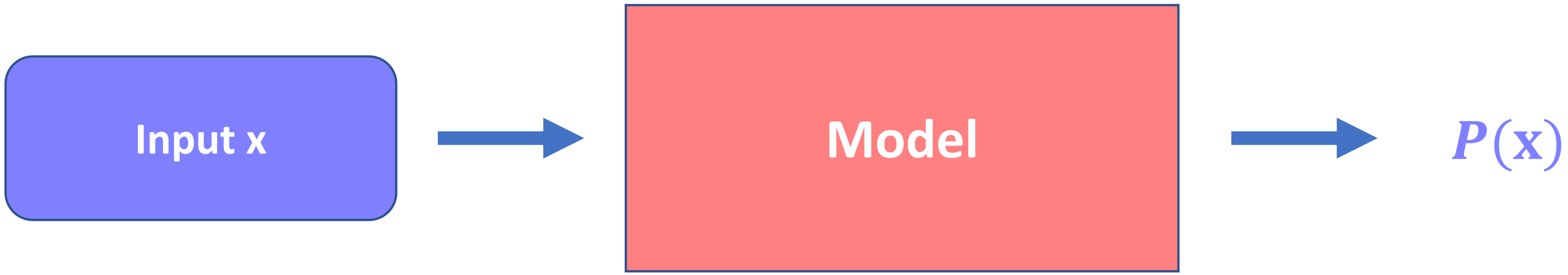
Goals

- Generate realistic samples
- Assign likelihood to samples



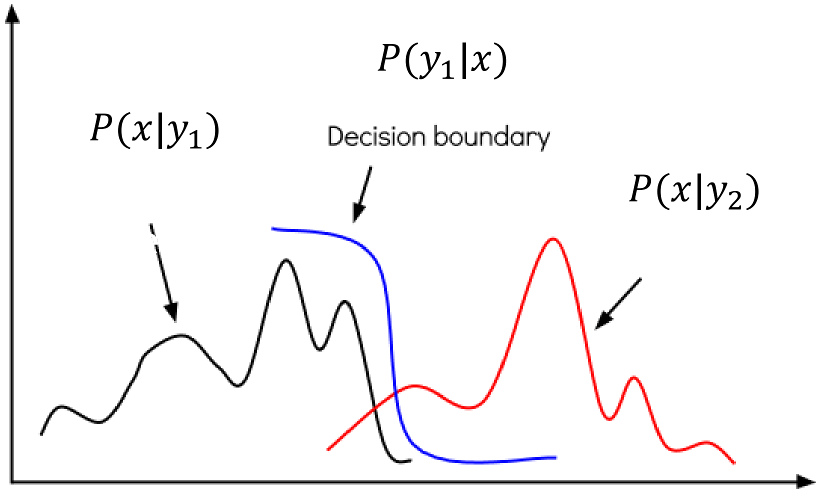
Goals

- Generate realistic samples
- Assign likelihood to samples



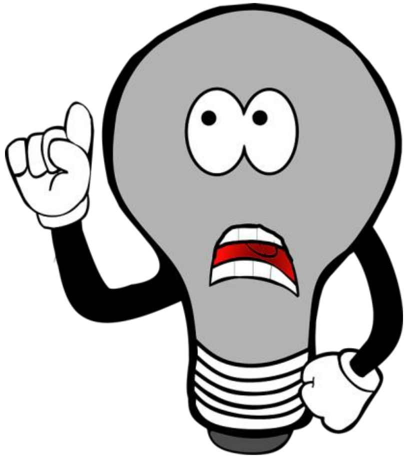
- Machine learning techniques as probabilistic models:

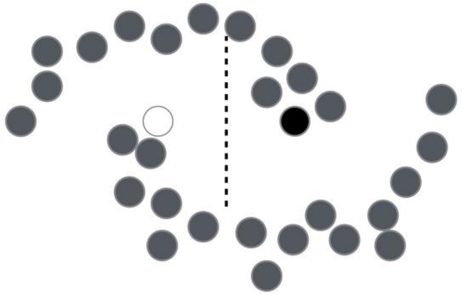
  - Generative Models: $P(\mathbf{x}, y)$ or $P(\mathbf{x})$
  - Discriminative models: $P(y|\mathbf{x})$
    - we get $P(y|\mathbf{x}) = P(\mathbf{x}, y)/P(\mathbf{x})$ in generative models



- Pros:
  - Out of distribution samples
  - Missing data
  - Missing dimensions
  - Semi-supervised learning
  - Synthetic sample generation

- Cons
  - Number of data points
  - Number of assumptions





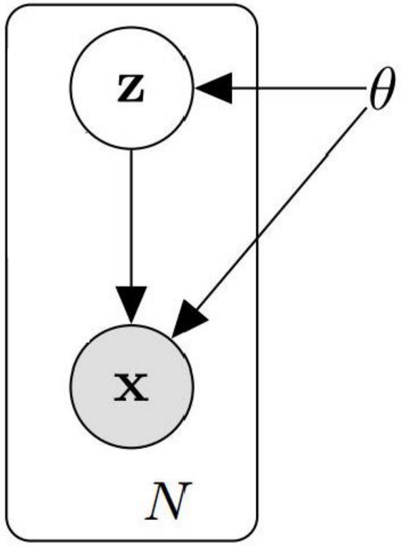
Generative Models

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

z_1: horizontal location
z_2: vertical location
z_3: rotation

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

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



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

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

Why are they called Generative models?

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

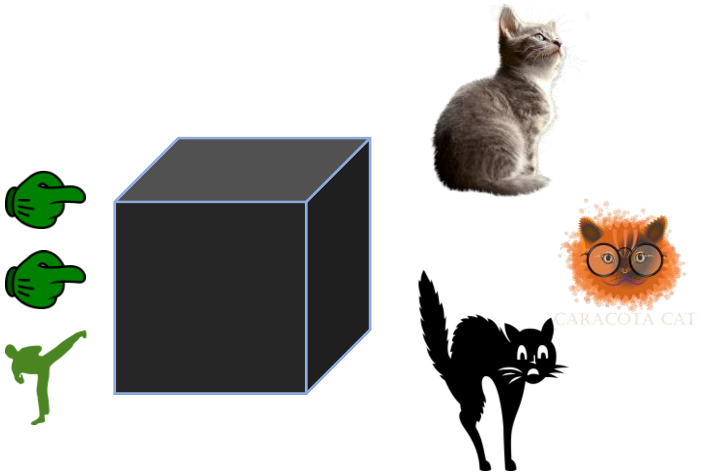
Sample from $P(\mathbf{x}, y)$

- 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?

Generative Models

- Machine learning techniques as applications:
  - Discriminative model: $\underset{y}{\operatorname{argmax}}\ score(\mathbf{x}, y)$
  - Generative model: $\mathrm{g}(\mathbf{z}) \;\rightarrow \mathbf{x} \sim P(\mathbf{x})\ or\ \mathbf{x}, y \sim P(\mathbf{x}, y)$



Deep Generative Models

- Deep networks that can sample $P(\mathbf{x})$ or $P(\mathbf{x}|y)$
- Often through a proxy z of $P(\mathbf{x}|\mathbf{z})$
- We focus on two main families: Variational Auto-Encoders (VAE) and Generative Adversarial Networks (GAN)

Deep Generative Models

we skip the historical methods such as:

- Restricted Boltzmann machines
  - Binary
  - Gaussian-Bernouli
  - DBM
- Deep belief networks

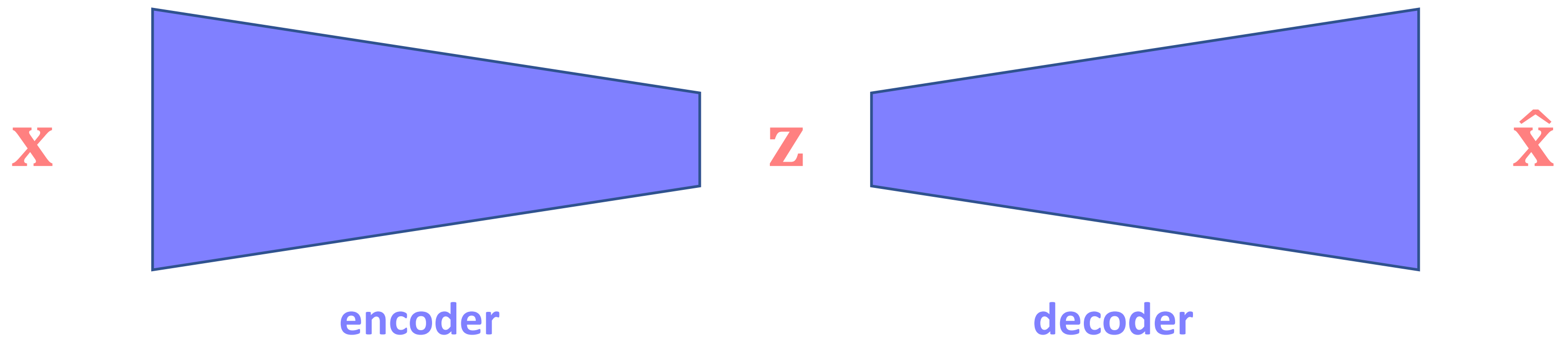
[Ruslan Salakhutdinov, “Learning Deep Generative Models”, Annual Review of Statistics and Its Applications, 2015]

Outline

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

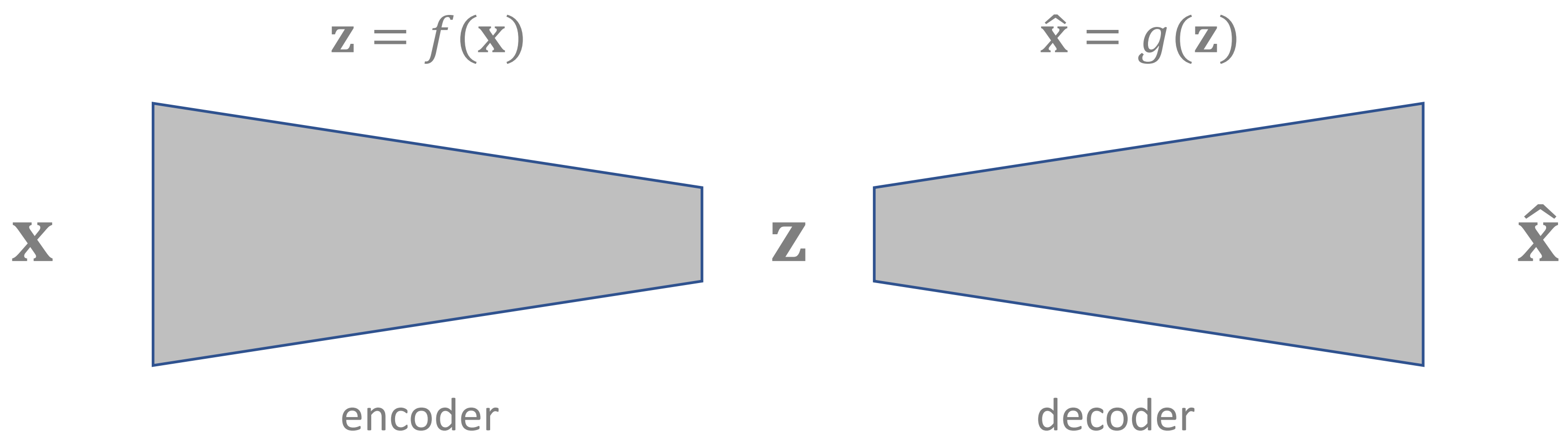
Bottleneck Auto-Encoder Networks

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



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

Bottleneck Auto-Encoder Networks

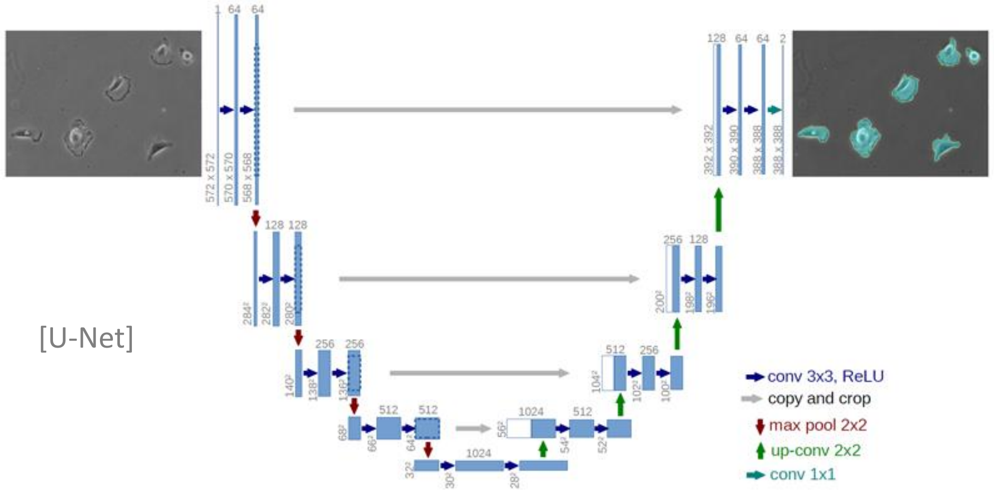


Loss function

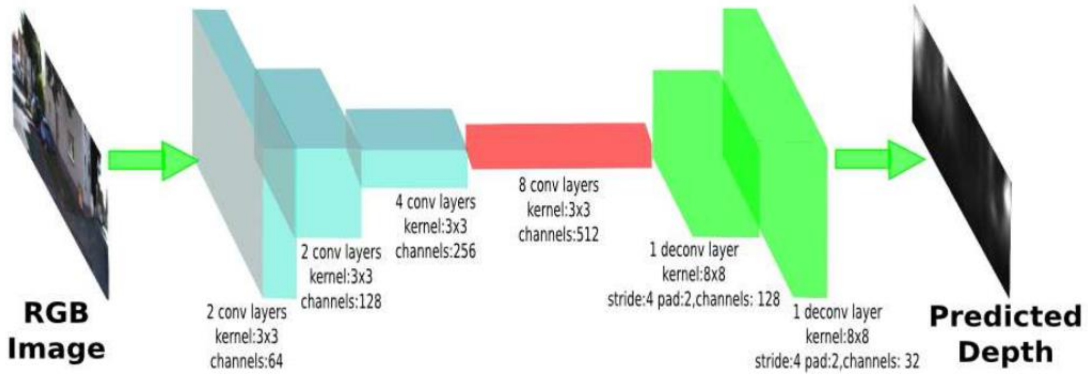
$$L = \sum ||\mathbf{x} - \hat{\mathbf{x}}||^2$$

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

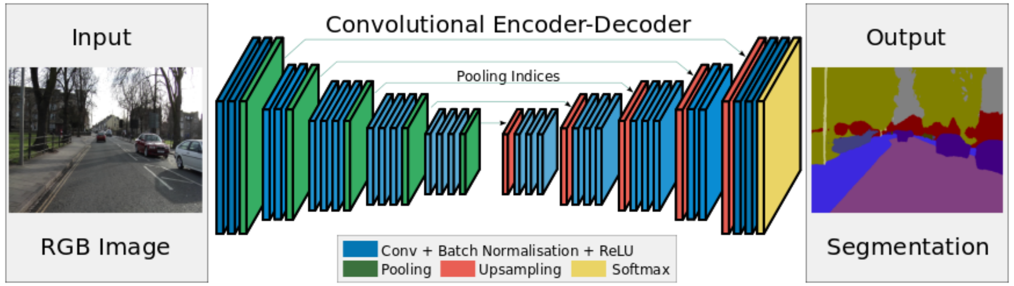
Have you seen that kind of bottleneck networks before?



[U-Net]



[Towards Domain Independence for Learning-Based Monocular Depth Estimation]



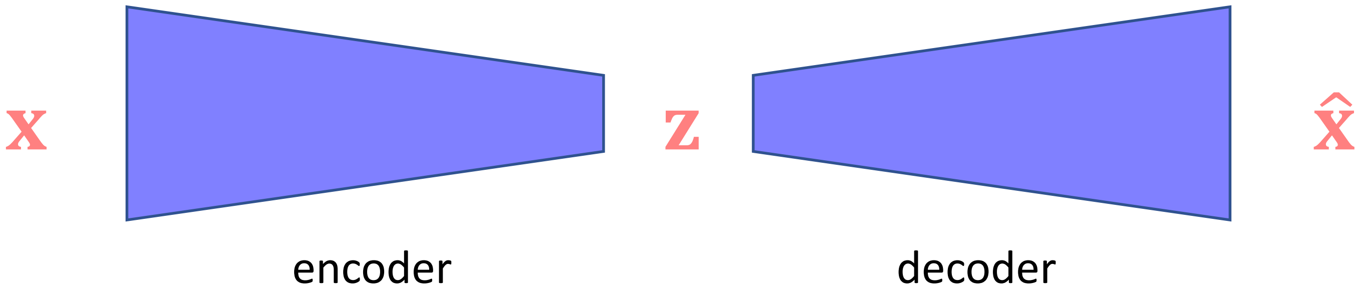
[SegNet]

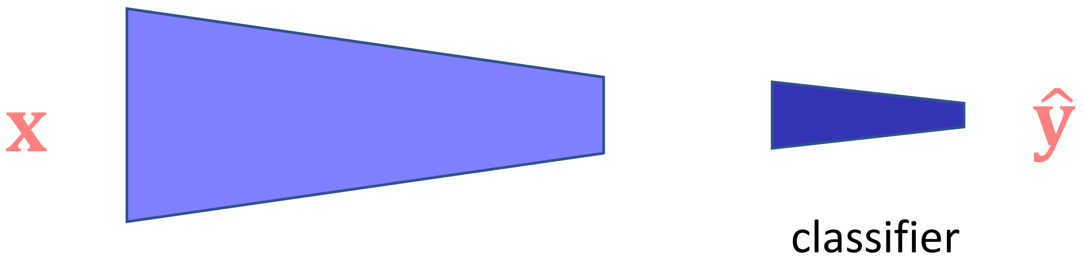
What can AEs be used for?

- Dimensionality Reduction
- Pretraining
- Denoising AutoEncoder
- Image Inpainting

AutoEncoder

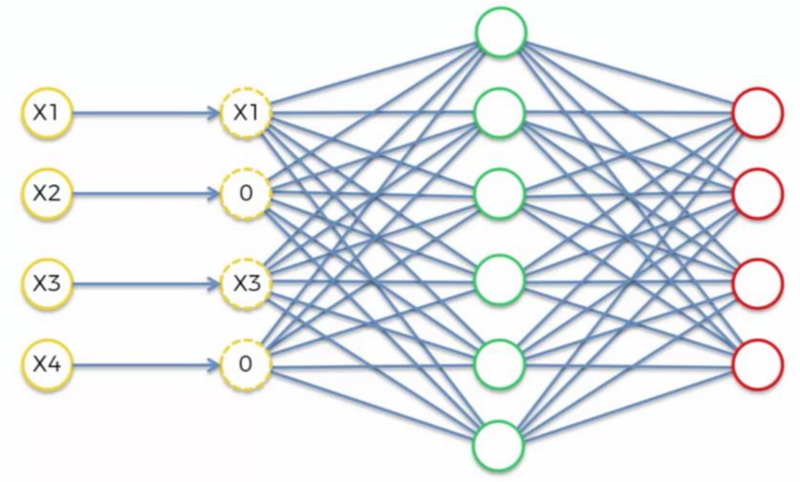
- Dimensionality Reduction
- Pretraining
- Denoising AutoEncoder
- Image Inpainting



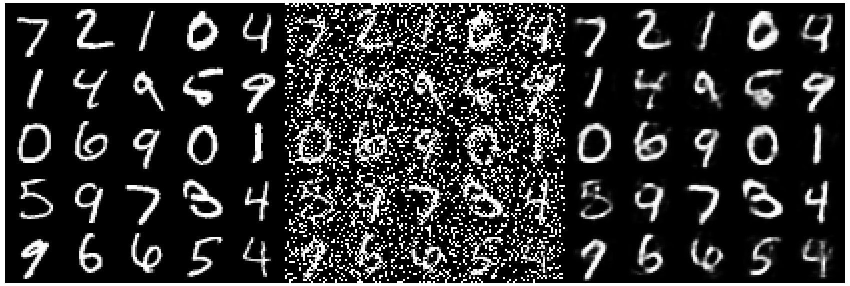


AutoEncoder

- Dimensionality Reduction
- Pretraining
- Denoising AutoEncoder
- Image Inpainting

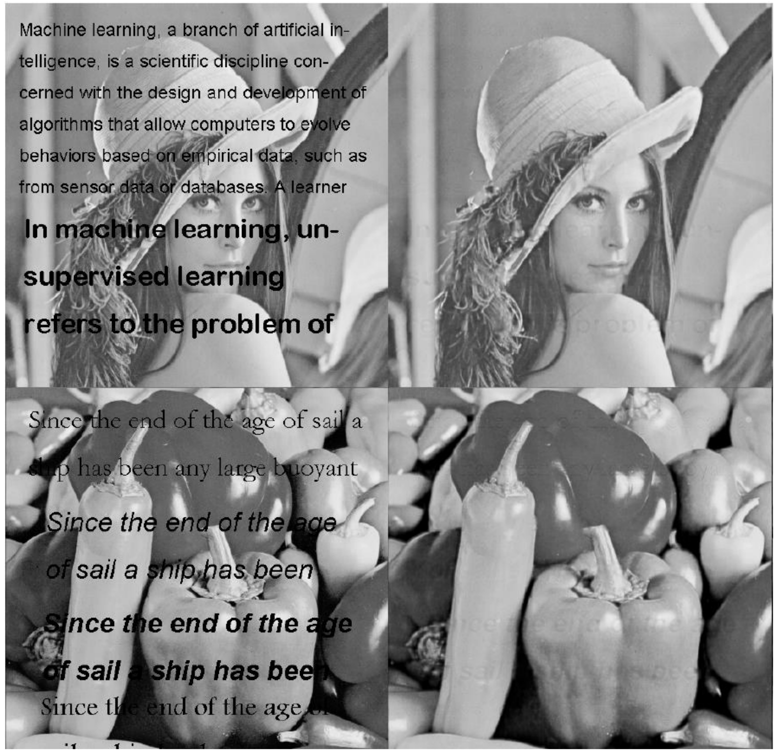


[source: Kirill Eremenko]



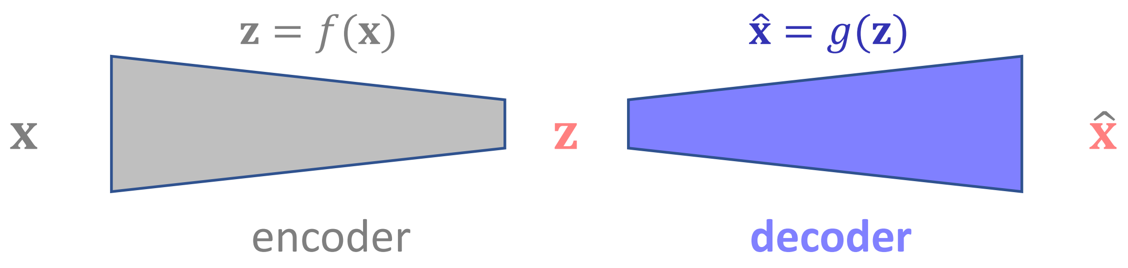
Vincent, H. Larochelle Y. Bengio and P.A. Manzagol, "**Extracting and Composing Robust Features with Denoising Autoencoders**", ICML 2008

- Dimensionality Reduction
- Pretraining
- Denoising AutoEncoder
- Image Inpainting



Are Auto-Encoders generative models?

not in principle!



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

Let’s make it probabilistic and truly generative!

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

Outline

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

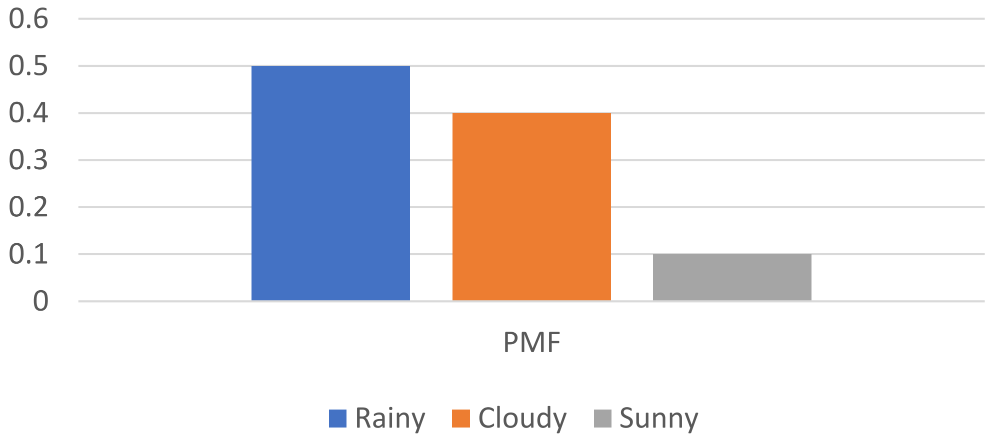
Variational Approximation

- We are usually dealing with an interactable inference
  - Sampling methods
  - techniques for approximate inference in probabilistic graphical models (e.g. Variational Inference)

KL-Divergence

- Imagine a set of possible events and their associated probabilities

- $x_1 = rainy, \quad p(x_1) = 0.5$
- $x_2 = cloudy, \quad p(x_2) = 0.4$
- $x_3 = sunny, \quad p(x_3) = 0.1$



Variational Approximation

Information: $\mathrm{I}_P(x) = -\log P(x)$ $\mathrm{I} \in [0, \infty),\ \mathbf{I}(x, y) = \mathrm{I}(x) + \mathrm{I}(y)$

Entropy: $\mathrm{H}(P) = \mathbb{E}_P[\mathrm{I}_P(x)]$ when is it 0? when is it maximized?

KL-divergence: $D_{KL}(P \parallel Q) = \mathbb{E}_P\left[\mathrm{I}_Q(x) - \mathrm{I}_P(x)\right]$

$$= \sum \big(-\log Q(x) + \log P(x)\big) P(x) \qquad =\sum P(x) log \frac{P(x)}{Q(x)}$$

$$= \sum -\mathrm{P}(x) \log Q(x) + \sum P(x) log P(x)$$

$$= CE(P, Q) - H(P)$$

KL-Divergence

- Also called “relative entropy”
- is asymmetric! $\sum P(x) log \frac{P(x)}{Q(x)}$
- defined only if for all *i*, *Q*(*x*) = 0 implies *P*(*x*) = 0 (absolute continuity)
- Always non-negative
  - Gibbs inequality: $\sum P(x) log P(x) \geq \sum P(x) log Q(x)$

Variational Inference

Variational Inference

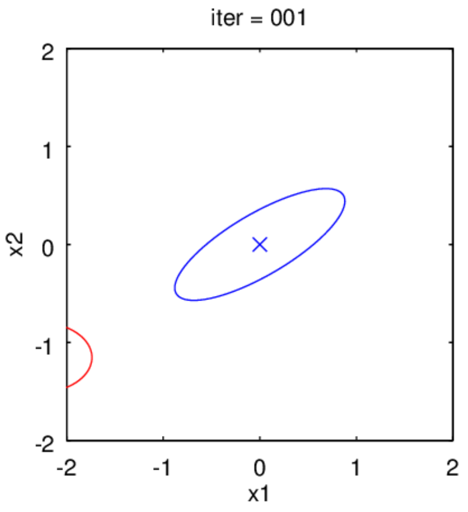
- We are interested in the true posterior $P(\mathbf{z}|\mathbf{x})$
- Bayes Rule: $P(\mathbf{z}|\mathbf{x}) = \frac{P(\mathbf{x}|\mathbf{z})P(\mathbf{z})}{P(\mathbf{x})}$ but $P(\mathbf{x})$ is intractable
- We consider a simple approximate distribution parametrized by $\boldsymbol{\theta}$ say $Q_{\boldsymbol{\theta}}(\mathbf{z})$
- And try to make them as similar as possible: $\min_{\boldsymbol{\theta}} KL(Q_{\boldsymbol{\theta}}(\mathbf{z})||P(\mathbf{z}|\mathbf{x}))$

Variational Inference

- What does it mean?

Variational Inference

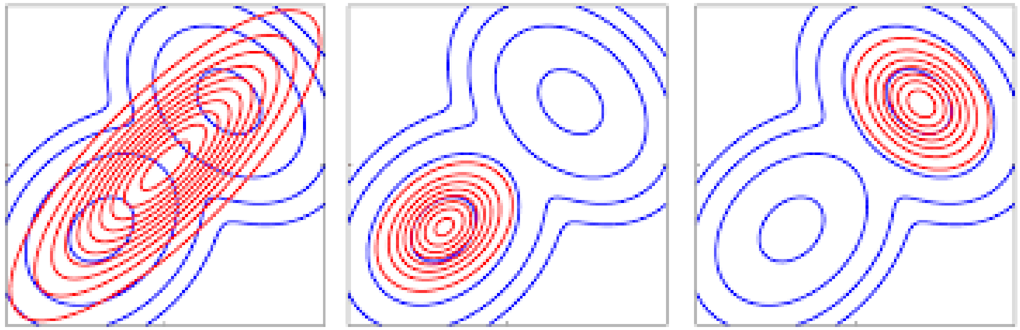
- The original inference becomes an optimization, and we are good at optimizations!
  - Optimizing parameters of the approximating family of distributions (referred to as variational parameters) for lowest distance between the approximating distribution and the true distribution (KL divergence)



KL-Divergence

- Asymmetry in KL divergence KL(P||Q): Intuitively, there are three cases, for an x
  - If P is high and Q is high then we are happy.
  - If P is high and Q is low then we pay a price.
  - If P is low then we don't care (because of the expectation over P(x))

$$\sum P(x) log \frac{P(x)}{Q(x)}$$



(a) (b) (c)

but wait, we didn't have access to the posterior in the first place, how can we optimize a distance to something we don't know then?

Variational Inference

$$KL(Q(\mathbf{z})||P(\mathbf{z}|\mathbf{x})) = -\sum Q(\mathbf{z}) \log \frac{P(\mathbf{z}|\mathbf{x})}{Q(\mathbf{z})}$$

$$= -\sum Q(\mathbf{z})[\log \frac{P(\mathbf{x},\mathbf{z})}{Q(\mathbf{z})} - \log P(\mathbf{x})]$$

$$= -\sum Q(\mathbf{z}) \log \frac{P(\mathbf{x},\mathbf{z})}{Q(\mathbf{z})} + \log P(\mathbf{x}) \sum Q(\mathbf{z})$$

$$\rightarrow \log P(\mathbf{x}) = KL(Q(\mathbf{z})||P(\mathbf{z}|\mathbf{x})) + \sum Q(\mathbf{z}) \log \frac{P(\mathbf{x},\mathbf{z})}{Q(\mathbf{z})}$$

constant

we want to minimize
And KL is always >= 0

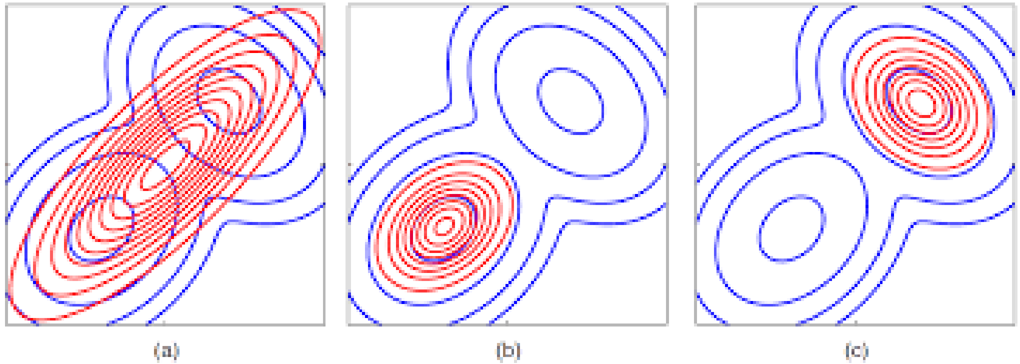
but, we can maximize this instead!

Variation Inference

- We call $\sum Q(z) log \frac{P(\mathbf{x},\mathbf{z})}{Q(\mathbf{z})}$
- Variational lower bound or evidence lower bound or evidence lower bound or ELBO for short!

- When we decided on the direction of the KL divergence to minimize, which one would make more sense:

- $KL(Q(\mathbf{z})||P(\mathbf{z}|\mathbf{x}))$ or
- $KL(P(\mathbf{z}|\mathbf{x})||Q(\mathbf{z}))$



But we won't get the nice lower bound anymore

How to solve variational approximation

- Assumptions!
- Mean field (factorization of latent varibles)
- EM
- …

$$\sum Q(\mathbf{z}) log \frac{P(\mathbf{x},\mathbf{z})}{Q(\mathbf{z})}$$

- We want to maximize ELBO given by $\sum Q(\mathbf{z}) log \frac{P(\mathbf{x},\mathbf{z})}{Q(\mathbf{z})}$

$$\sum Q(\mathbf{z}) log \frac{P(\mathbf{x},\mathbf{z})}{Q(\mathbf{z})} = \sum Q(\mathbf{z}) log \frac{P(\mathbf{x}|\mathbf{z})P(\mathbf{z})}{Q(\mathbf{z})}$$

$$= \sum Q(\mathbf{z})[log P(\mathbf{x}|\mathbf{z}) + log \frac{P(\mathbf{z})}{Q(\mathbf{z})}]$$

$$= E_{Q(\mathbf{z})} log P(\mathbf{x}|\mathbf{z}) - KL(Q(\mathbf{z})||P(\mathbf{z}))$$

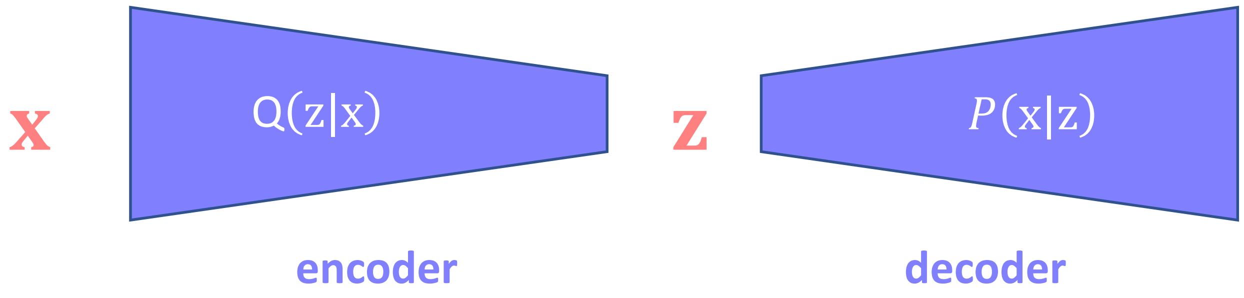
$$\max_{\boldsymbol{\theta}}\left(E_{Q_{\boldsymbol{\theta}}(\mathbf{z})} log P(\mathbf{x}|\mathbf{z}) - KL(Q_{\boldsymbol{\theta}}(\mathbf{z})||P(\mathbf{z}))\right)$$

data likelihood

prior

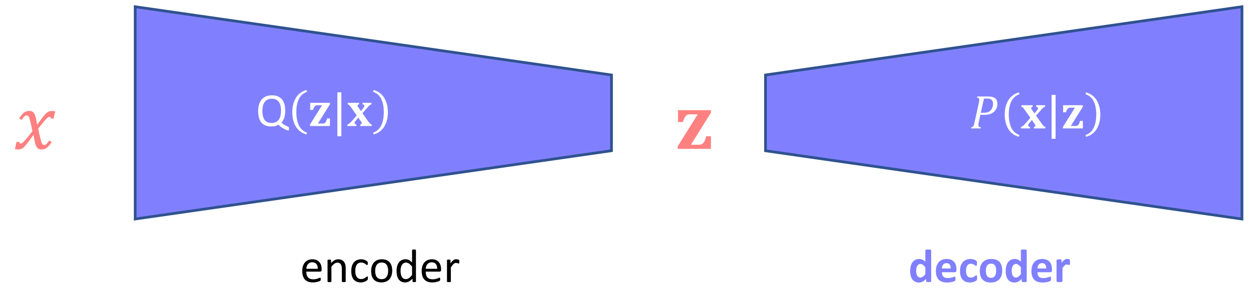
Variational AutoEncoder

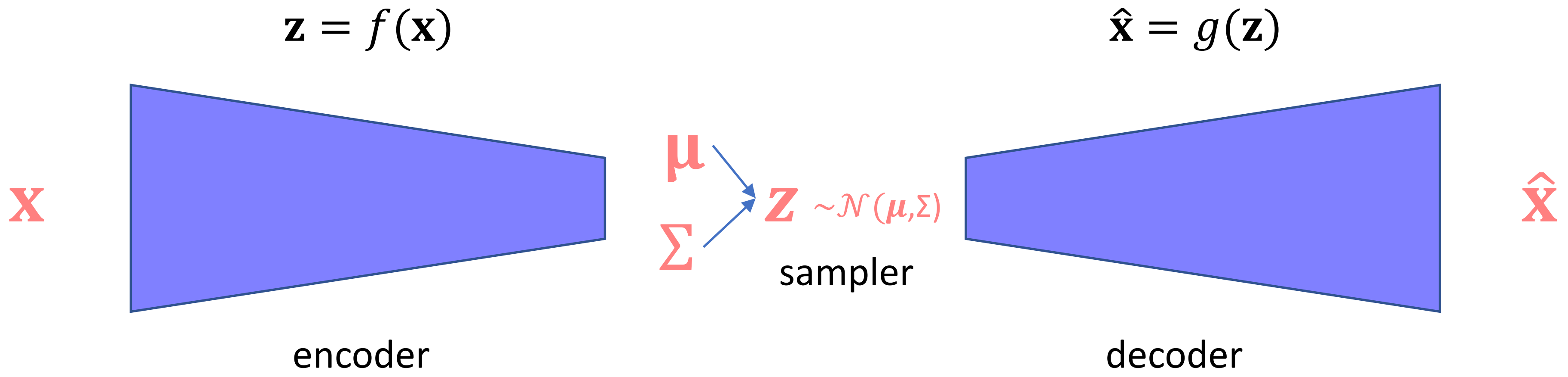
- In VAE $\mathbf{z}$ is the code
- Assume we model $P(\mathbf{x}|\mathbf{z})$ as a (deterministic) decoder
- Also assume that our approximate $\mathrm{Q}(\mathbf{z}|\mathbf{x})$ as a probabilistic encoder



- $E_{Q(\mathbf{z})} log P(\mathbf{x}|\mathbf{z}) - KL(Q(\mathbf{z})||P(\mathbf{z}))$

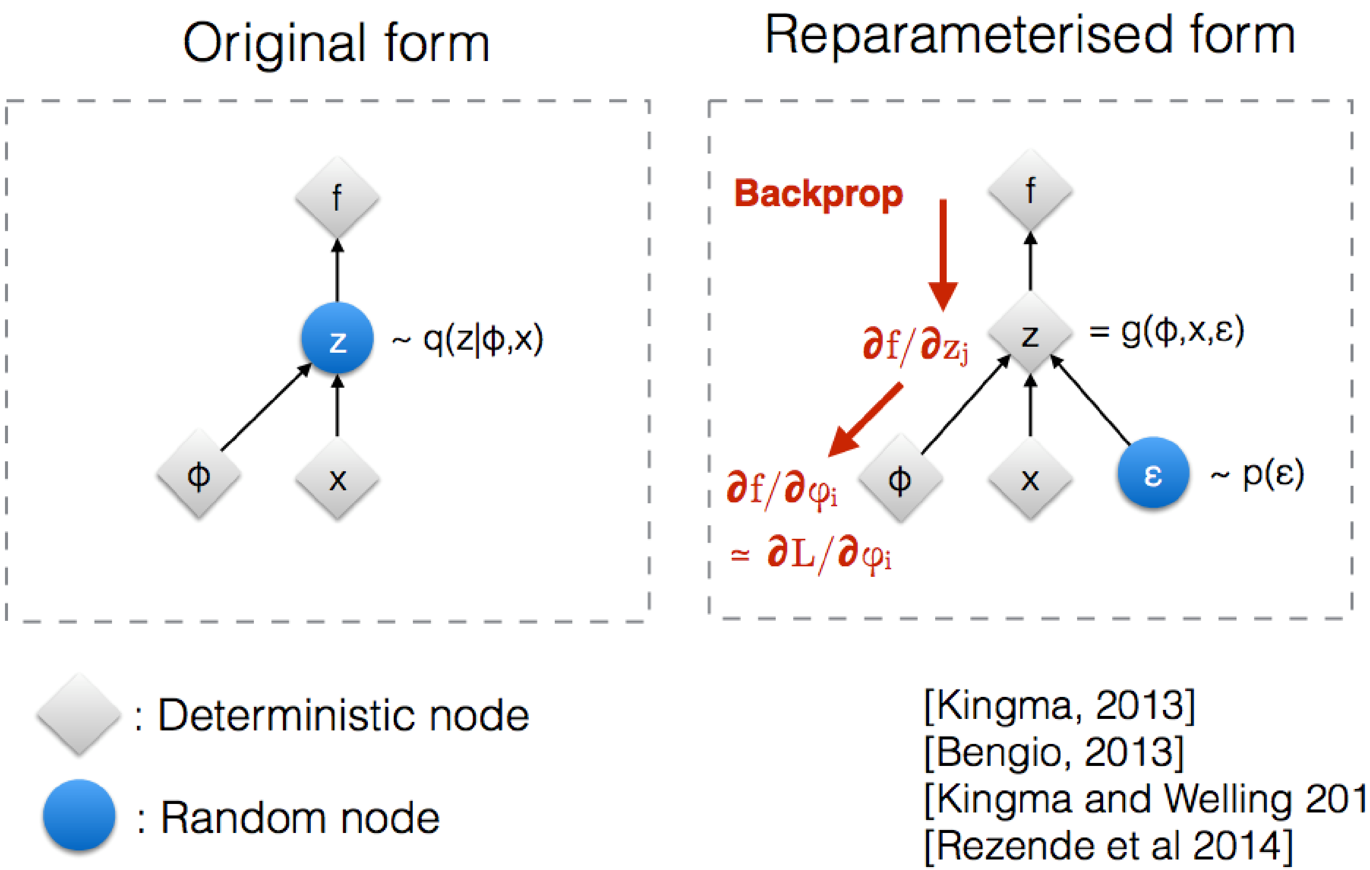
- Now we choose simple/proper form for P(z) and Q(z) for tractability!
  - Gaussian!
  - $P(\mathbf{z}) \quad \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$
  - $Q(\mathbf{z}|\mathbf{x}) \sim \mathcal{N}\big(\mu(\mathbf{x}), \Sigma(\mathbf{x})\big)$
  - $P(\mathbf{x}|\mathbf{z}) \sim \mathcal{N}\big(\mu(\mathbf{z}), \Sigma(\mathbf{z})\big)$ or some deterministic $g(\mathbf{x})$





But how can we backpropagate through a stochastic node?!

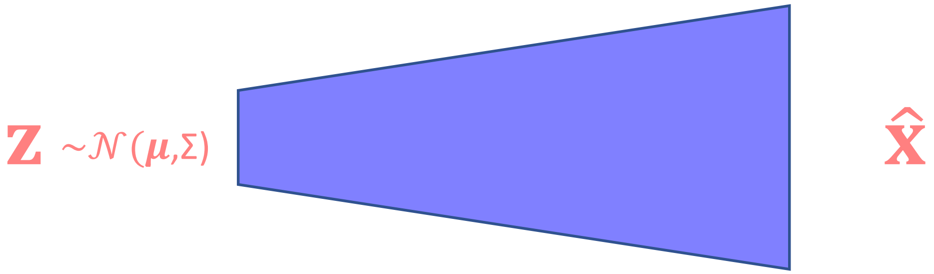
- But how can we backpropagate through a stochastic node?!



[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

VAE as generative model

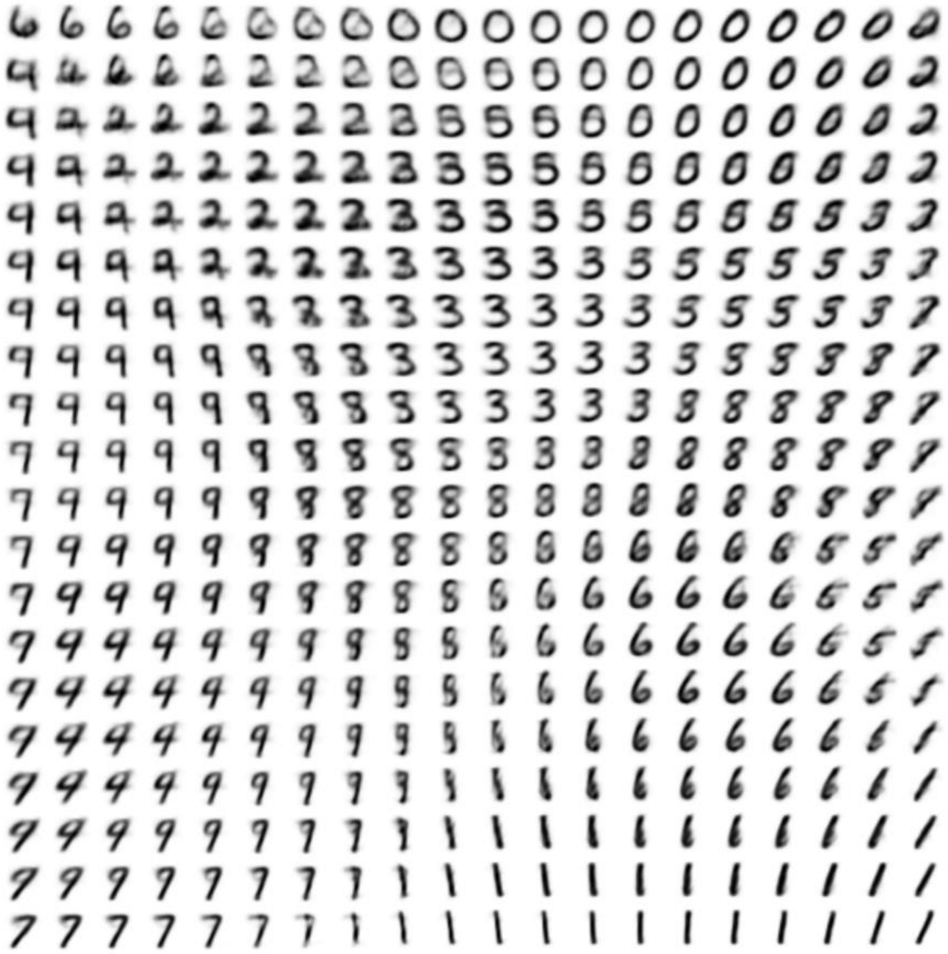
- Sample $\mathbf{z}$ from $\mathcal{N}(\mathbf{0}, \mathbf{1})$
- Then sample $\mathbf{x}$ using from $P(\mathbf{x}|\mathbf{z})$



Outline

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

VAE as generative model



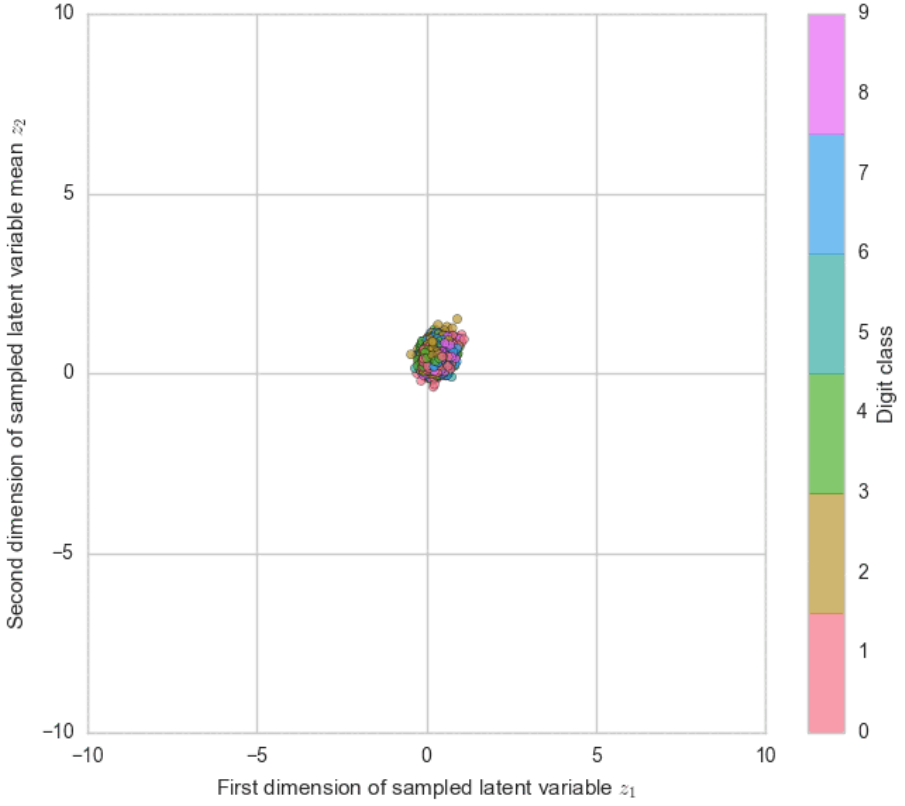
Intuition of Variotional AutoEncoder as a Generative Model

- Learning the original manifold is hard (high dimensional, non-linear, etc.)

- Learn a mapping from the hard distribution to a simple distribution (Encoder)

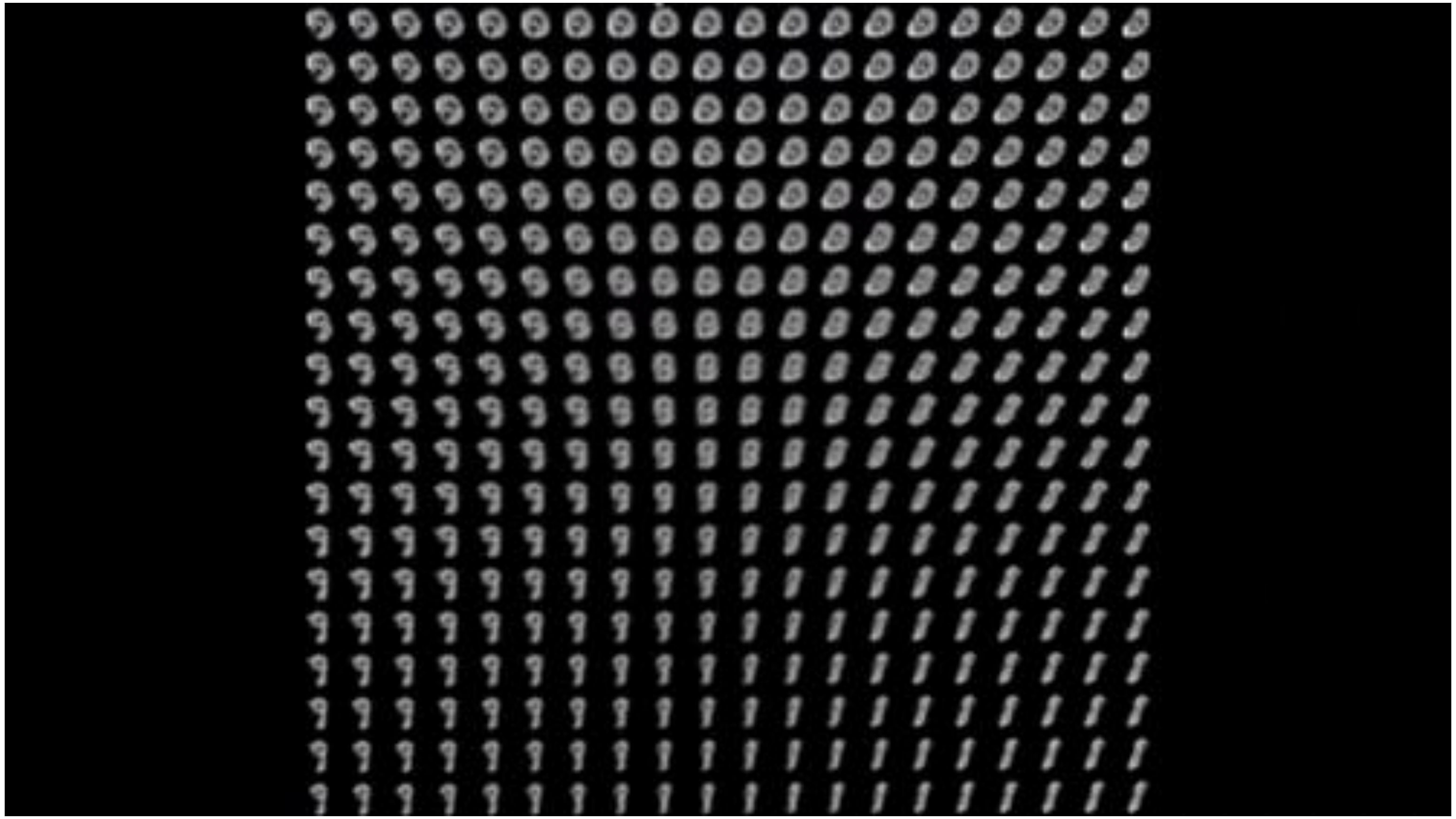
- Learn a remapping from the simple distribution to actual space (Decoder)

VAE (Example)



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

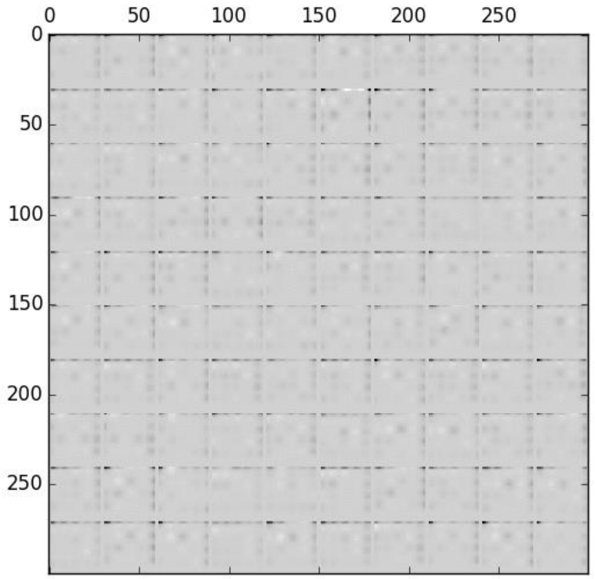
VAE (example)



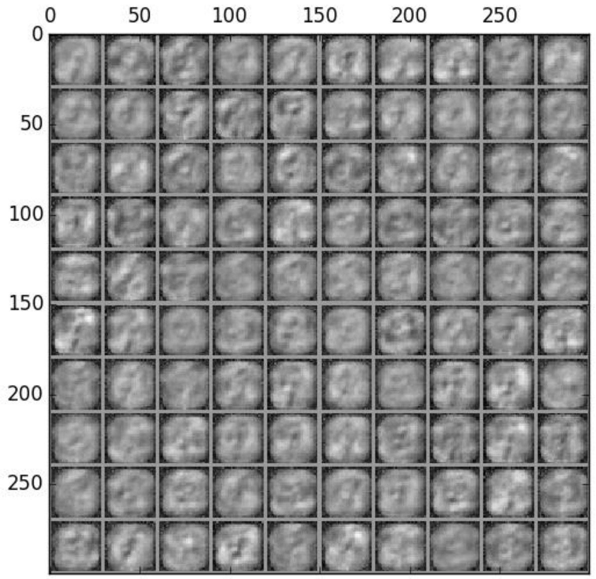
VAE (Example)

- DRAW

with attention



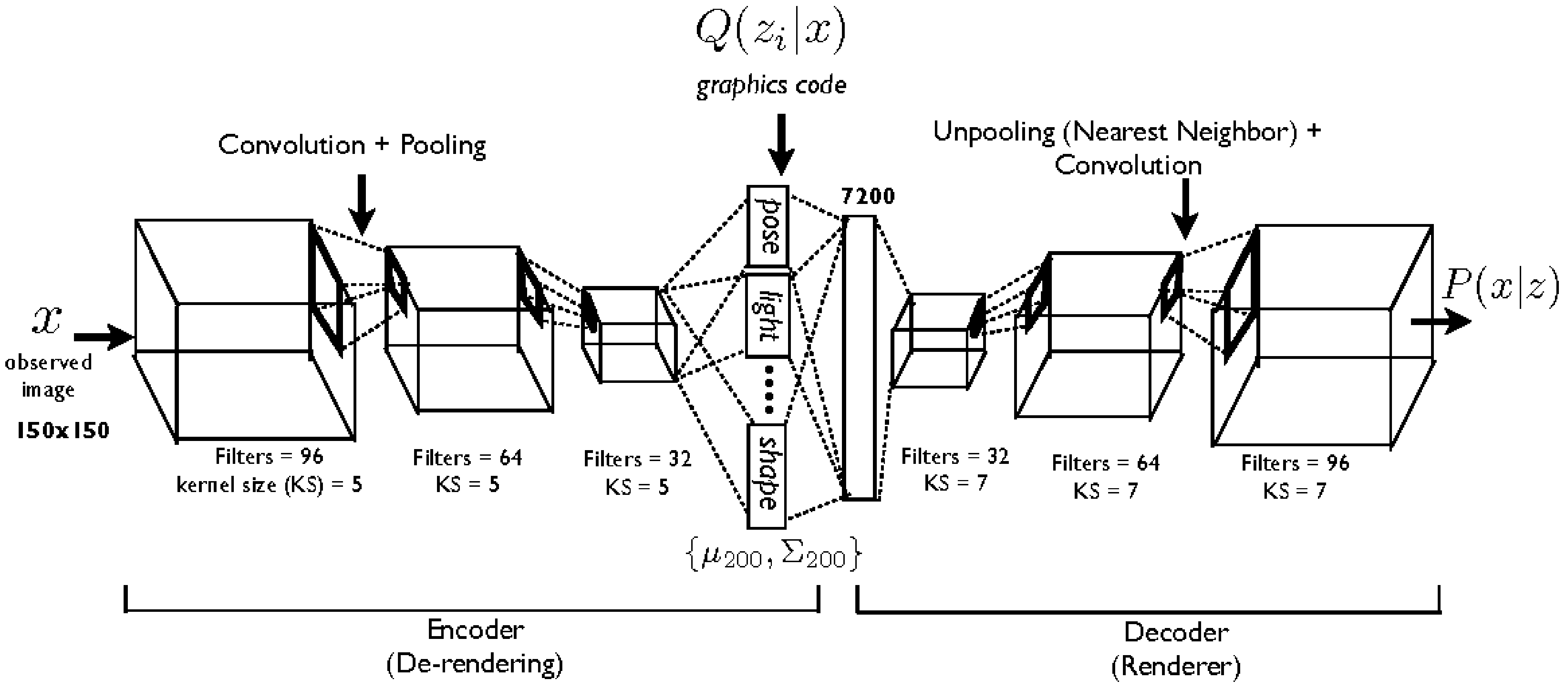
without attention



Gregor et al “**DRAW: A Recurrent Neural Network For Image Generation**” ICML 2015

https://github.com/ericjang/draw

Disentangled Variational AutoEncoder



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

- Probabilistic inference with simple interpretations
- (Slightly) Blurry images

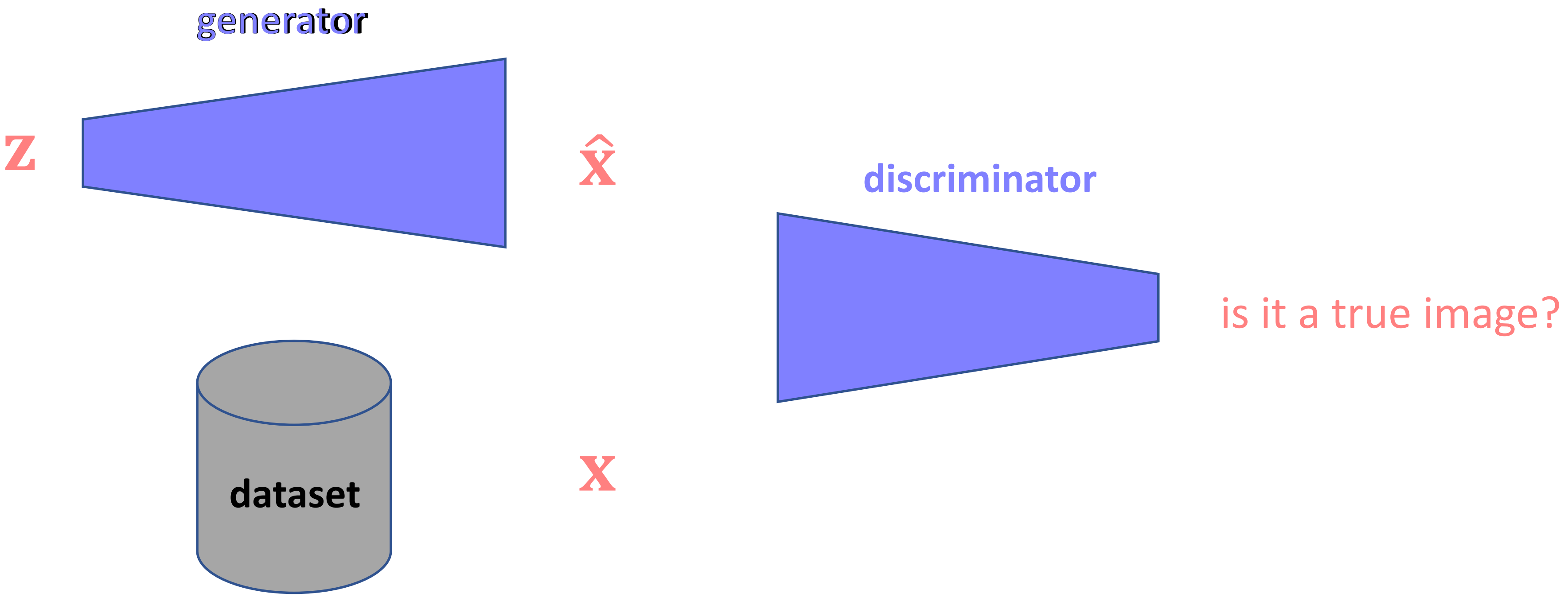
Outline

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

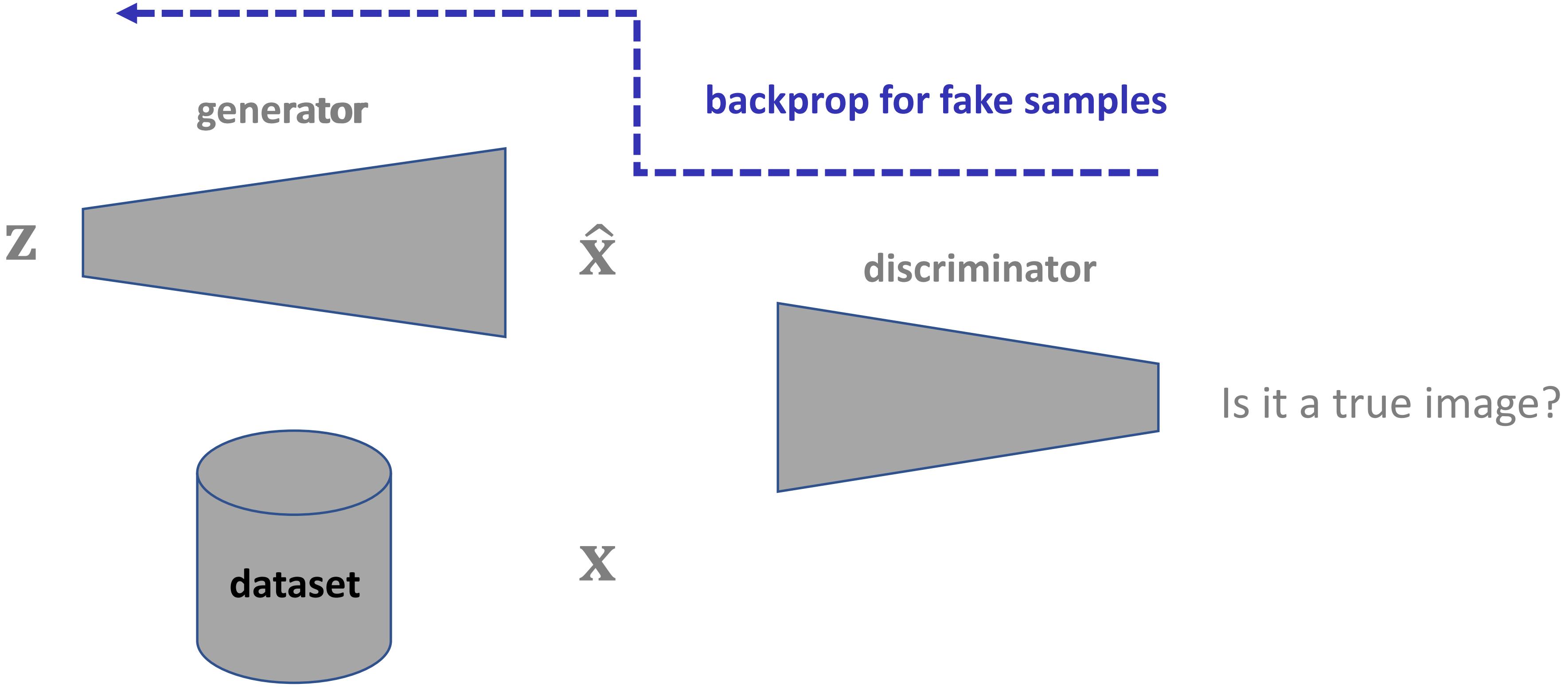
Generative Adversarial Networks

- 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

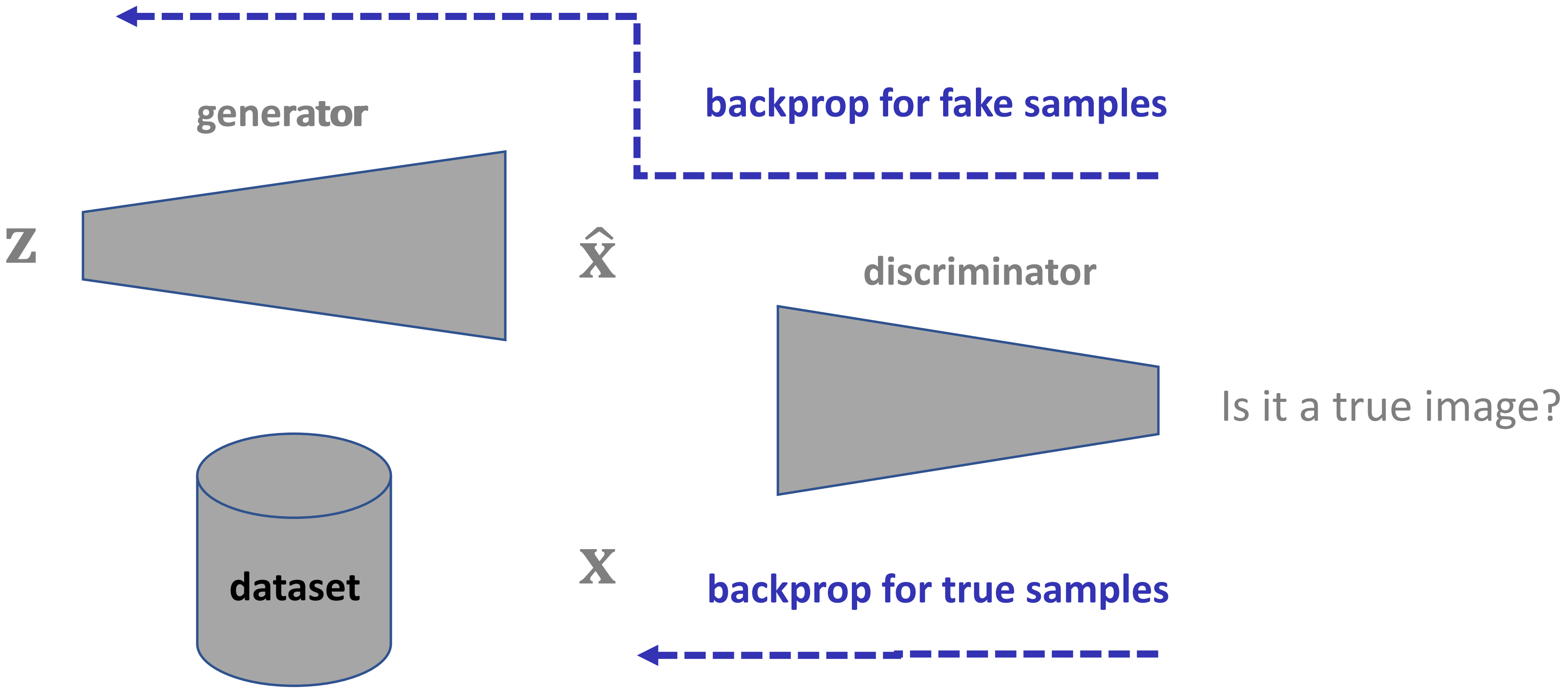
- Adversarial:



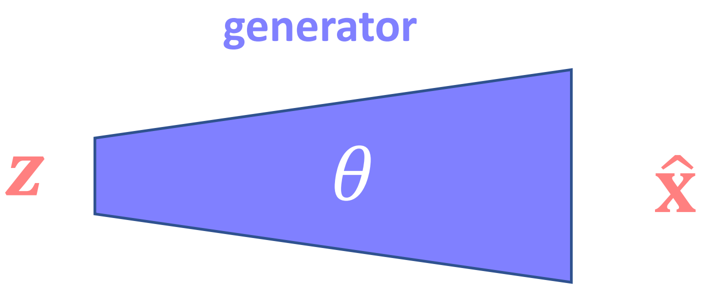
- Key is end-to-end learning (backprop):



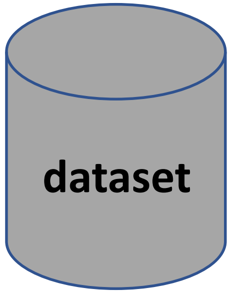
- Key is end-to-end learning (backprop):



- Loss function

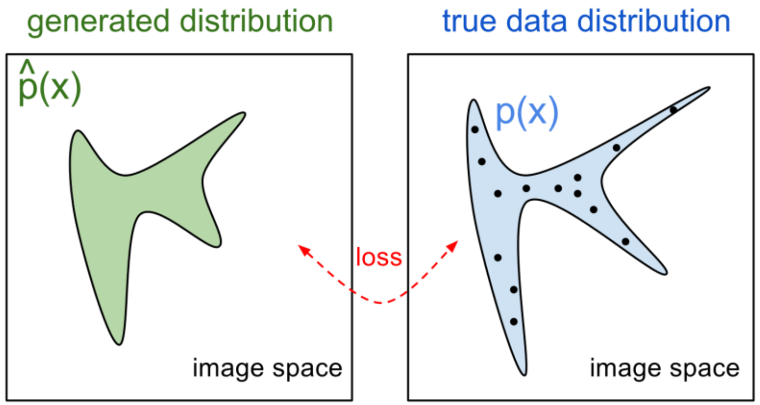


$$\hat{P}_\theta(\mathbf{x})$$



$\mathbf{x}$

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

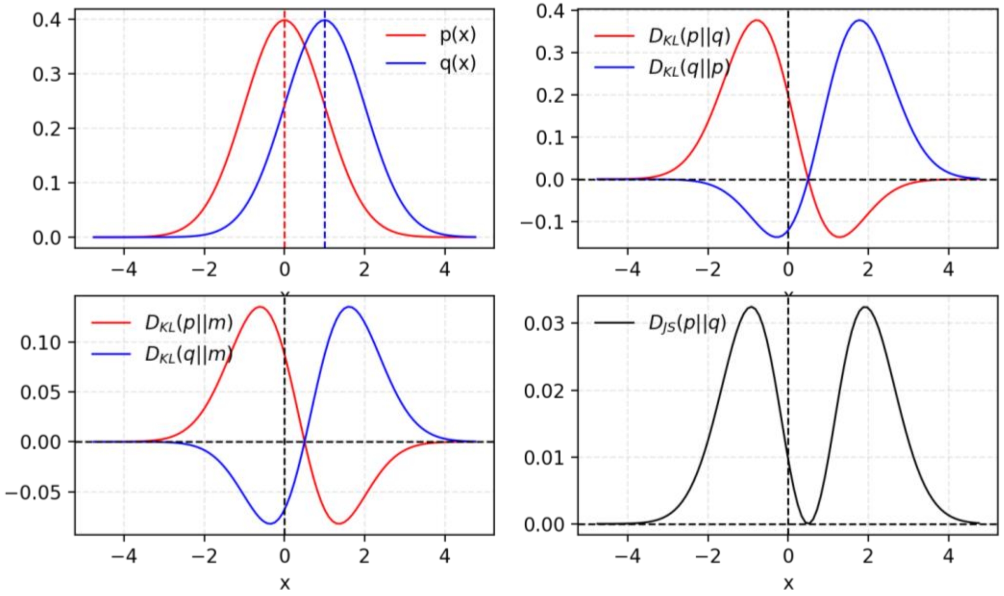


[openAI]

- KL-divergence: $D_{KL}(P||Q) = \sum_x P(x) log \frac{P(x)}{Q(x)}$

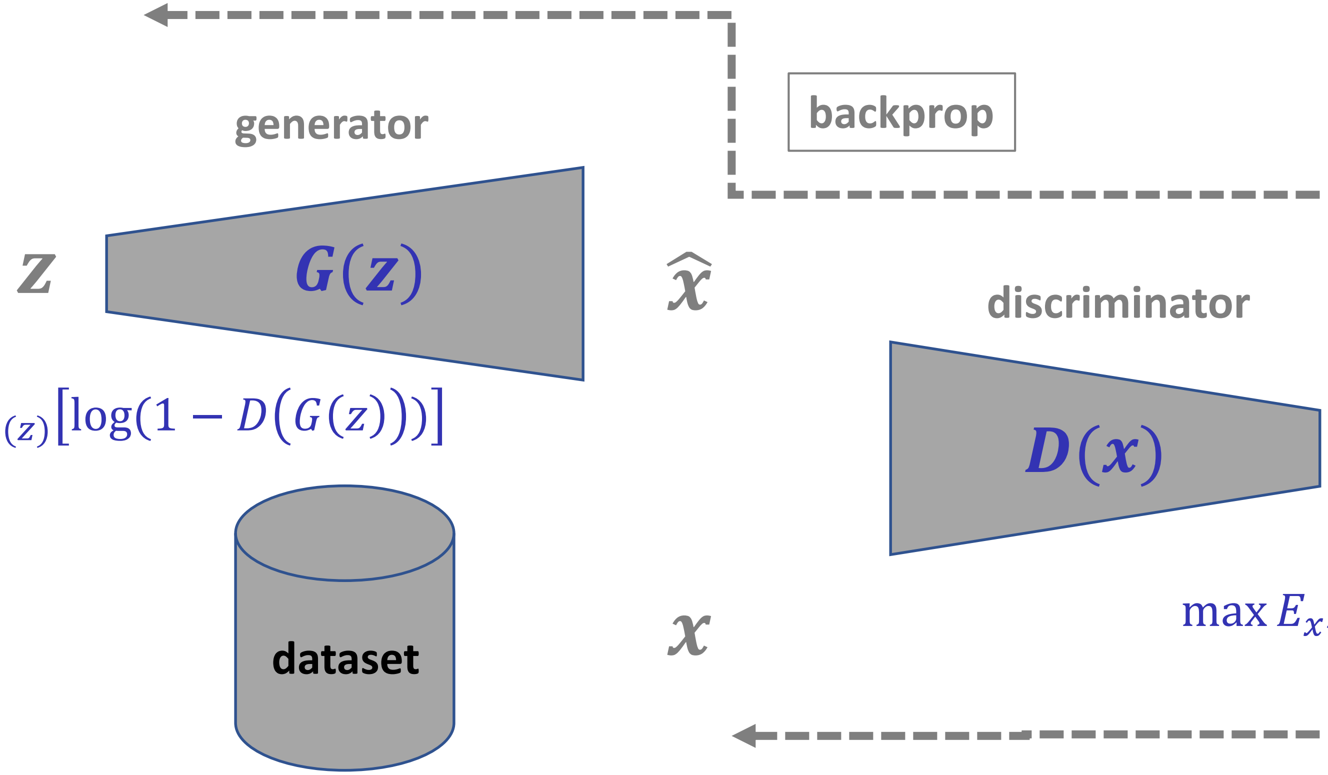
- Jensen–Shannon Divergence: $D_{JS}(P||Q) = .5 * D_{KL}\left(P||^{(P+Q)}/_2\right) + .5 * D_{KL}\left(Q||^{(P+Q)}/_2\right)$

m=(p+q)/2



https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

Let’s see what objective we optimize in GANs



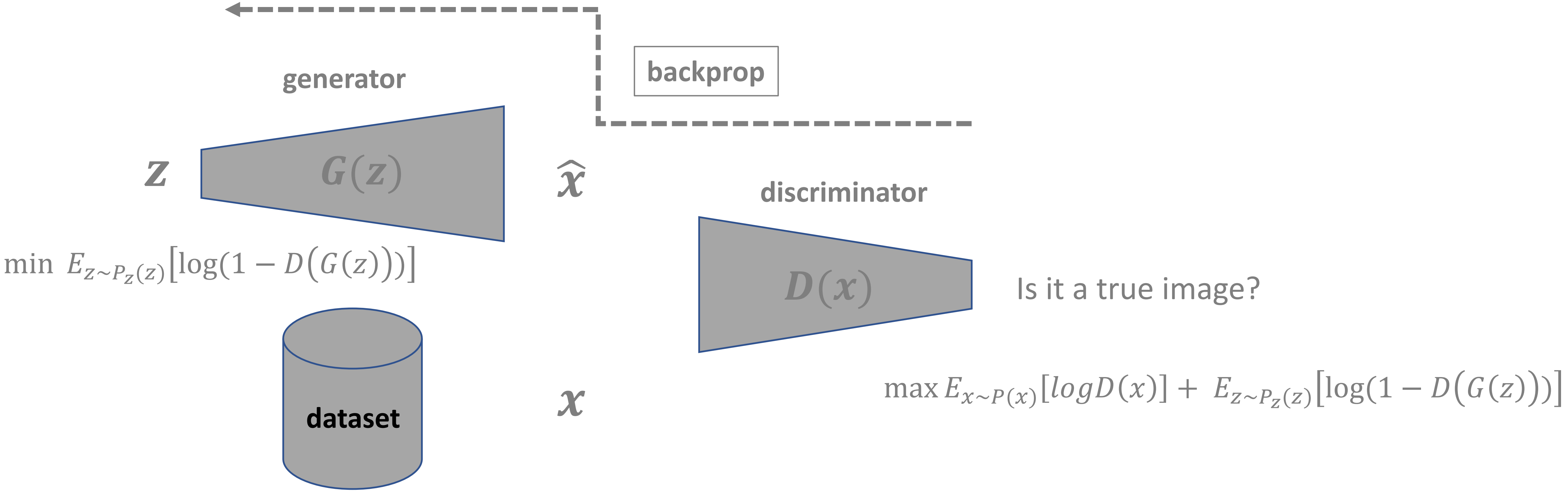
$$\min \; E_{z \sim P_z(z)}[\log(1 - D(G(z)))]$$

Is it a true image?

$$\max E_{x \sim P(x)}[log D(x)] + \; E_{z \sim P_z(z)}[\log(1 - D(G(z)))]$$

Let's see what objective we optimize in GANs

$$\min_{G} \max_{D} \; E_{x \sim P(x)}[\log D(x)] + E_{z \sim P_z(z)}[\log(1 - D(G(z)))]$$



Training GAN

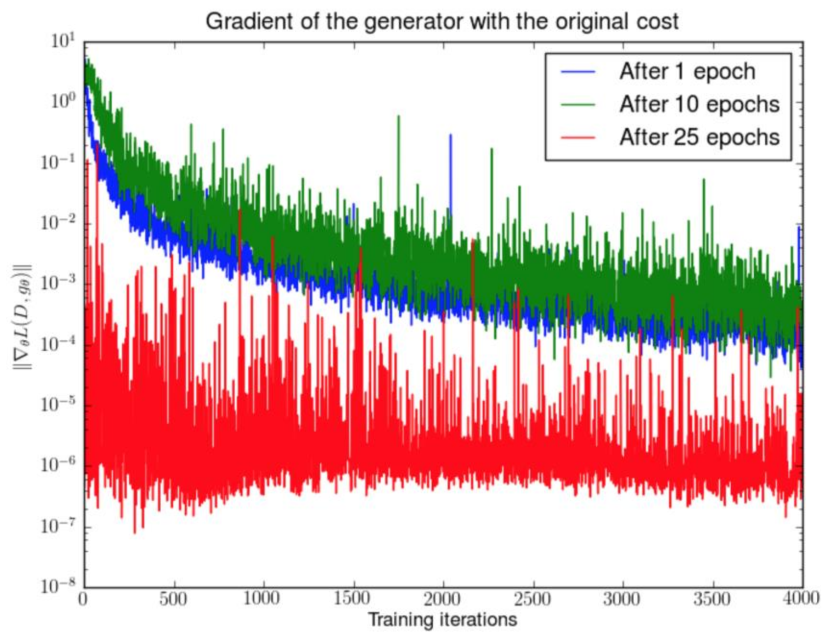
- Iterative Alternating Fashion

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

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

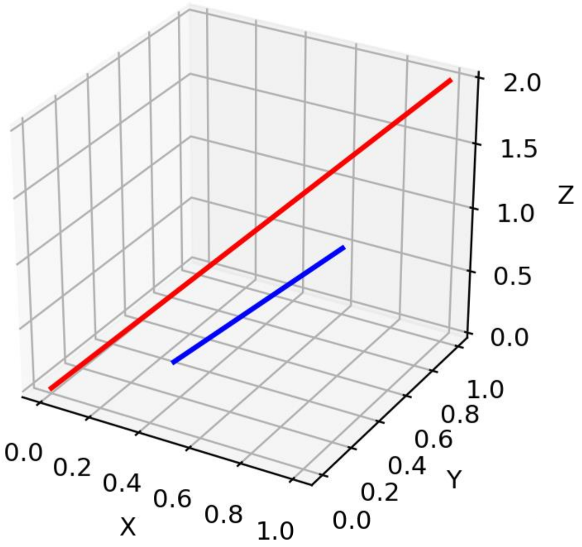
- Balance of power

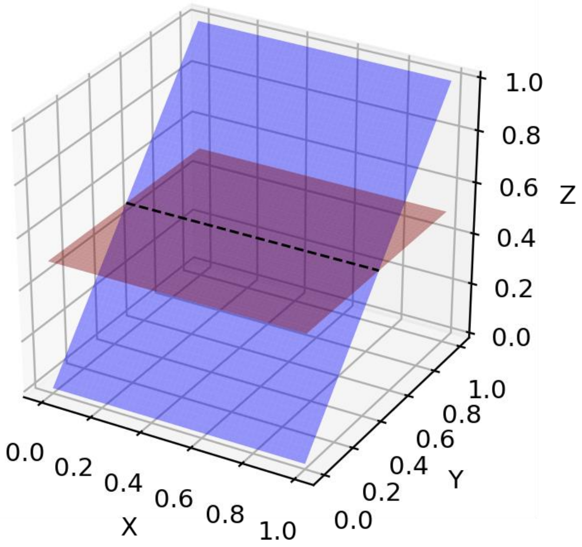


[Arjovsky and Bottou, 2017]

- Low-dimensional support of both $P(\boldsymbol{x})$ and $\hat{P}_\theta(\boldsymbol{x})$

- D can discriminate all the time!



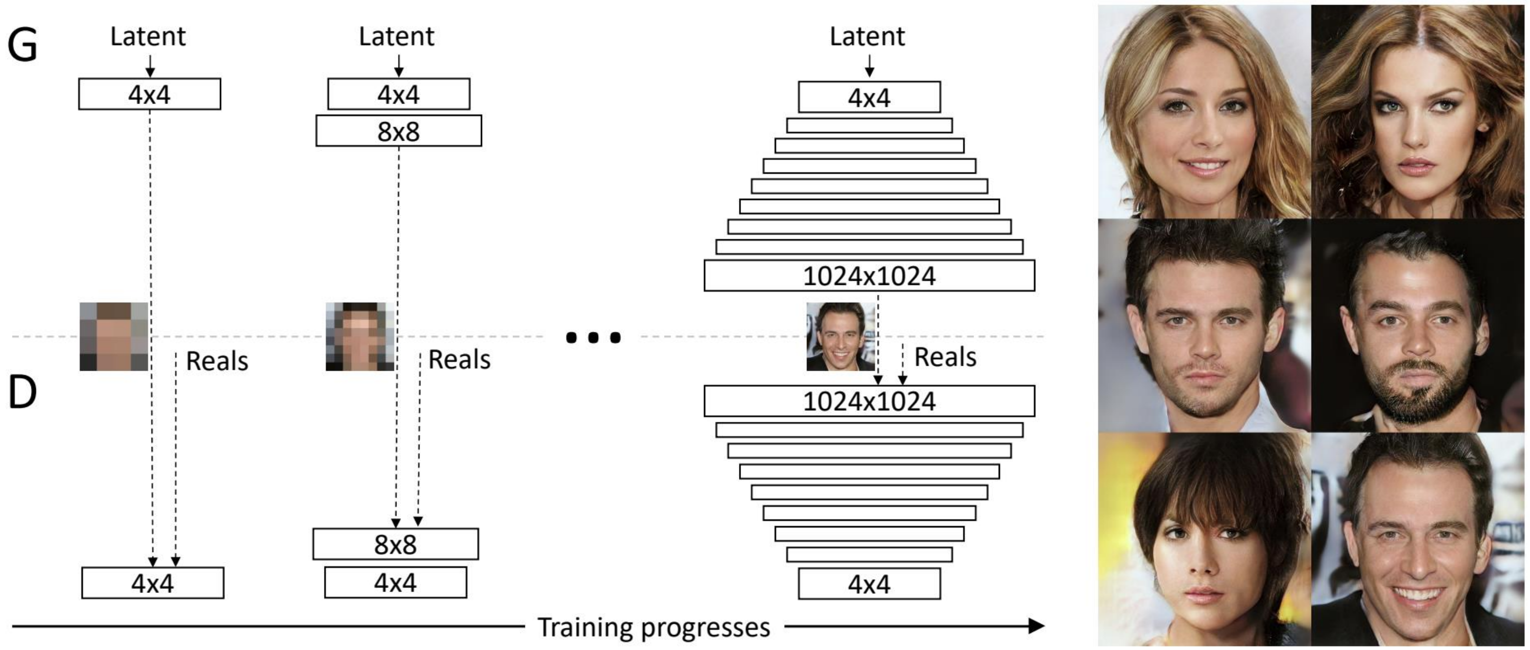


[Arjovsky&Bottou "TOWARDS PRINCIPLED METHODS FOR TRAINING GENERATIVE ADVERSARIAL NETWORKS"]

Training GAN

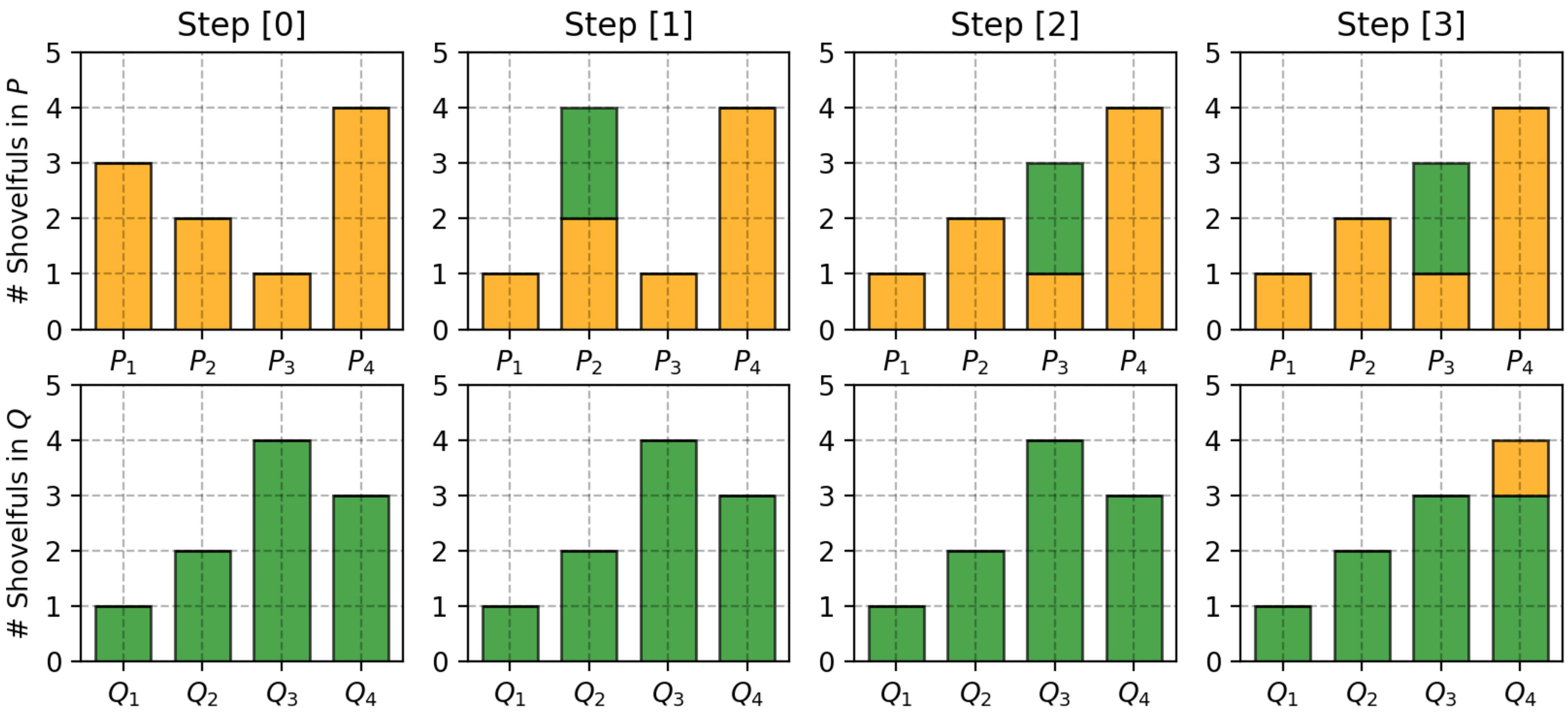
- Usually the discriminator "wins"
- This is a good thing—the theoretical justifications are based on assuming *D* is perfect
- Usually *D* is bigger and deeper than *G*
- Sometimes run *D* more often than *G*. Mixed results.
- Do not try to limit *D* to avoid making it "too smart"

Progressive GAN



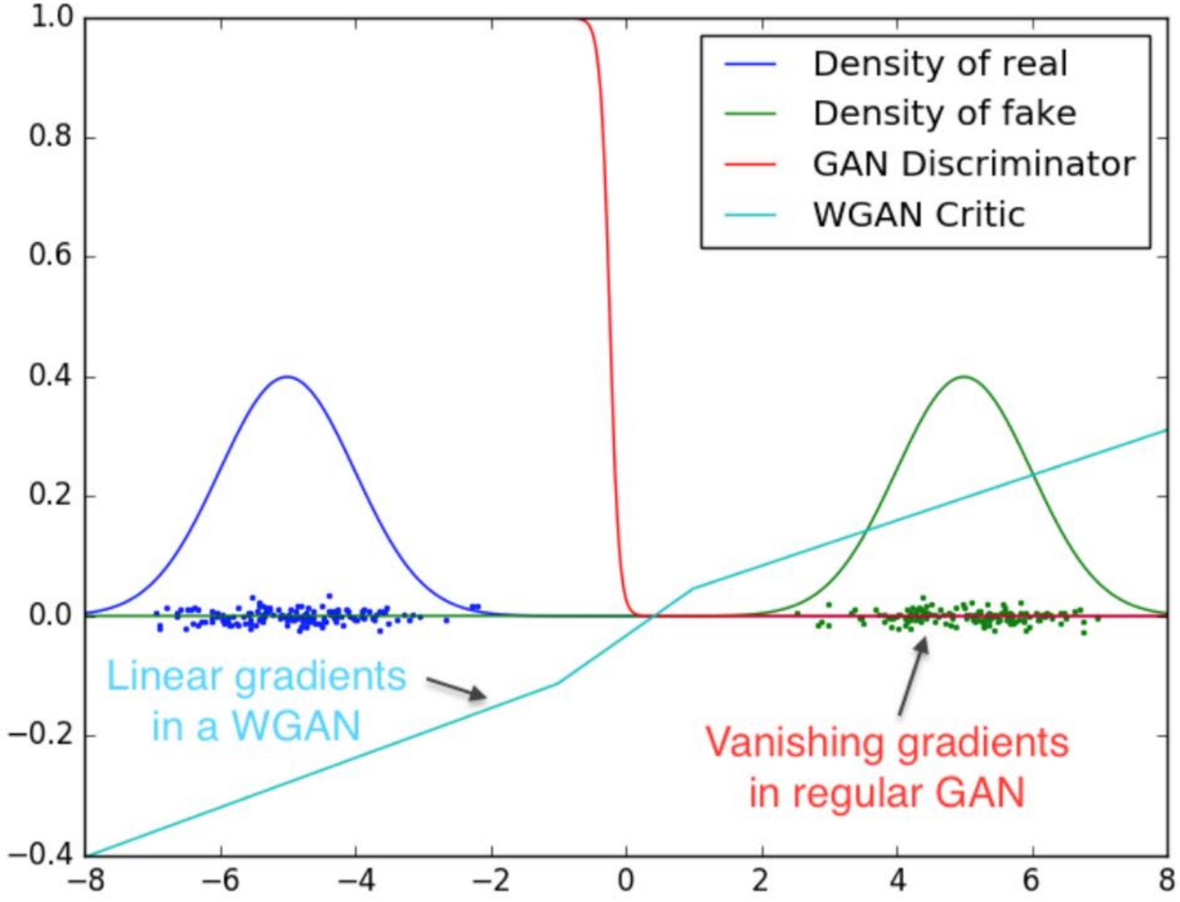
Karras et al. "**Progressive Growing of GANs for Improved Quality, Stability, and Variation**", ICLR 2018

Or Earth Mover's distance



https://lilianweng.github.io/

Wasserstein GAN



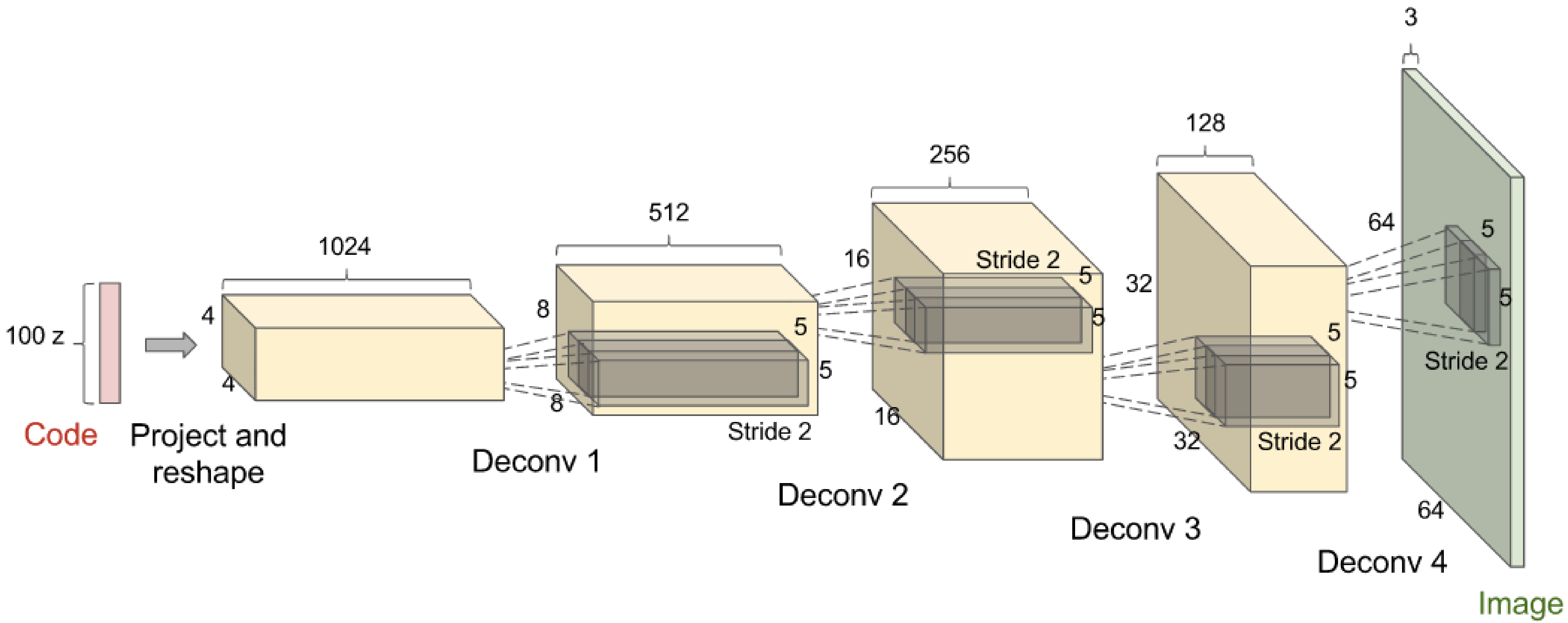
[Martin Arjovsky1 , Soumith Chintala2 , and L´eon Bottou, “Wasserstein GAN” 2017]]

Wasserstein GAN

- Works better when there is no overlap between true and fake distributions
- Attacks two problems:
  - Stabilizing the training
  - Convergence criteria
- No log in the loss. The output of DD is no longer a probability, hence we do not apply sigmoid at the output of DD
- Clip the weight of DD
- Train DD more than GG
- Use RMSProp instead of ADAM
- Lower learning rate, the paper uses α=0.00005
- Gradient Penalty (Gulrajani et al. 2017)

Arjovsky et al "**Wasserstein GAN**", 2016

DC GAN

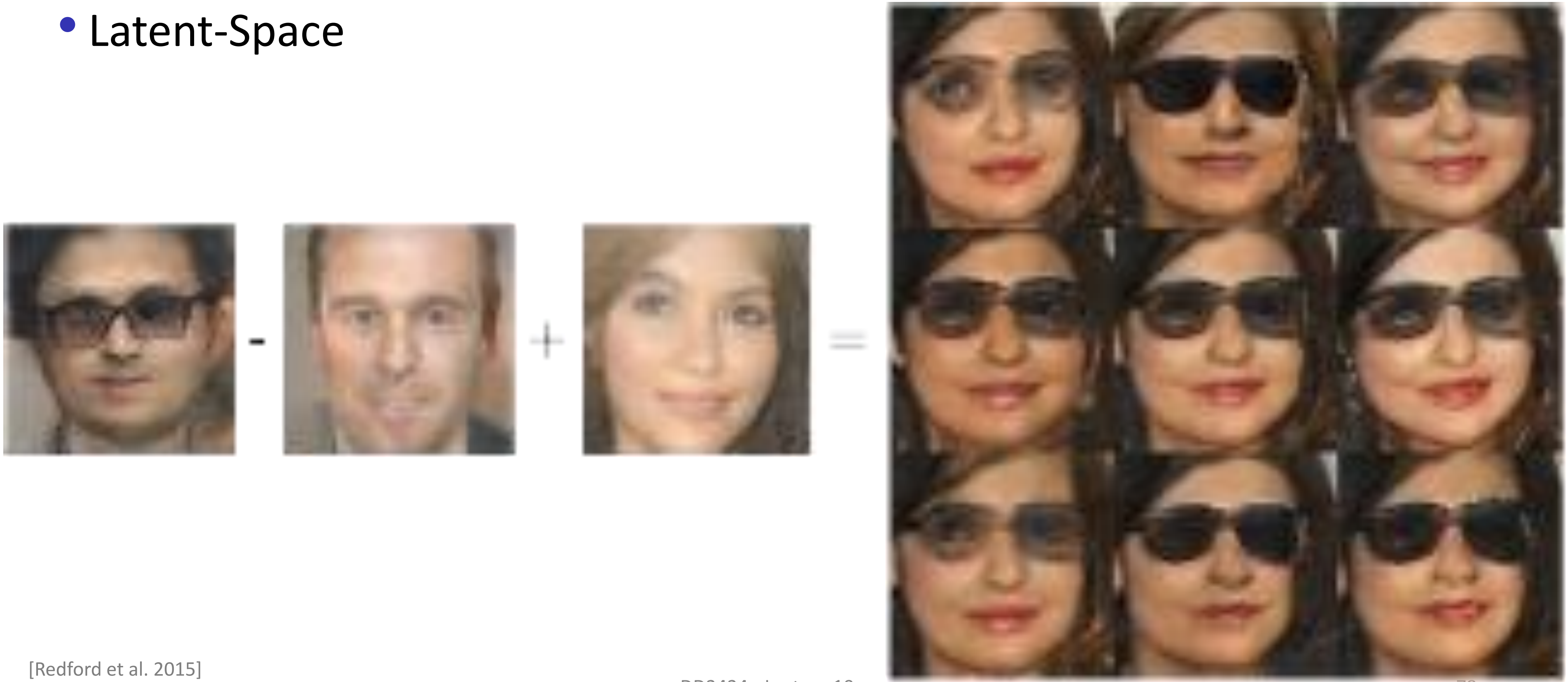


[Alec Radford, Luke Metz, Soumith Chintala, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016]

- LSUN Bedrooms

[Redford et al. 2015]

- Latent-Space

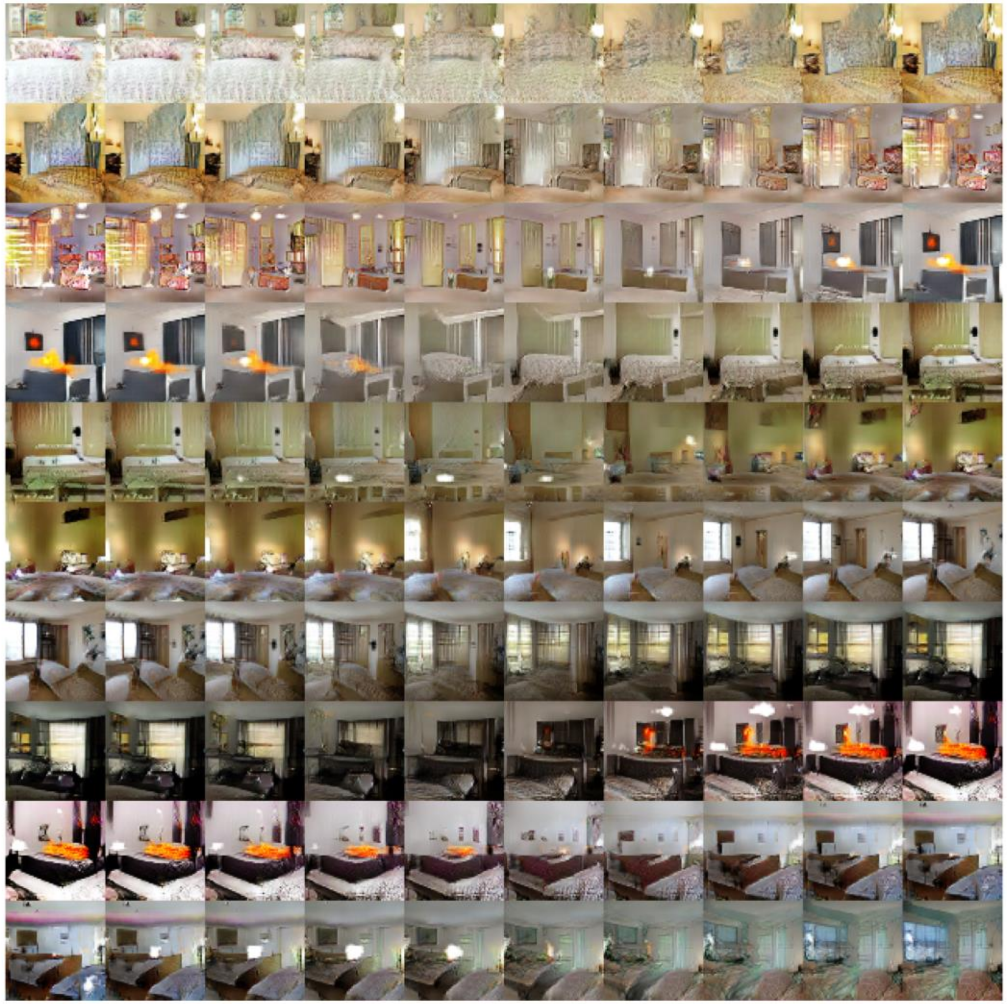


[Redford et al. 2015]

- Latent-Space

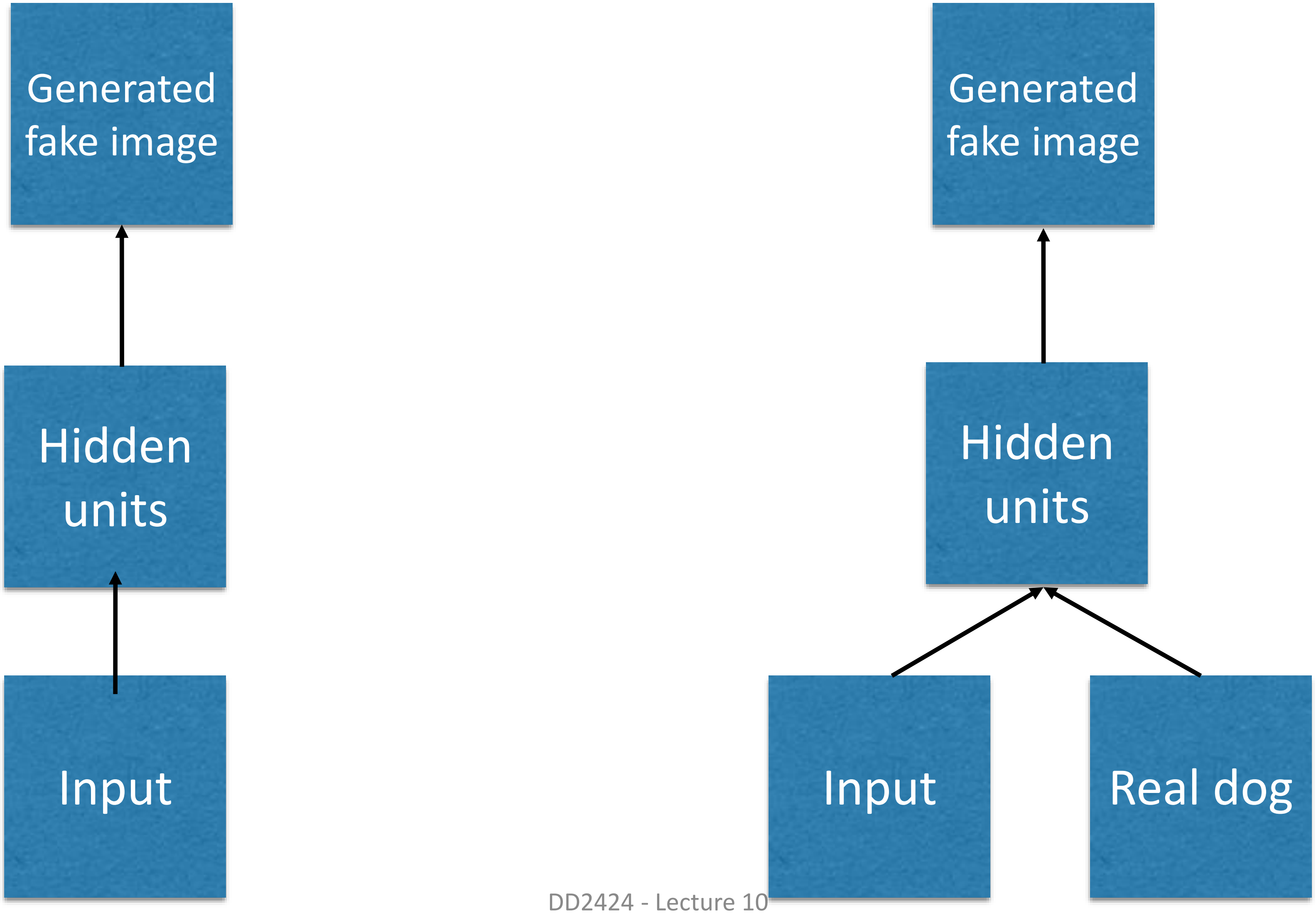
[Redford et al. 2015]

- Latent-Space



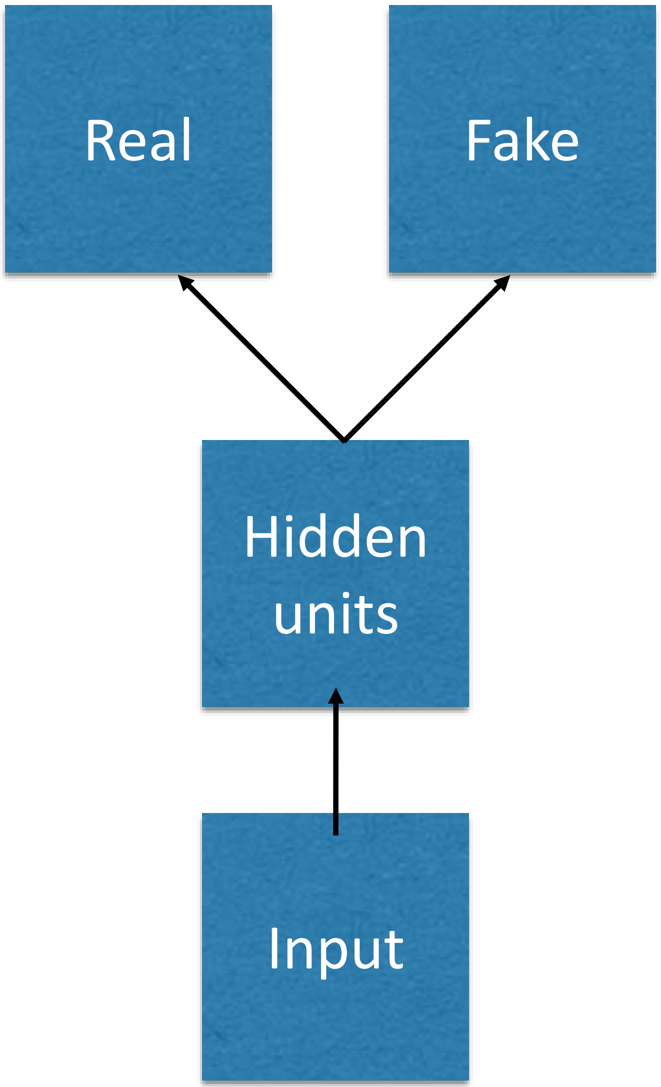
[Redford et al. 2015]

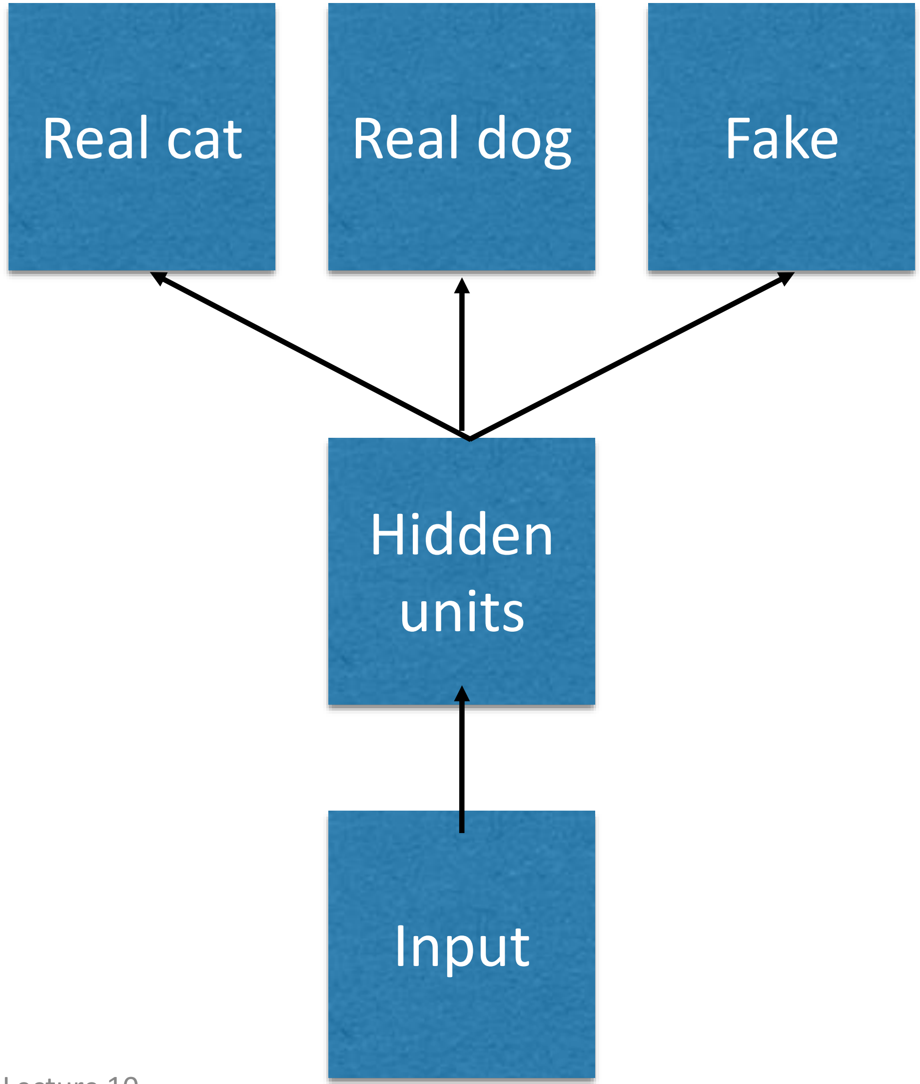
Conditional GAN



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

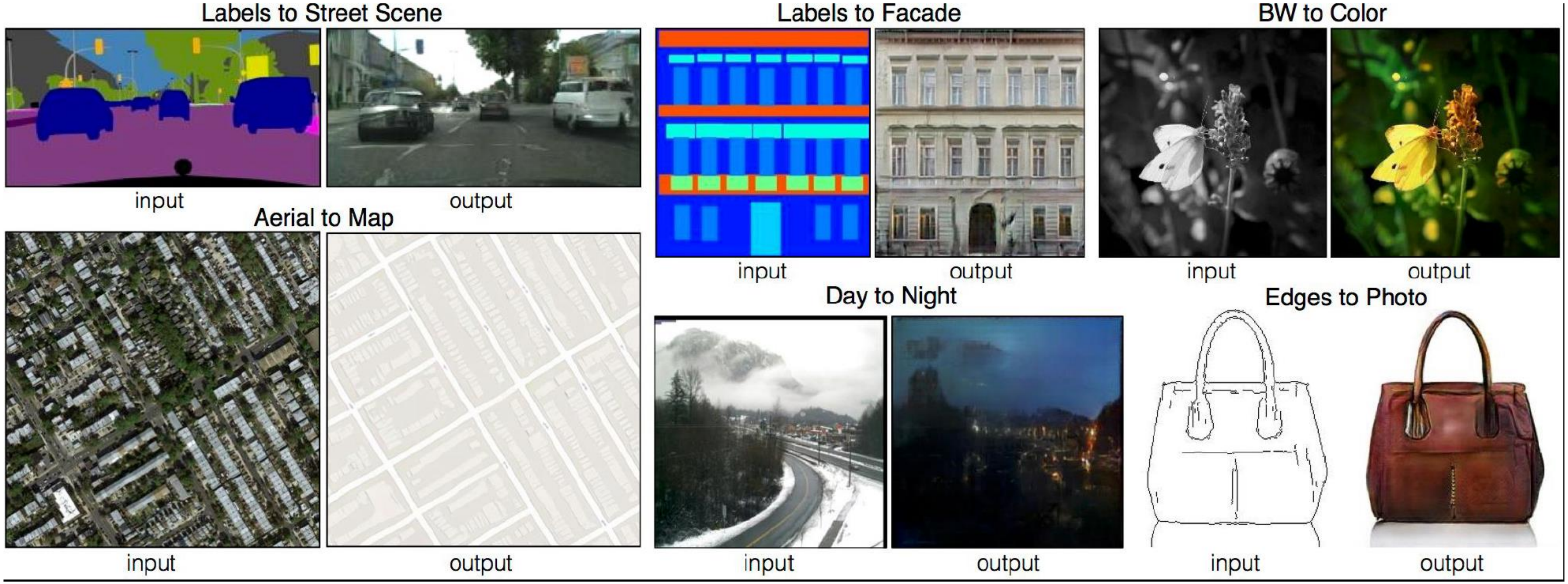
Conditional GAN





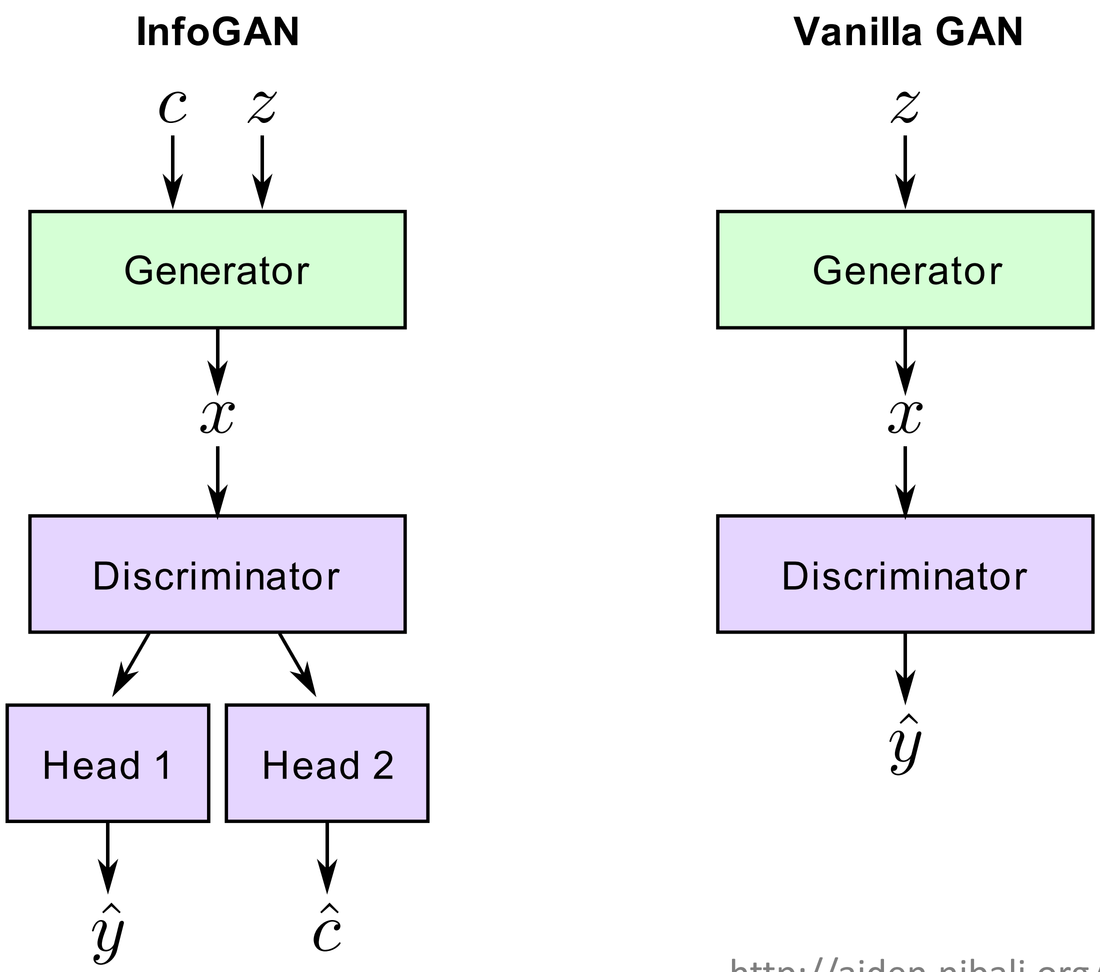
[Odena et al 2016, Salimans et al. 2106)

Conditional GAN



[Zhu et al. 2016]

InfoGAN



http://aiden.nibali.org/blog/2016-12-01-implementing-infogan/

Label Smoothing

- Default discriminator cost:
  - ```
    cross_entropy(1., discriminator(data))
    + cross_entropy(0., discriminator(samples))
    ```
- One-sided label smoothed cost (Salimans et al 2016):
  - ```
    cross_entropy(.9, discriminator(data))
    + cross_entropy(0., discriminator(samples))
    ```

- Benefits
  - Good regularizer (Szegedy et al 2015)
  - Does not reduce classification accuracy, only confidence
- Benefits specific to GANs:
  - Prevents discriminator from giving very large gradient signal to generator
  - Prevents extrapolating to encourage extreme samples

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

BN in GAN

- Too much dependence to mini-batch members

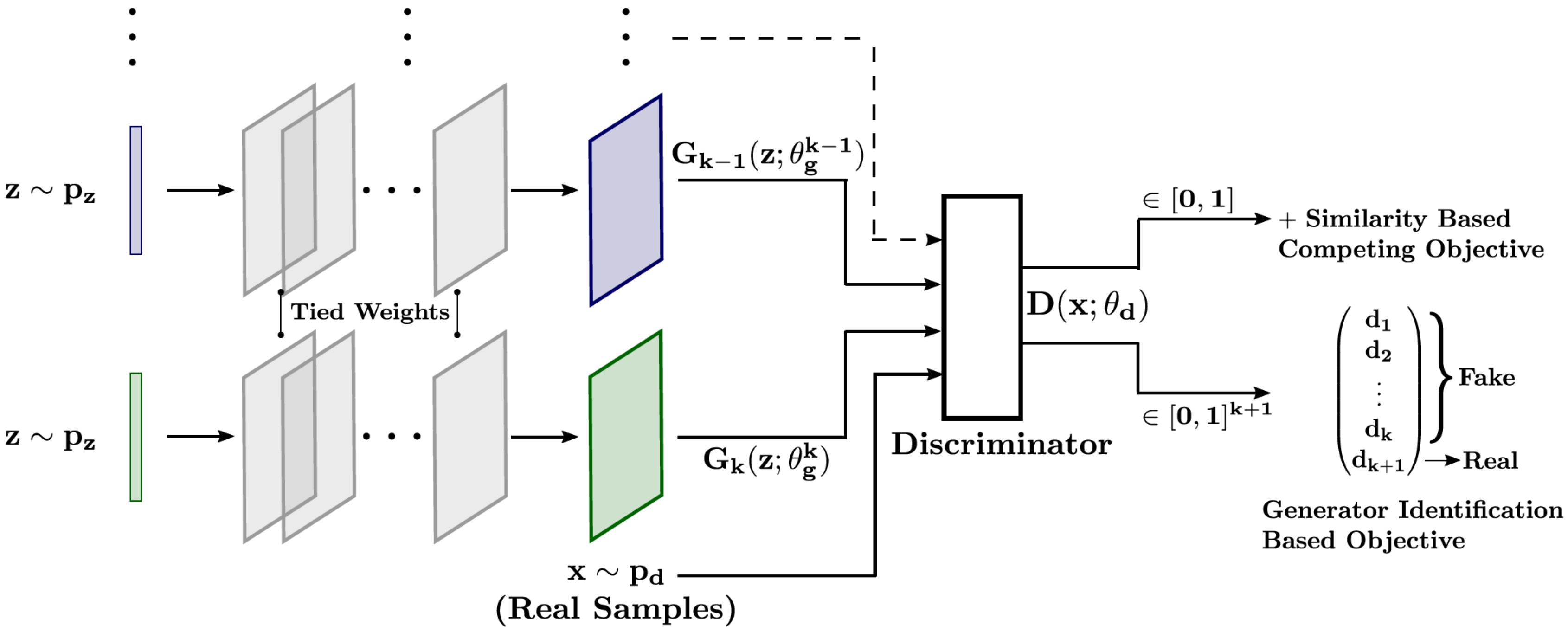
Fixed-batch BN in GAN

- Fix a *reference batch* $R=\{\boldsymbol{r}^{(1)}, \boldsymbol{r}^{(2)}, .., \boldsymbol{r}^{(m)}\}$
- Given new inputs $\boldsymbol{X}=\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, .., \boldsymbol{x}^{(m)}\}$
- **Compute mean and standard deviation of features of *R***
  - Note that though *R* does not change, the feature values change when the parameters change
- Normalize the features of *X* using the mean and standard deviation from *R*
- **Every $\boldsymbol{x}^{(i)}$ is always treated the same, regardless of which other examples appear in the minibatch**

[Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, “Improved Techniques for Training GANs”, NIPS 2016]

Mode collapse

- GANs often seem to collapse to far fewer modes than the model can represent



- multiple parallel generators
- share parameters up to layer l
- applies a diversity loss on different generators
- Alternatively, have D predict which generator the fake sample came from!

[Ghosh et al. Multi-Agent Diverse Generative Adversarial Networks (2017)]

Can we combine the ideas of GAN and VAE?

VAE/GAN

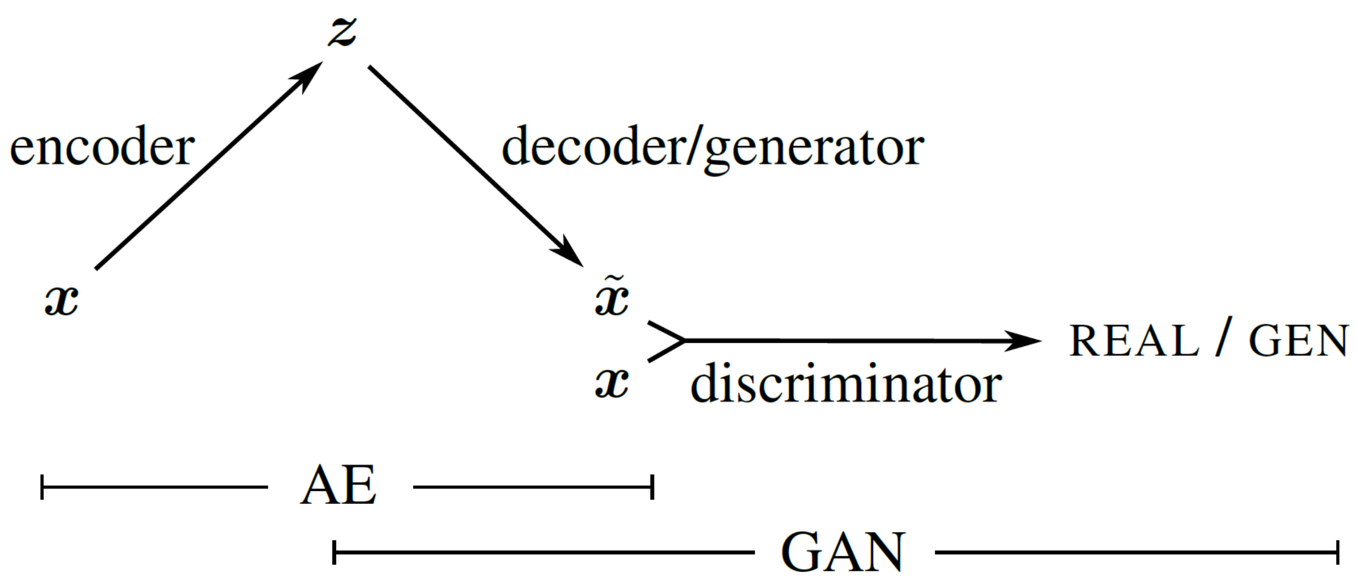
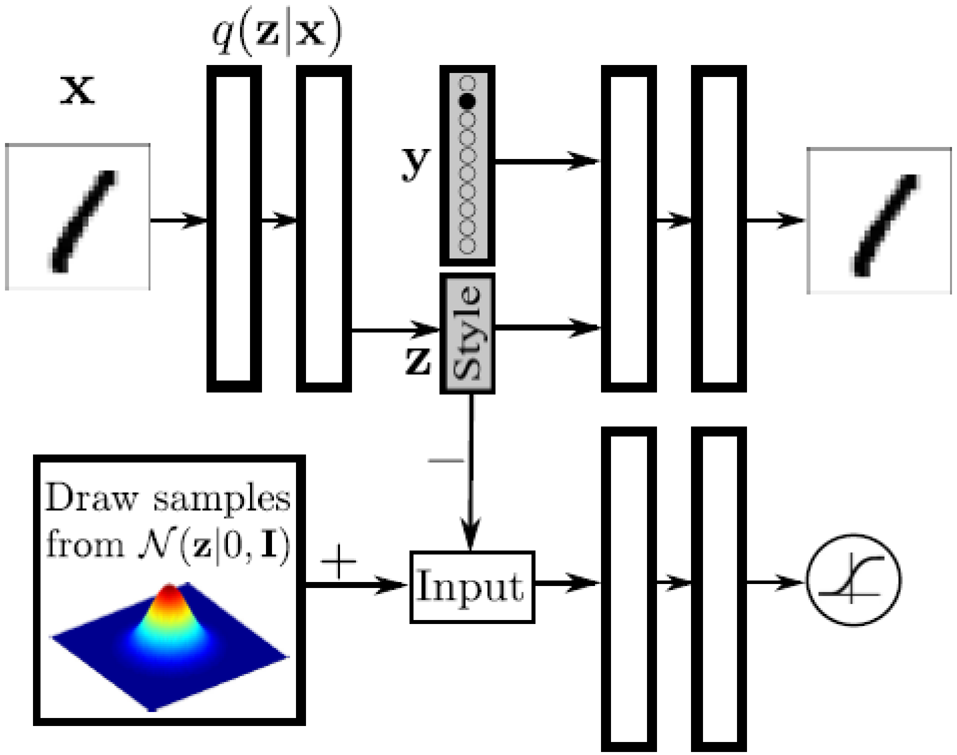


Figure 1. Overview of our network. We combine a VAE with a GAN by collapsing the decoder and the generator into one.

[Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, Ole Winther, "Autoencoding beyond pixels using a learned similarity metric", ICML 2016]

Adversarial AutoEncoder



[Makhzani et al. "Adversarial Autoencoders" ICLR 2016]

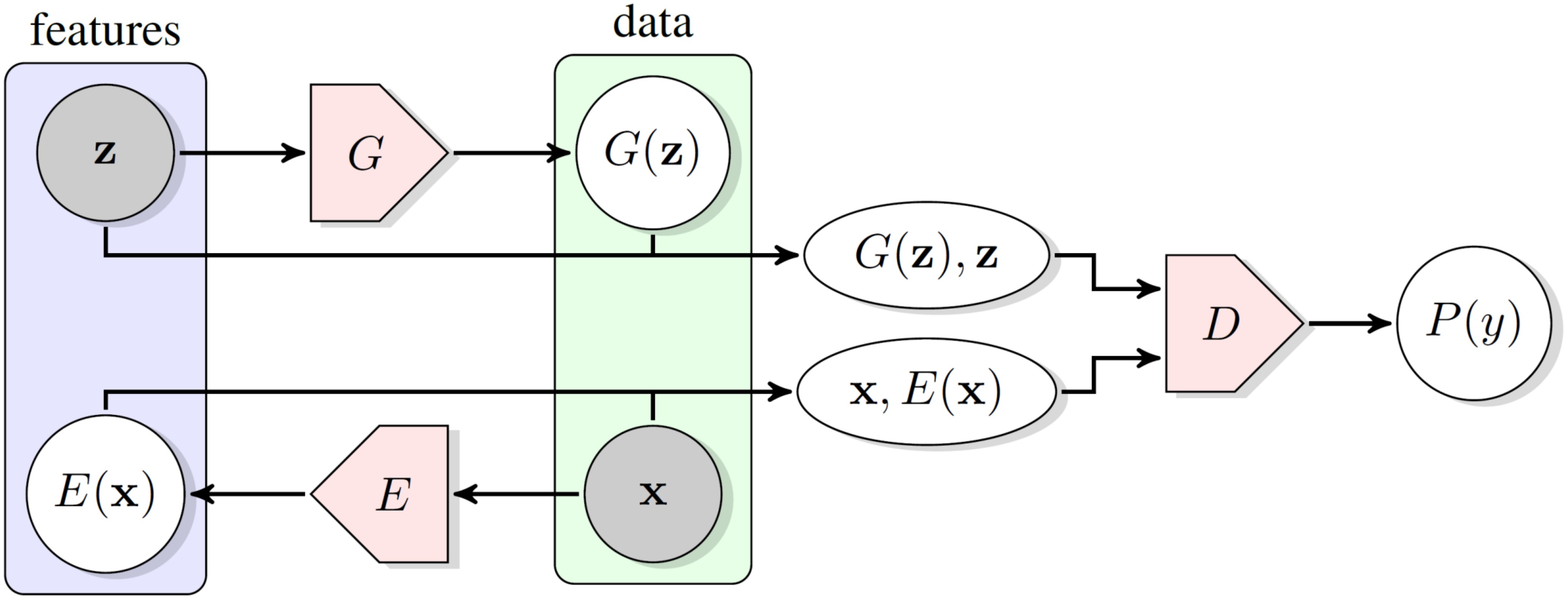
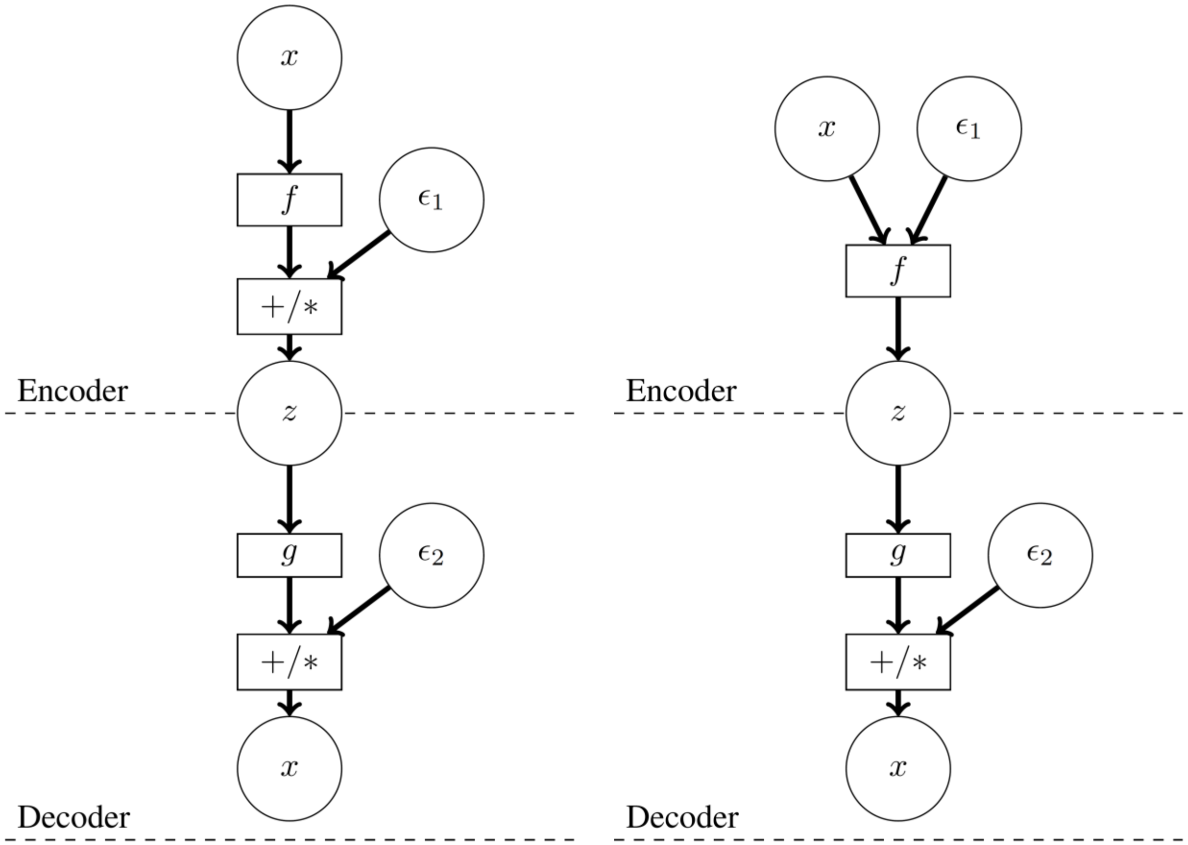


Figure 1: The structure of Bidirectional Generative Adversarial Networks (BiGAN).

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

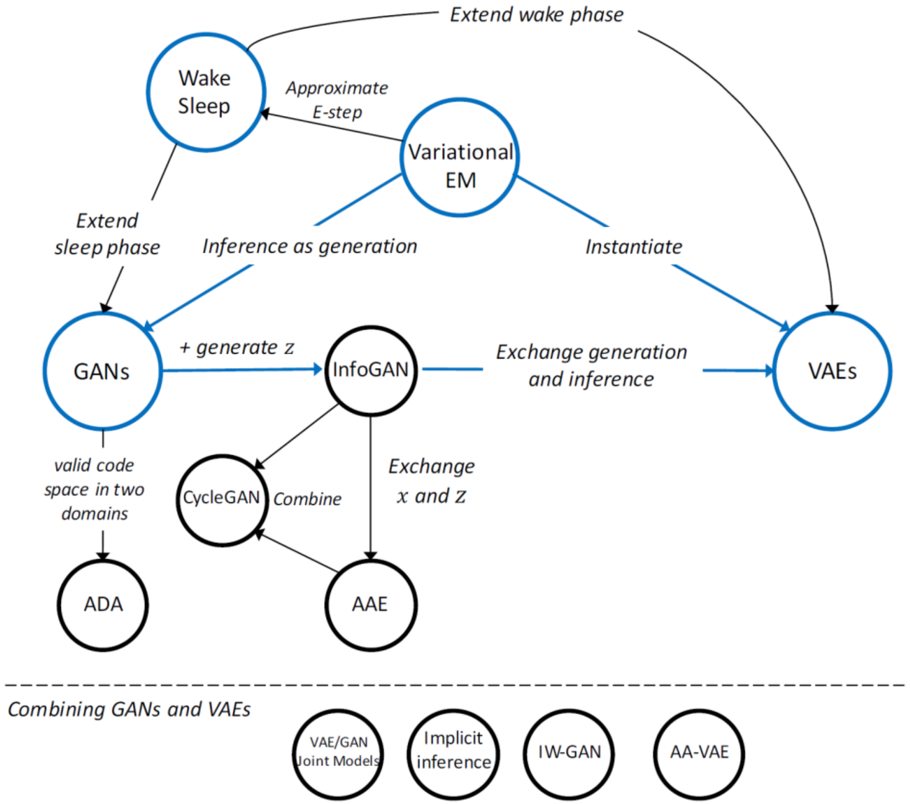
Adversarial VAE



(a) Standard VAE (b) Our model

[Lars Mescheder, Sebastian Nowozin, Andreas Geiger, "Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks", ICML 2018]

General Formulation



[Hu et al. “On Unifying Deep Generation Models”, ICLR 2018]

Outline

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

Other topics

- Adversarial Domain Adaptation
  - Coupled GAN [Liu&Tuzel "Coupled generative adversarial networks", NIPS 2016]
  - Cycle GAN [Zhu et al "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks" CVPR 2017]
  - UNIT [Liu et al. "Unsupervised Image to Image Translation Networks" NIPS 2017]
  - Multi Modal UNIT [Huang et al. Multimodal Unsupervised Image-to-Image Translation arXiv 2018]

- Pixel RNN [Oord et al. Pixel Recurrent Neural Networks ICML 2016]

Summary

- Variational AutoEncoder
  - Pretty principled
  - Efficient learning and inference in Sophisticated Bayesian Learning
  - Very useful codes
  - But slightly blurry
- GANs
  - Revolutionary idea
  - Sharp images
  - More difficult to optimize
  - Useful code
- Pixel RNN
  - Very simple and stable training process (softmax loss)
  - Relatively Slow compared to other approaches
  - Codes not so useful
- Combine or new Ideas!