Deep Generative Models

Sebastian Nowozin
Microsoft Research

x
f
y

ϵ
f
y

x
ϵ
f
y

Taxonomy

Non-probabilistic

Probabilistic

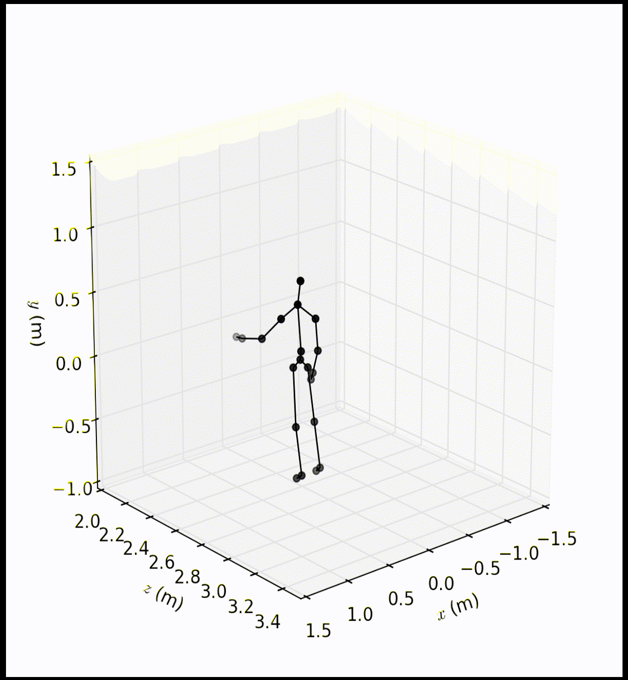
Generative

Discriminative

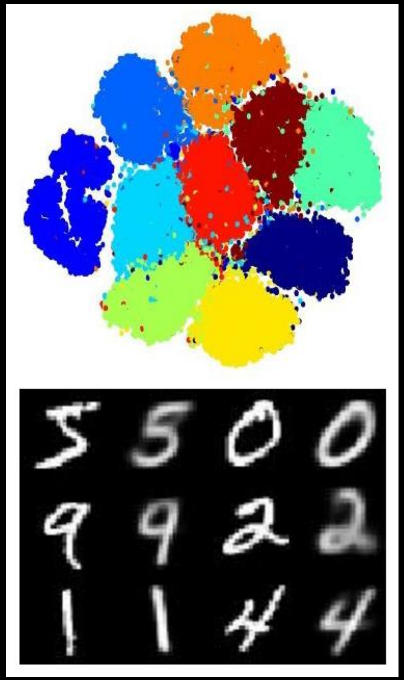
"Conditionally Generative"

ϵ
f
y

NVIDIA's Progressive GANs [Karras et al., 2018]

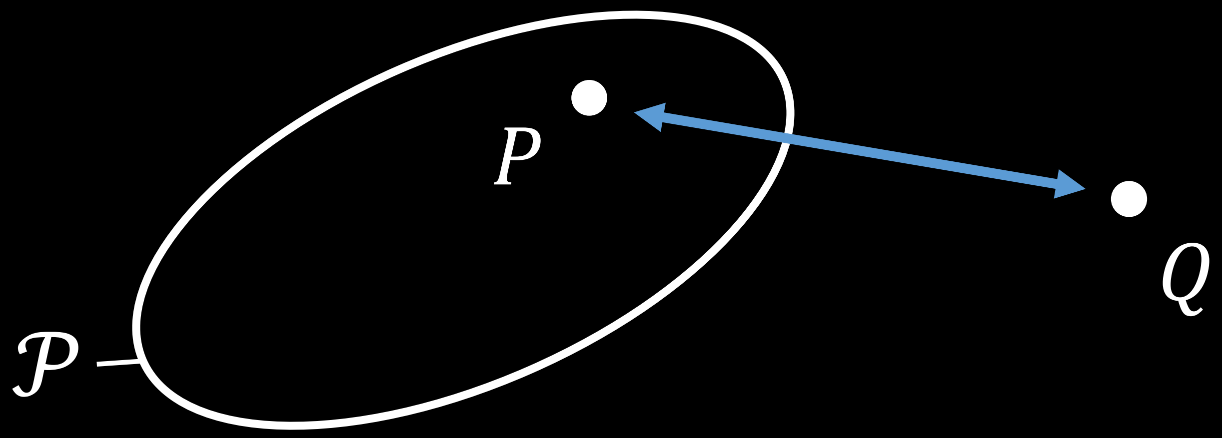


Generating non-image data

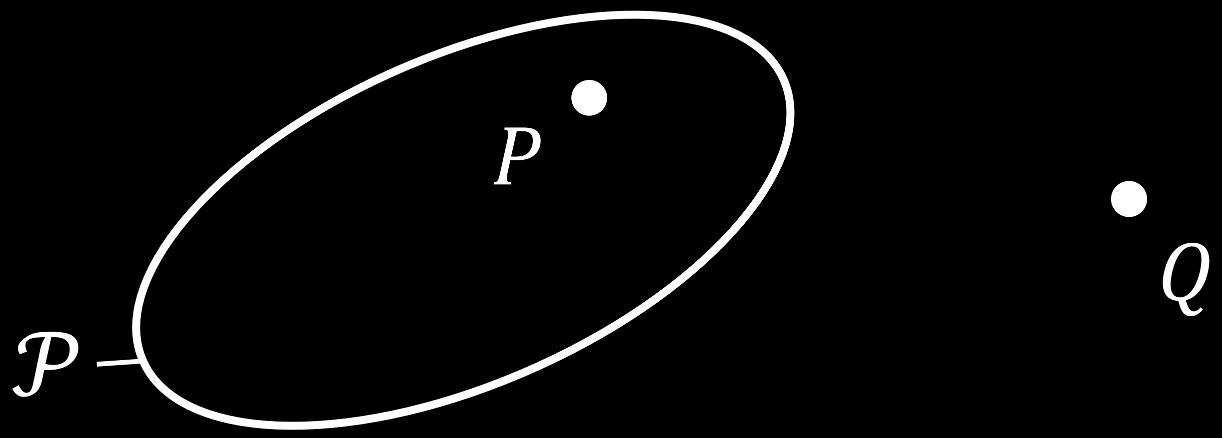


Representation learning
[Zheng et al., 2018]

Learning Probabilistic Models



Learning Probabilistic Models



Assumptions on *P*:

- tractable sampling
- tractable parameter gradient with respect to sample
- tractable likelihood function

Principles of Density Estimation

Integral Probability Metrics
[Müller, 1997]
[Sriperumbudur et al., 2010]

$$\gamma_{\mathcal{F}}(P, Q) = \sup_{f \in \mathcal{F}} \left| \int f \mathrm{d}P - \int f \mathrm{d}Q \right|$$

- Kernel MMD
- Wasserstein GANs

Proper scoring rules
[Gneiting and Raftery, 2007]

$$S(P, Q) = \int S(P, x) \mathrm{d}Q(x)$$

- Variational Autoencoders
- Autoregressive models
- DISCO networks

f-divergences
[Ali and Silvey, 1966],
[Nguyen et al., 2010]

$$D_f(P \parallel Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x$$

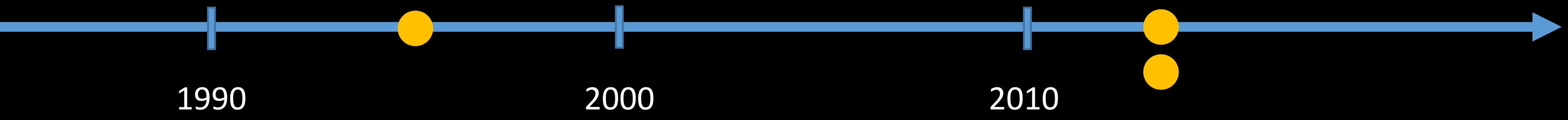
- Generative adversarial networks
- f-GAN, b-GAN

Variational Autoencoders (VAE)

[Kingma and Welling, 2014], [Rezende et al., 2014]

ϵ
f
y

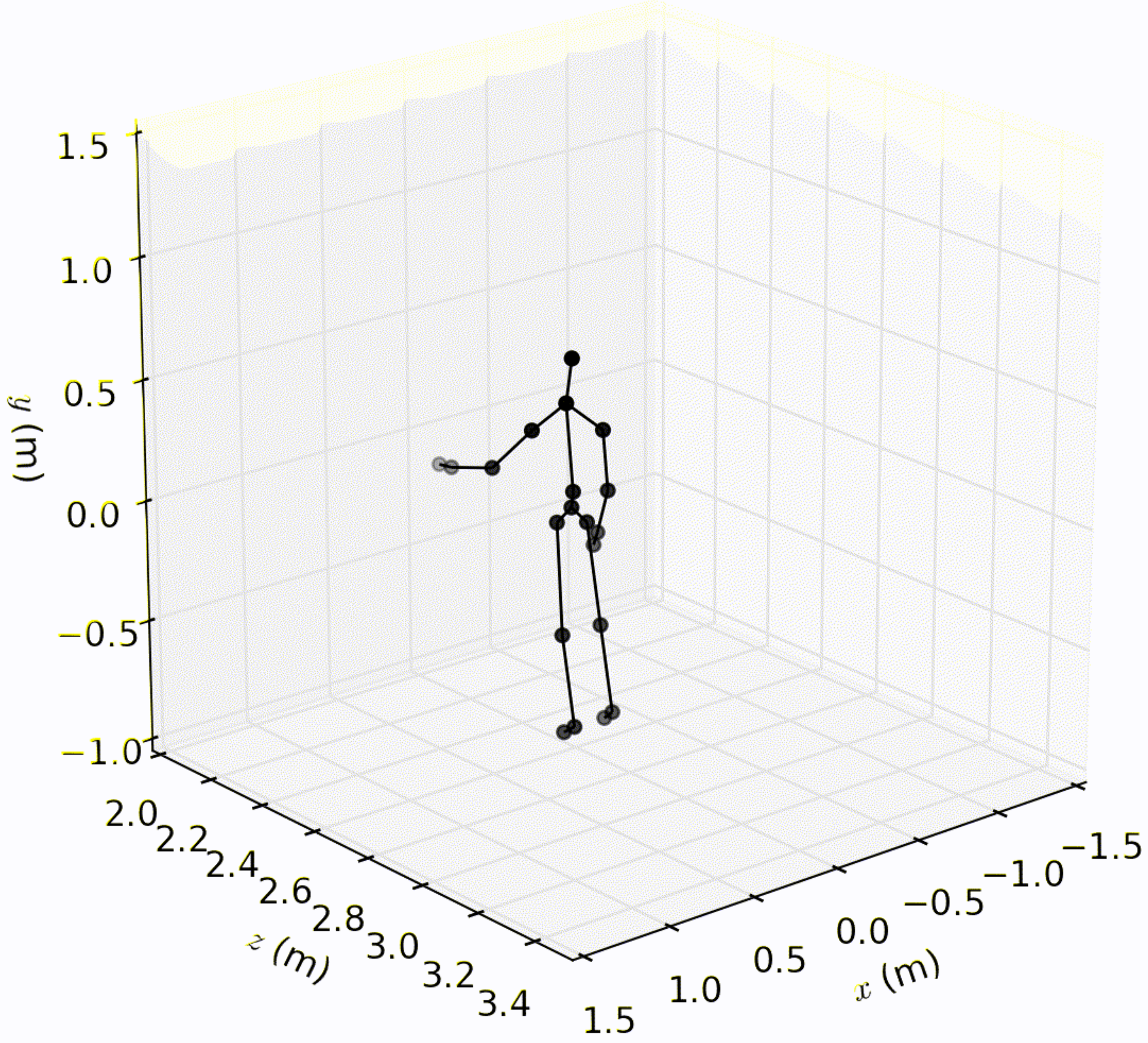
Variational Autoencoders



1. Dayan et al. (1995). The Helmholtz machine. *Neural Computation*
2. Kingma and Welling (2014). Auto-encoding Variational Bayes. NIPS
3. Rezende et al. (2014). Stochastic backpropagation and approximate inference in deep generative models. *ICML*

```python
def encode(self, x):
    h = F.crelu(self.qlin0(x))
    h = F.crelu(self.qlin1(h))
    h = F.crelu(self.qlin2(h))
    h = F.crelu(self.qlin3(h))

    self.qmu = self.qlin_mu(h)
    self.qln_var = self.qlin_ln_var(h)

def decode(self, z):
    h = F.crelu(self.plin0(z))
    h = F.crelu(self.plin1(h))
    h = F.crelu(self.plin2(h))
    h = F.crelu(self.plin3(h))

    self.pmu = self.plin_mu(h)
    self.pln_var = self.plin_ln_var(h)

def __call__(self, x):
    # Compute q(z|x)
    self.encode(x)

    self.kl = gaussian_kl_divergence(self.qmu, self.qln_var)
    self.logp = 0
    for j in xrange(self.num_zsamples):
        # z ~ q(z|x)
        z = F.gaussian(self.qmu, self.qln_var)

        # Compute p(x|z)
        self.decode(z)

        # Compute objective
        self.logp += gaussian_logp(x, self.pmu, self.pln_var)

    self.logp /= self.num_zsamples
    self.obj = self.kl - self.logp

    return self.obj
```

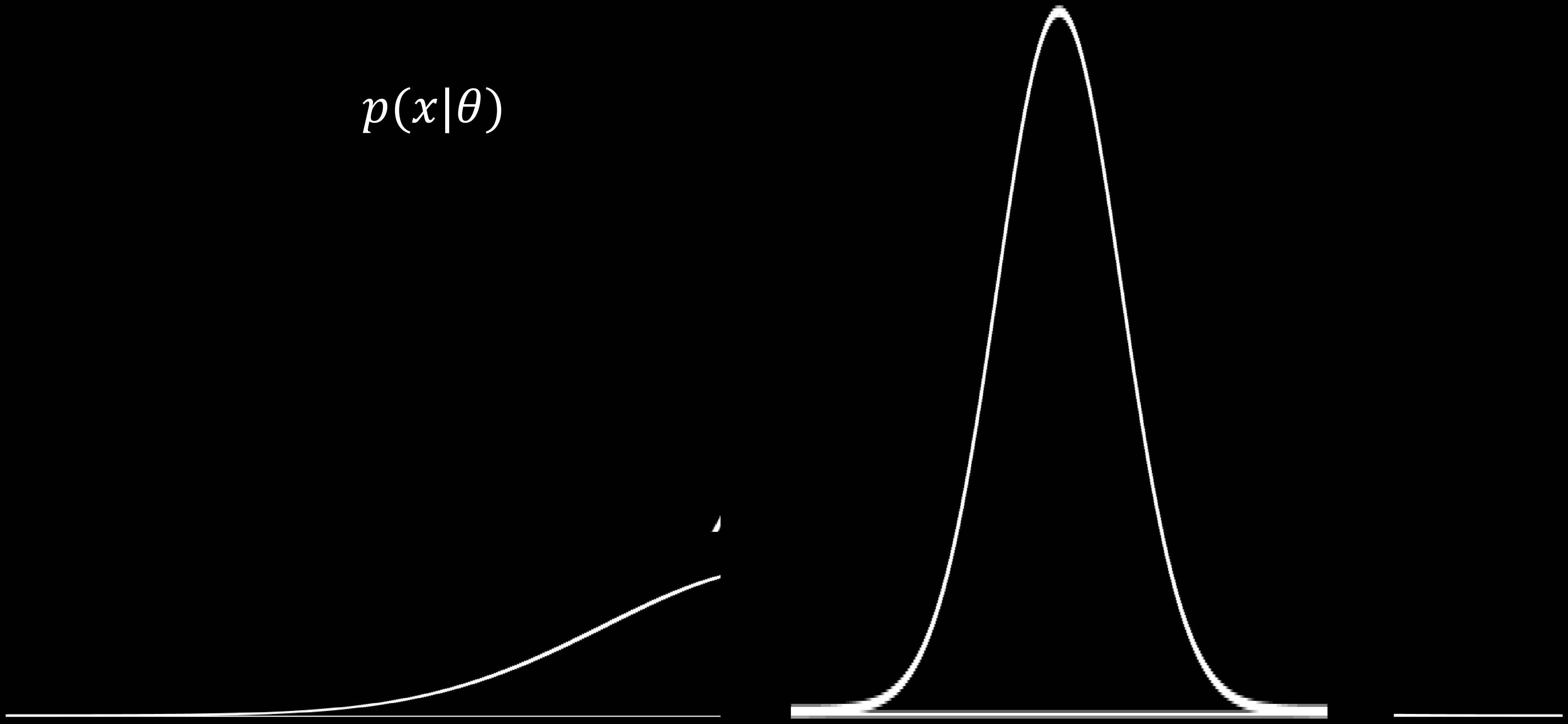
https://msrc-gitlab/senowozi/posevae/

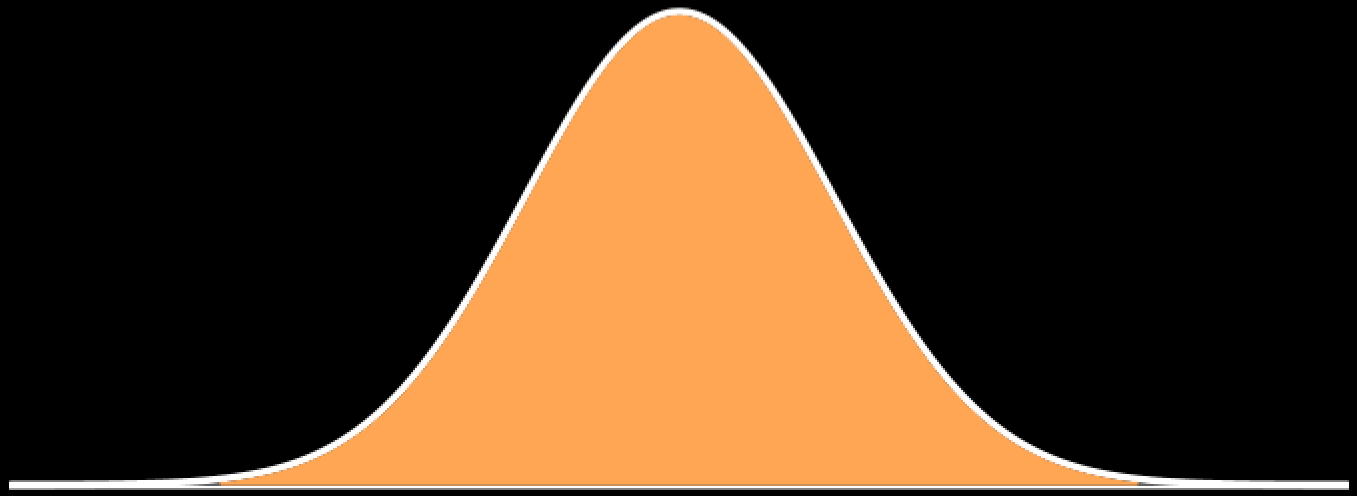


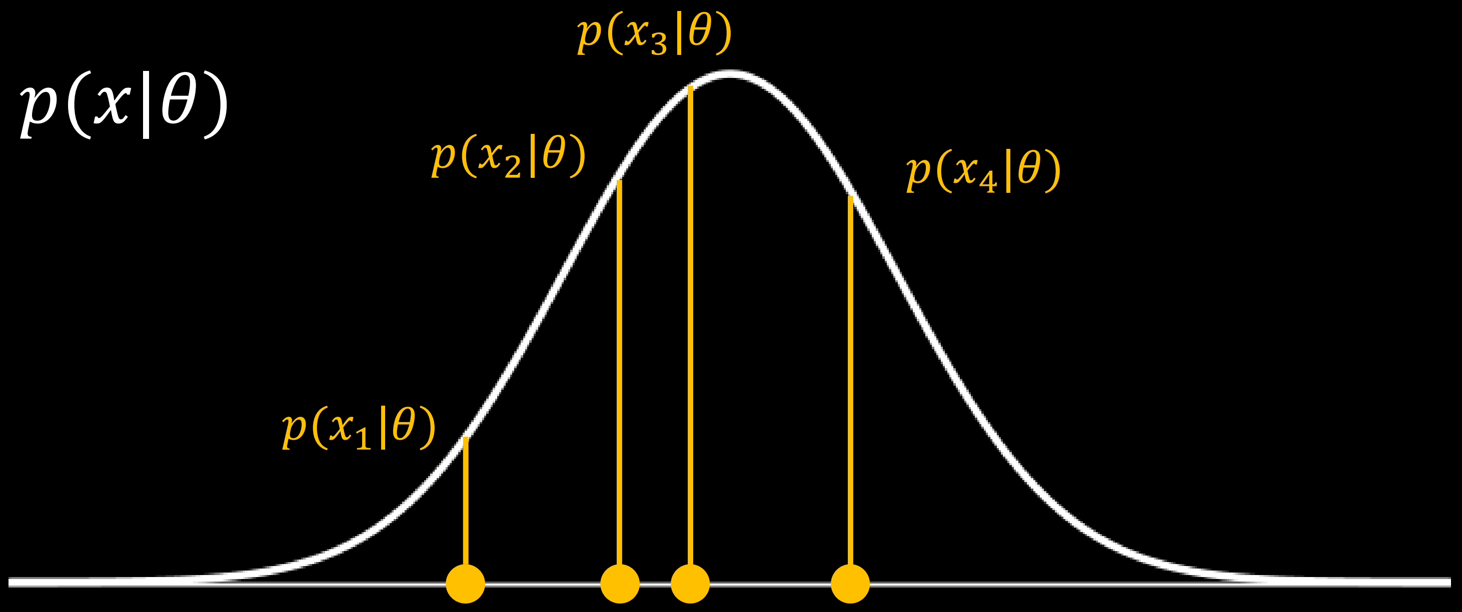
Maximum Likelihood Estimation (MLE)

[Fisher, 1929]

- MLE extremely successful:
  e.g. least squares and cross-entropy are MLE estimators


$p(x|\theta)$



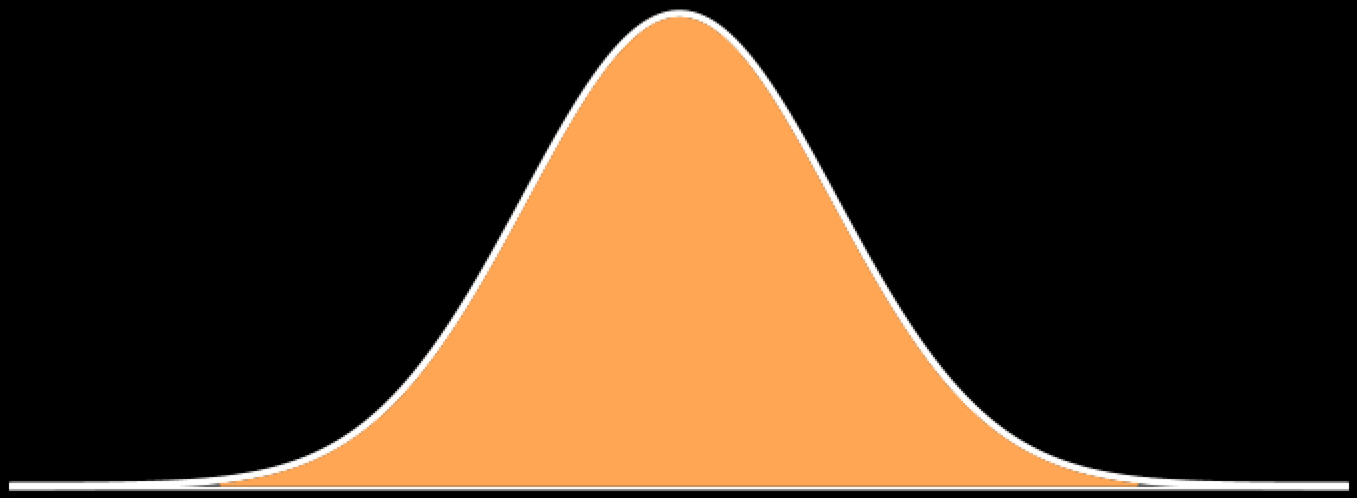


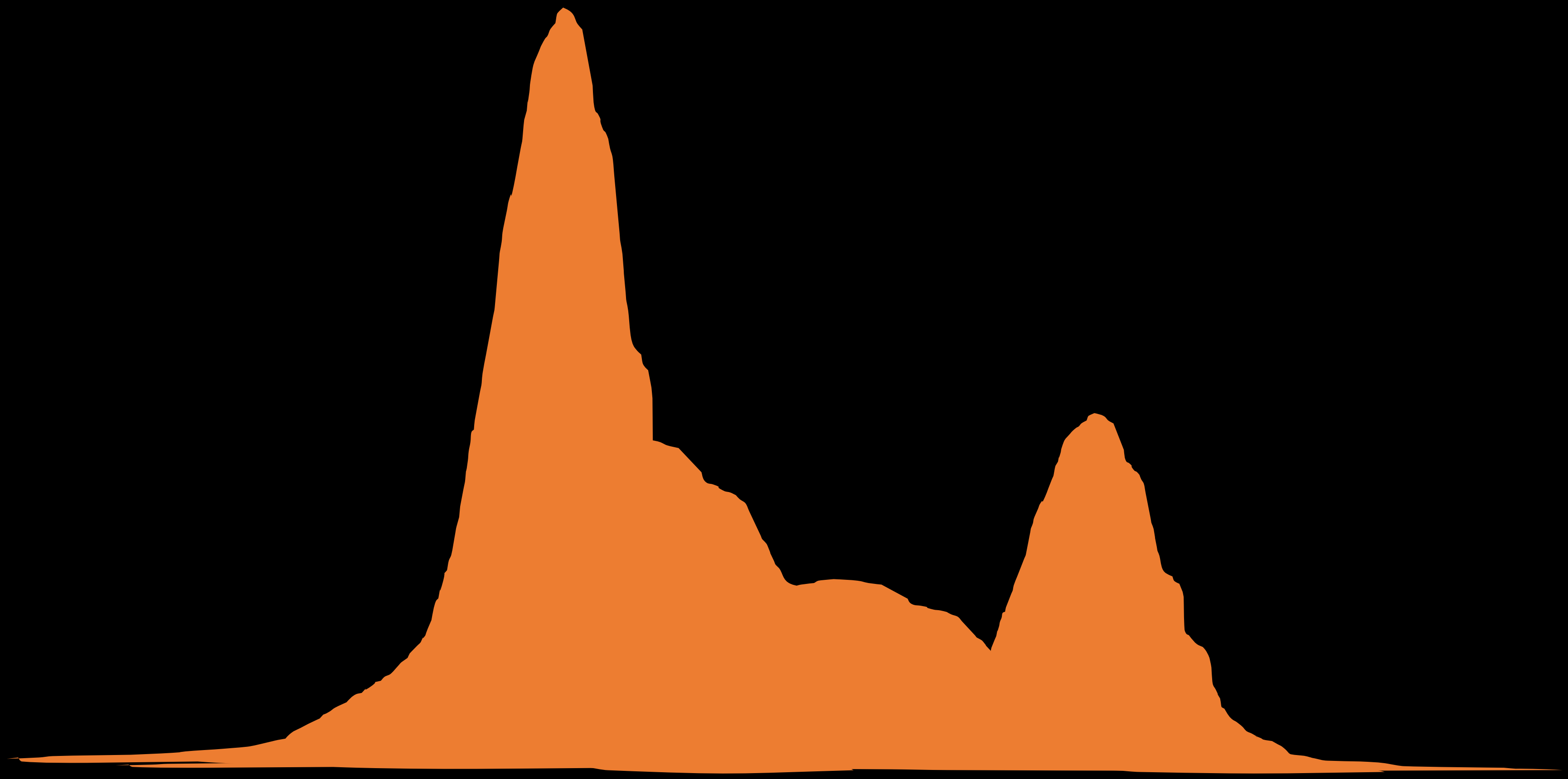
Maximize the *likelihood function*

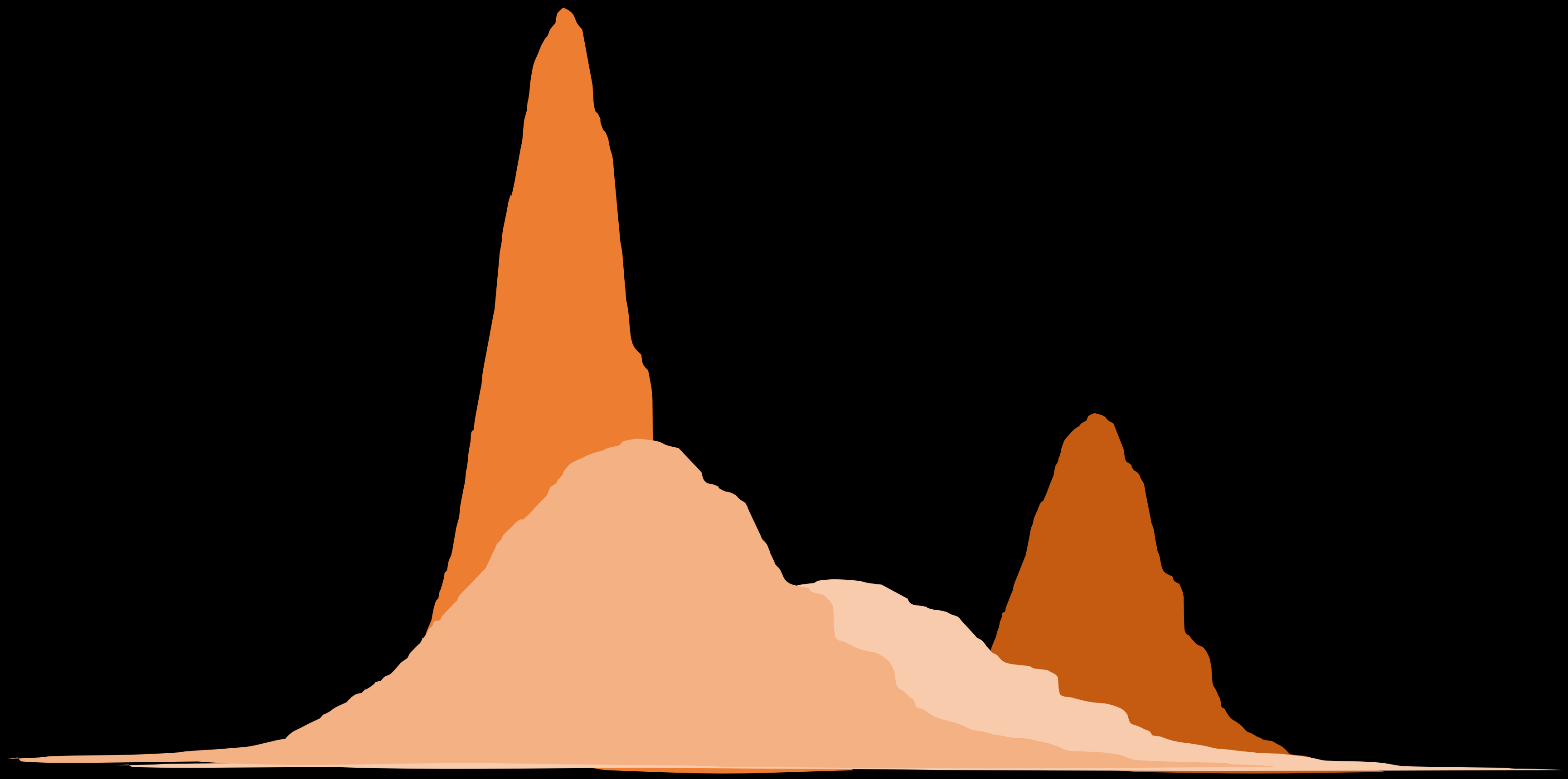
$$L(\theta) = \prod_i p(x_i|\theta)$$

$$\log L(\theta) = \log \prod_i p(x_i|\theta)$$

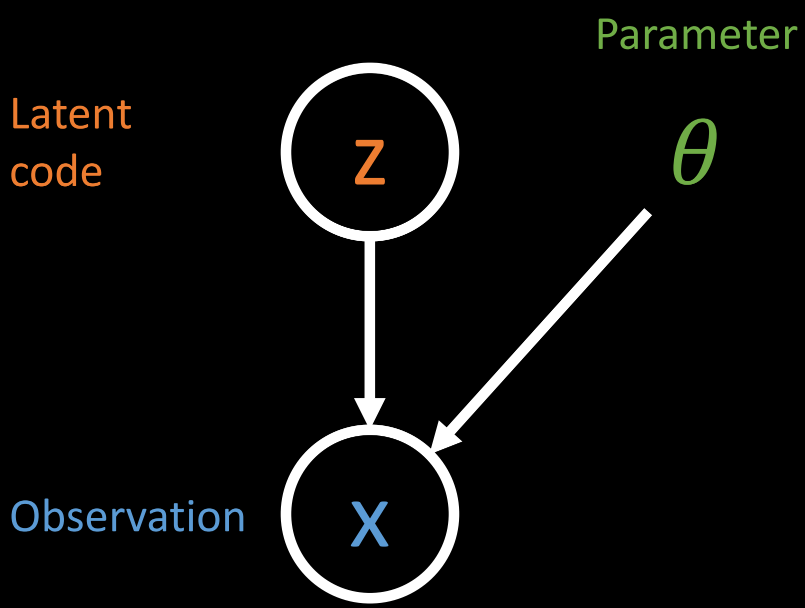
$$\log L(\theta) = \sum_i \log p(x_i|\theta)$$







VAE: Model



$$p(x|\theta) = \int \mathrm{p}(\mathrm{x}|\mathrm{z}, \theta)\, p(z)\mathrm{d}z$$

Example

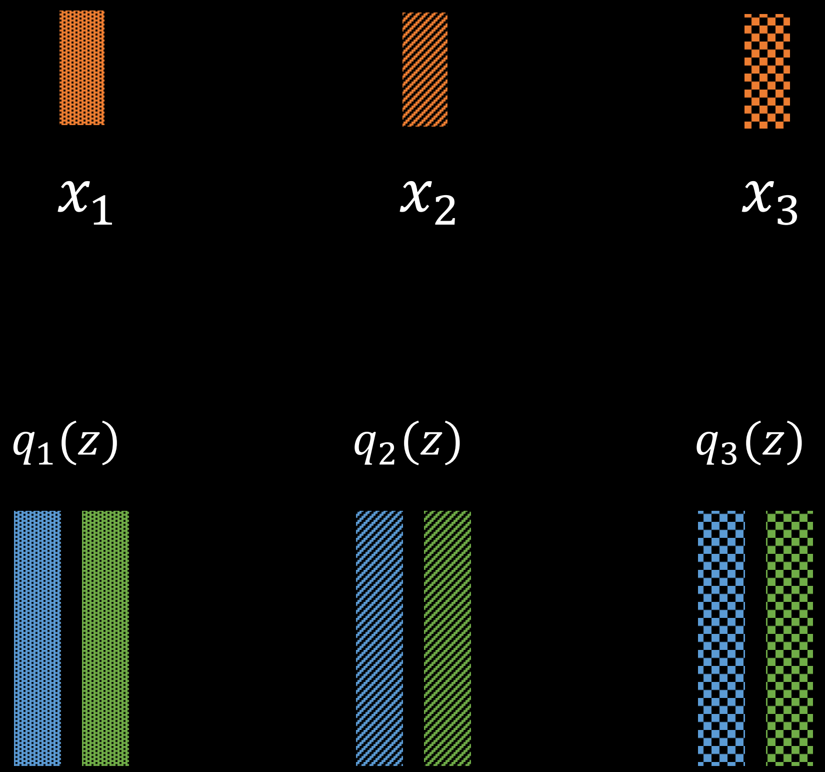
- $p(z)$ is a multivariate standard Normal
- $p(x|z, \theta)$ is a neural network outputting a simple distribution (e.g. diagonal Normal)

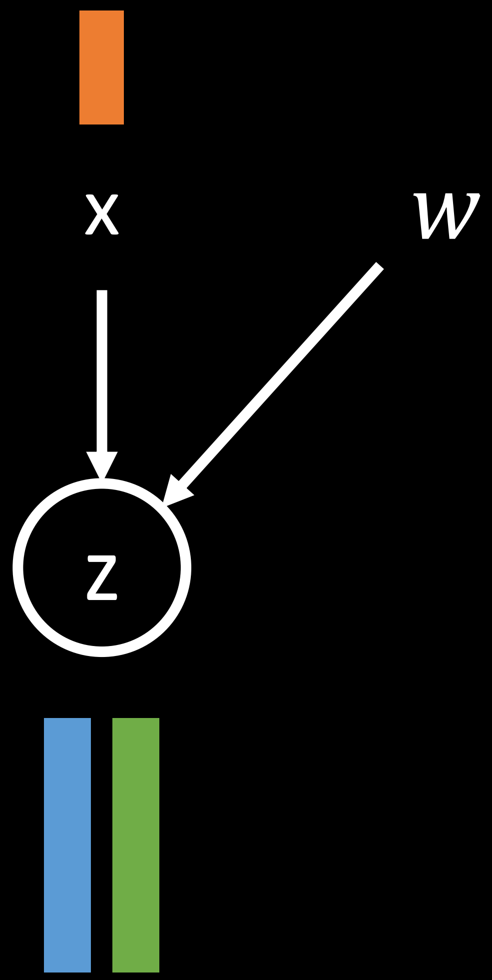
VAE: Maximum Likelihood Training

- Maximize the data log-likelihood, per-instance variational approximation

$$
\begin{aligned}
\log p(x|\theta) &= \log \int \mathrm{p}(\mathrm{x}|\mathrm{z}, \theta) p(z) \mathrm{dz} \\
&= \log \int \mathrm{p}(\mathrm{x}|\mathrm{z}, \theta) \frac{q(z)}{q(z)} p(z)\, \mathrm{dz} \\
&= \log \int \mathrm{p}(\mathrm{x}|\mathrm{z}, \theta) \frac{p(z)}{q(z)} q(z)\, \mathrm{dz} \\
&= \log \mathbb{E}_{z\sim q(z)}\left[\mathrm{p}(\mathrm{x}|\mathrm{z}, \theta) \frac{p(z)}{q(z)}\right] \\
&\geq \mathbb{E}_{z\sim q(z)}\left[\log \mathrm{p}(\mathrm{x}|\mathrm{z}, \theta) \frac{p(z)}{q(z)}\right] \\
&= \mathbb{E}_{z\sim q(z)}[\log \mathrm{p}(\mathrm{x}|\mathrm{z}, \theta)] - D_{\mathrm{KL}}(q(z) \parallel p(z))
\end{aligned}
$$

Inference networks





Inference networks

- Amortized inference [Stuhlmüller et al., NIPS 2013]
- Inference networks, recognition networks [Kingma and Welling, 2014]
- “Informed sampler” [Jampani et al., 2014]
- “Memory-based approach” [Kulkarni et al., 2015]

VAE: Maximum Likelihood Training

- Maximize the data log-likelihood, inference network variational approximation

$$\log p(x|\theta) = \log \int \mathrm{p}(\mathrm{x}|\mathrm{z},\theta) p(z)\mathrm{dz}$$

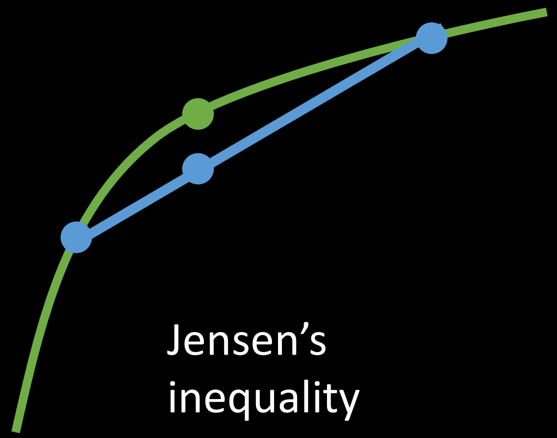
$$= \log \int \mathrm{p}(\mathrm{x}|\mathrm{z},\theta) \frac{q(z|x,w)}{q(z|x,w)} p(z)\, \mathrm{dz}$$

$$= \log \int \mathrm{p}(\mathrm{x}|\mathrm{z},\theta) \frac{p(z)}{q(z|x,w)} q(z|x,w)\, \mathrm{dz}$$

$$= \log \mathbb{E}_{z\sim q(z|x,w)} \left[\mathrm{p}(\mathrm{x}|\mathrm{z},\theta) \frac{p(z)}{q(z|x,w)}\right]$$

$$\geq \mathbb{E}_{z\sim q(z|x,w)} \left[\log \mathrm{p}(\mathrm{x}|\mathrm{z},\theta) \frac{p(z)}{q(z|x,w)}\right]$$

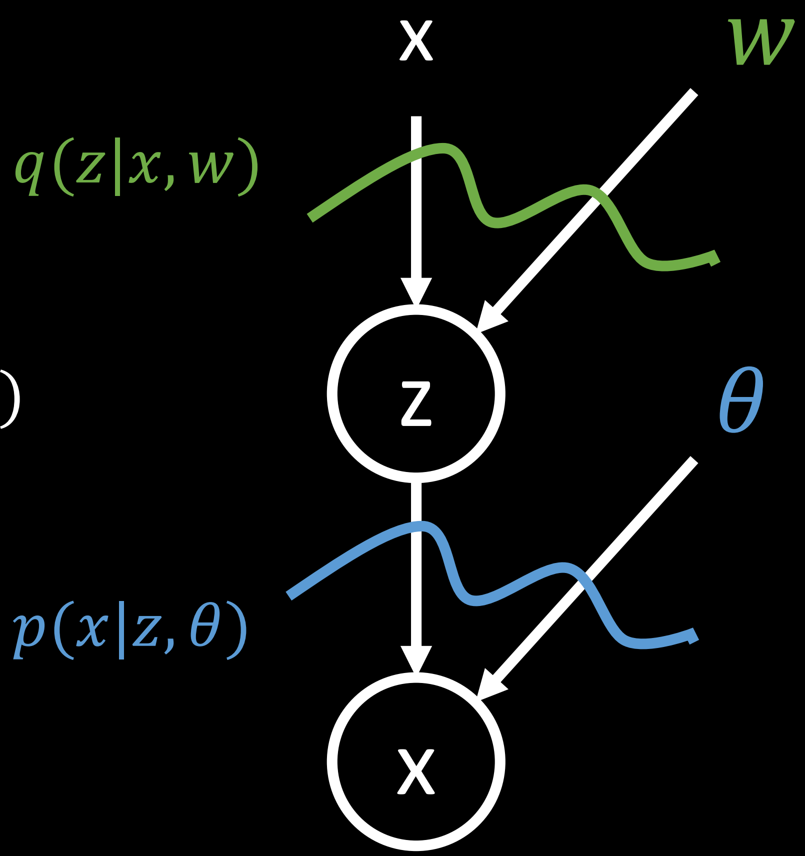
$$= \mathbb{E}_{z\sim q(z|x,w)} [\log \mathrm{p}(\mathrm{x}|\mathrm{z},\theta)] - D_{\mathrm{KL}}(q(z|x,w) \parallel p(z))$$

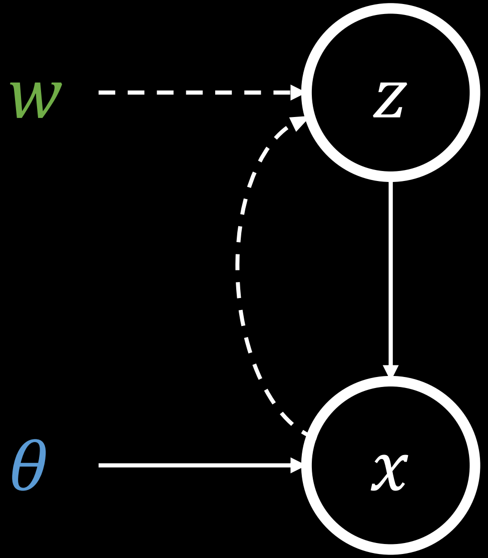


Jensen's inequality

Autoencoder viewpoint

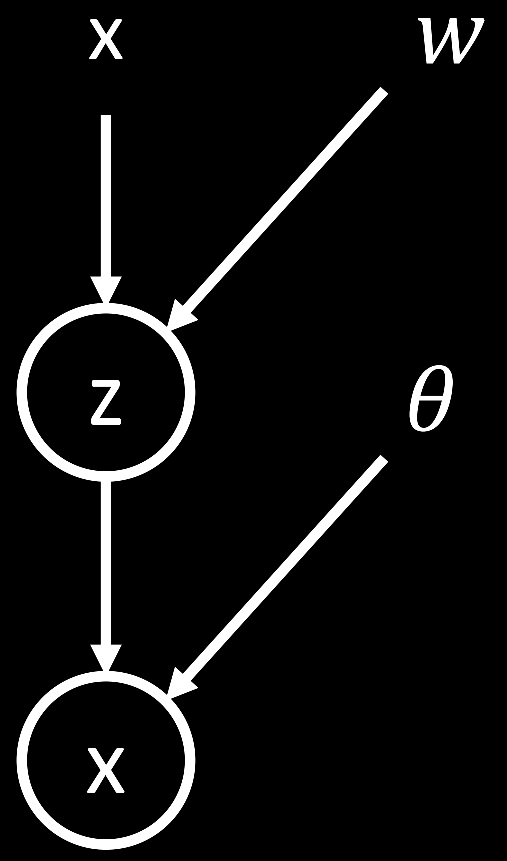
$$\max_{w,\theta} \; \mathbb{E}_{z\sim q(z|x,w)}[\log \mathrm{p}(\mathrm{x}|\mathrm{z},\theta)] - D_{\mathrm{KL}}(q(z|x,w) \parallel p(z))$$





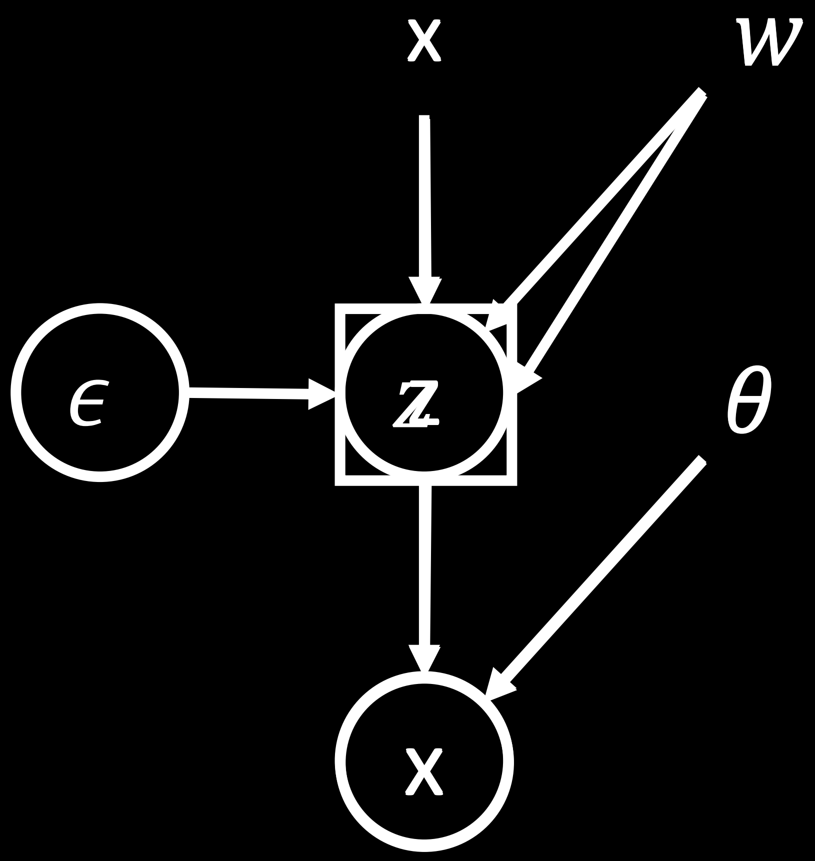
Reparametrization Trick

- [Rezende et al., 2014]
  [Kingma and Welling, 2014]



Reparametrization Trick

- [Rezende et al., 2014]
  [Kingma and Welling, 2014]
- Stochastic computation graphs
  [Schulman et al., 2015]



(Live Azure Demo)

https://notebooks.azure.com/nowozin/libraries/variational-autoencoder

Problems in VAEs

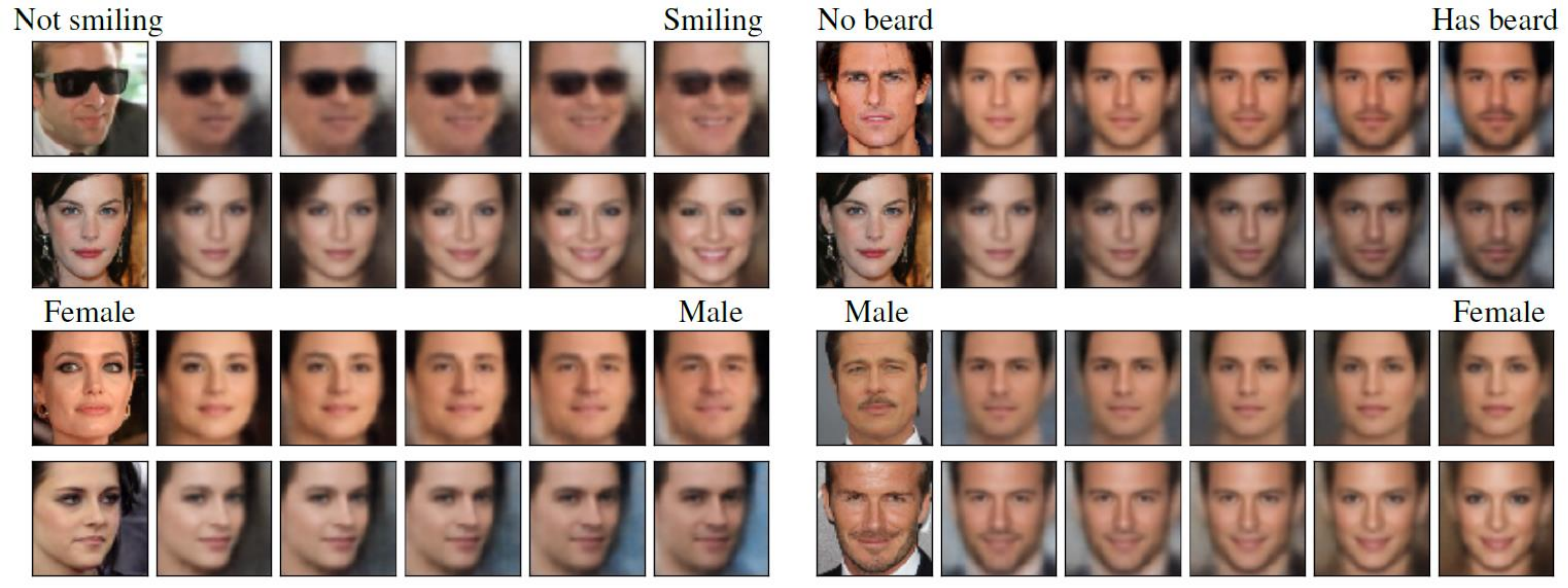
- Inadequate inference networks
  - Loose ELBO
  - Limits what the generative model can learn

- Parametric conditional likelihood assumptions
  - Limits the expressivity of the generative model
  - “Noise term has to explain too much”

- No control over latent representation that is learned

“Blurry images” in VAE models

from [Tulyakov et al., 2017]



Improving Inference Networks

- State of the art in inference network design:
  - NICE [Dinh et al., 2015]
  - Hamiltonian variational inference (HVI) [Salimans et al., 2015]
  - Importance weighted autoencoder (IWAE) [Burda et al., 2016]
  - Normalizing flows [Rezende and Mohamed, 2016]
  - Auxiliary deep generative networks [Maaløe et al., 2016]
  - Inverse autoregressive flow (IAF) [Kingma et al., NIPS 2016]
  - Householder flows [Tomczak and Welling, 2017]
  - Adversarial variational Bayes (AVB) [Mescheder et al., 2017]
  - Deep and Hierarchical Implicit Models [Tran et al., 2017]
  - Variational Inference using Implicit Distributions [Huszár, 2017]
  - Adversarial Message Passing for Graphical Models [Karaletsos, 2016]
  - Jackknife Variational Inference [Nowozin, 2018]

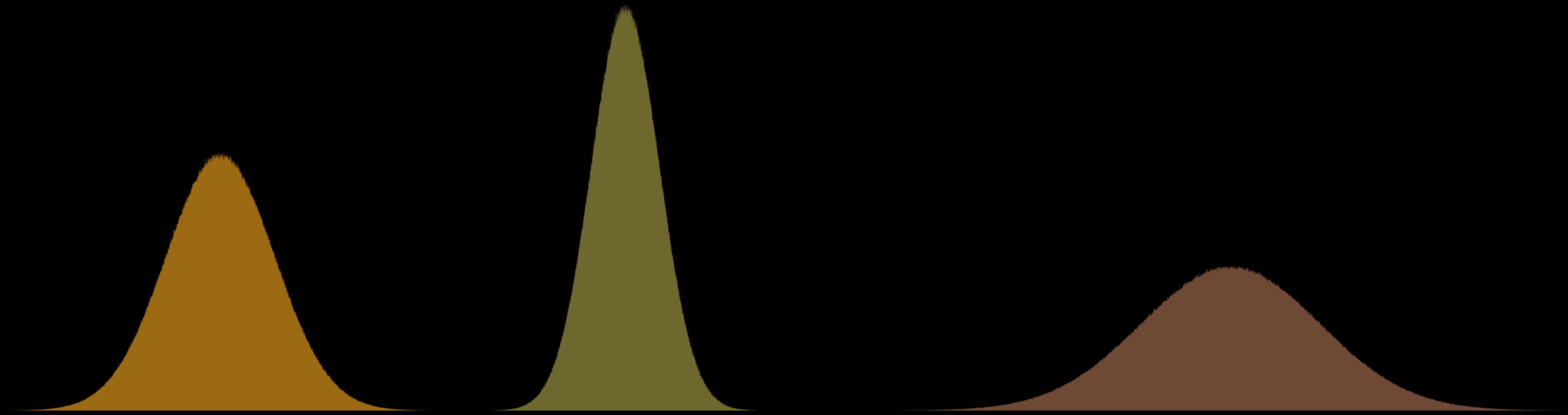
Generative Adversarial Networks

[Goodfellow et al., NIPS 2014], [Nowozin, Cseke, Tomioka, NIPS 2016]


ϵ
f
y

GAN = Implicit Models +
Estimation procedure

Classic parametric models

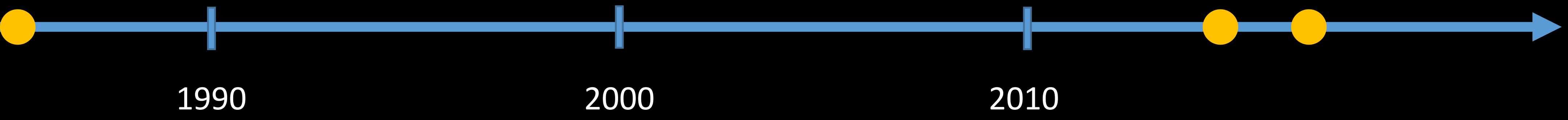


- Density function available
- Limited expressive power
- Mature field in statistics and learning theory

Implicit Model / Neural Sampler / Likelihood-free Model

- Highly expressive model class
- Density function not defined or intractable
- Lack of theory and learning algorithms
- Basis for generative adversarial networks (GANs)

Implicit Models

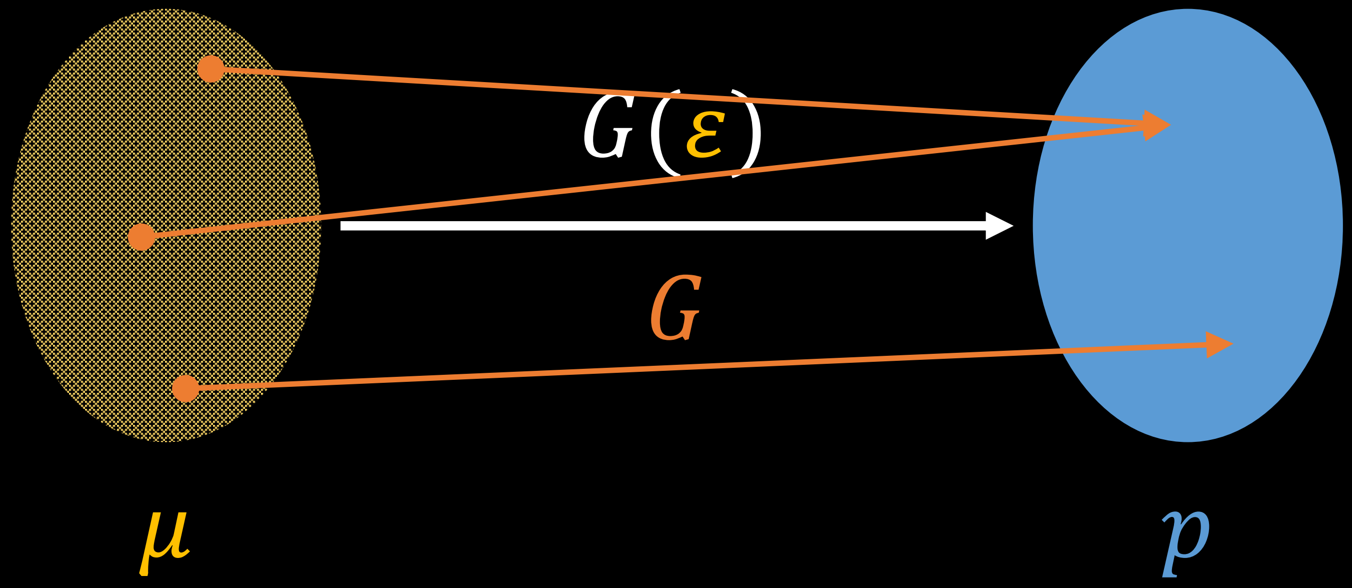


1. Diggle and Gratton (1984). Monte Carlo methods of inference for implicit statistical models. *JRSS B*
2. Goodfellow et al. (2014). Generative Adversarial Nets. NIPS
3. Mohamed and Lakshminarayanan (2016). Learning in Implicit Generative Models. *arXiv:1610.03483*

Implicit models as building blocks

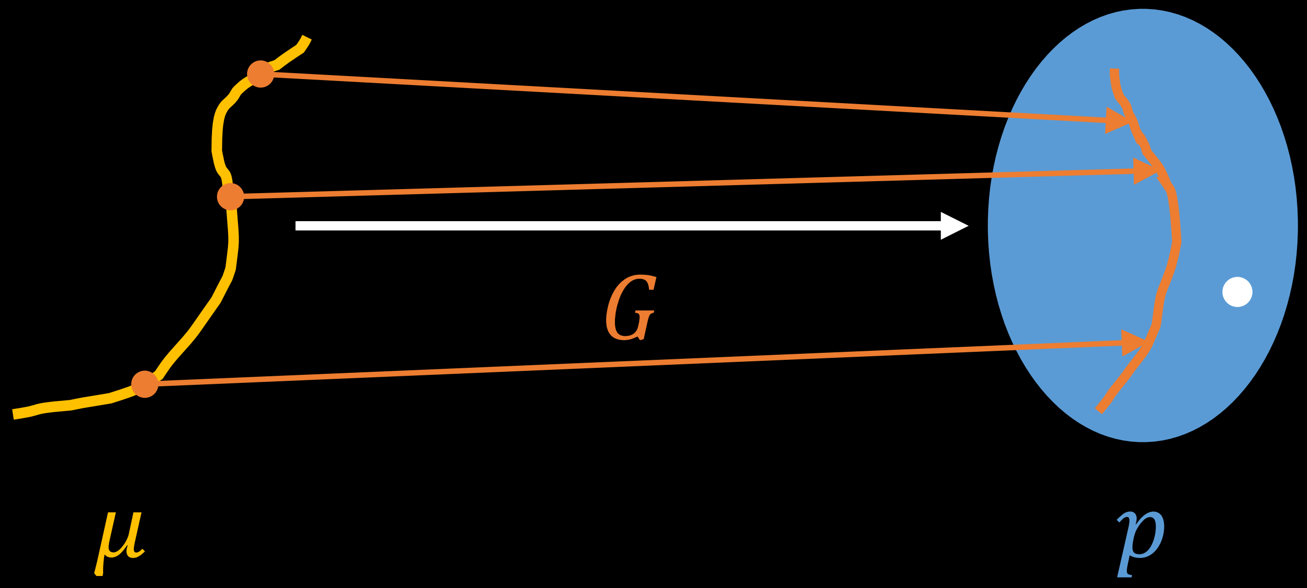
- For inference (as in AVB), or
- As model component, or
- As regularizer

Implicit Models: Problem 1



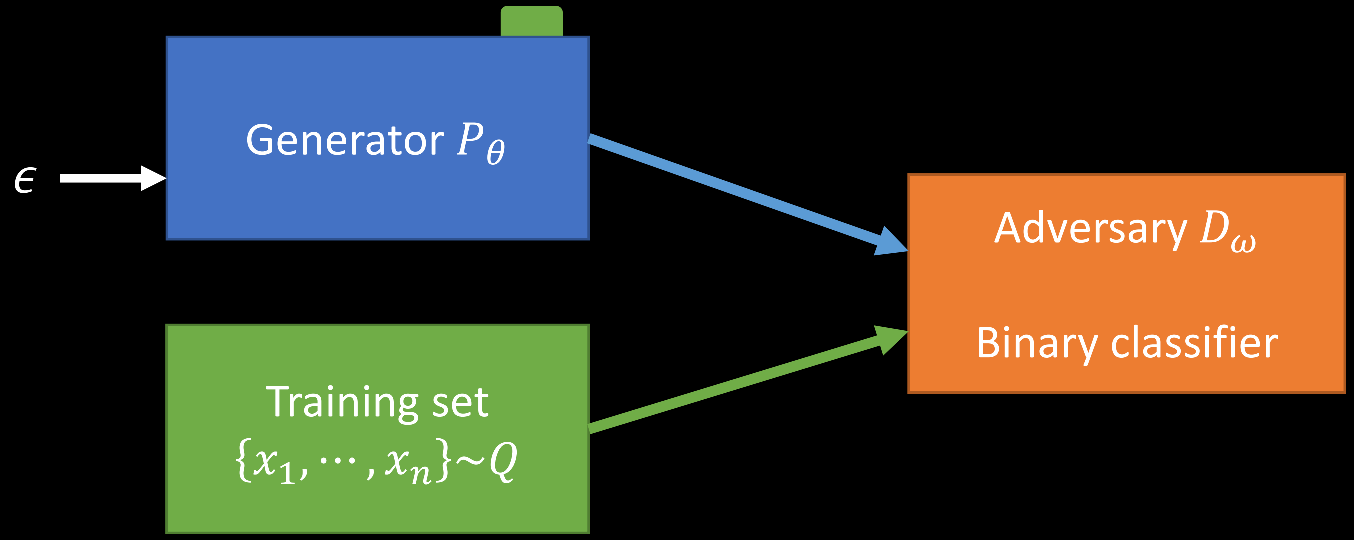
$$p(x) = \int_{z \in G^{-1}(x)} \mu(z)\,\mathrm{dz}$$

Implicit Models: Problem 2



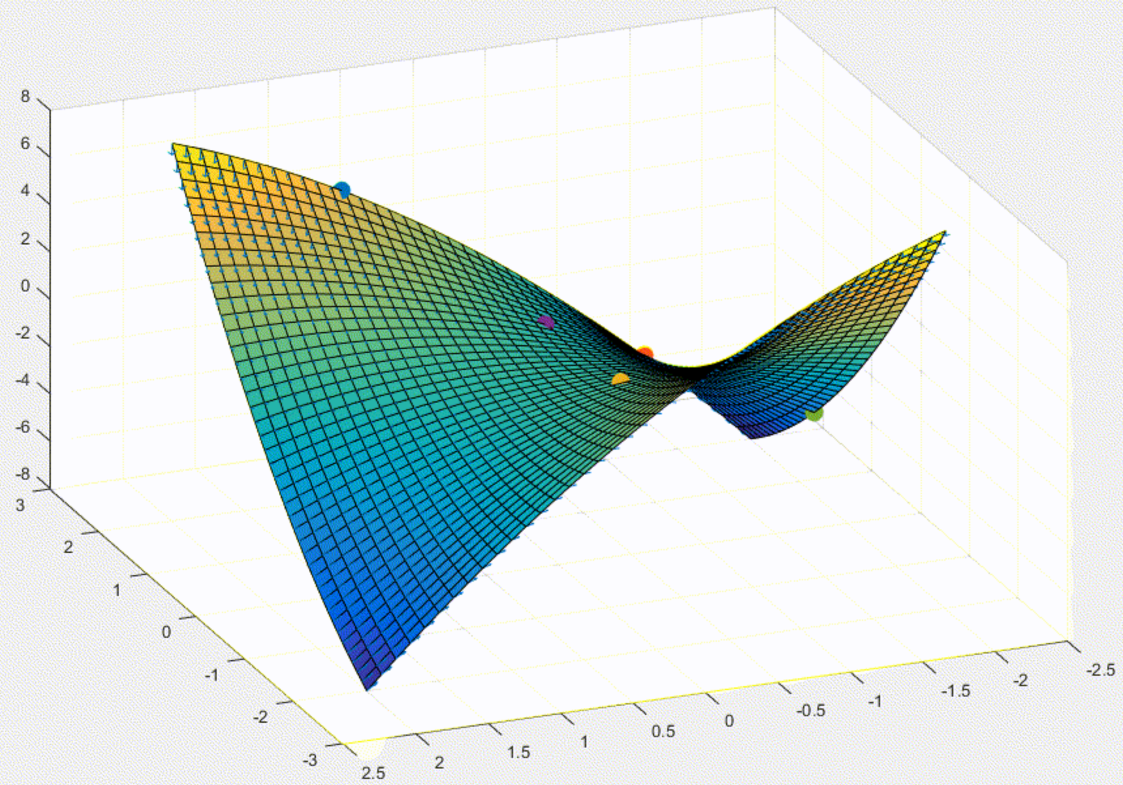
$p(x)$ not defined a.e.

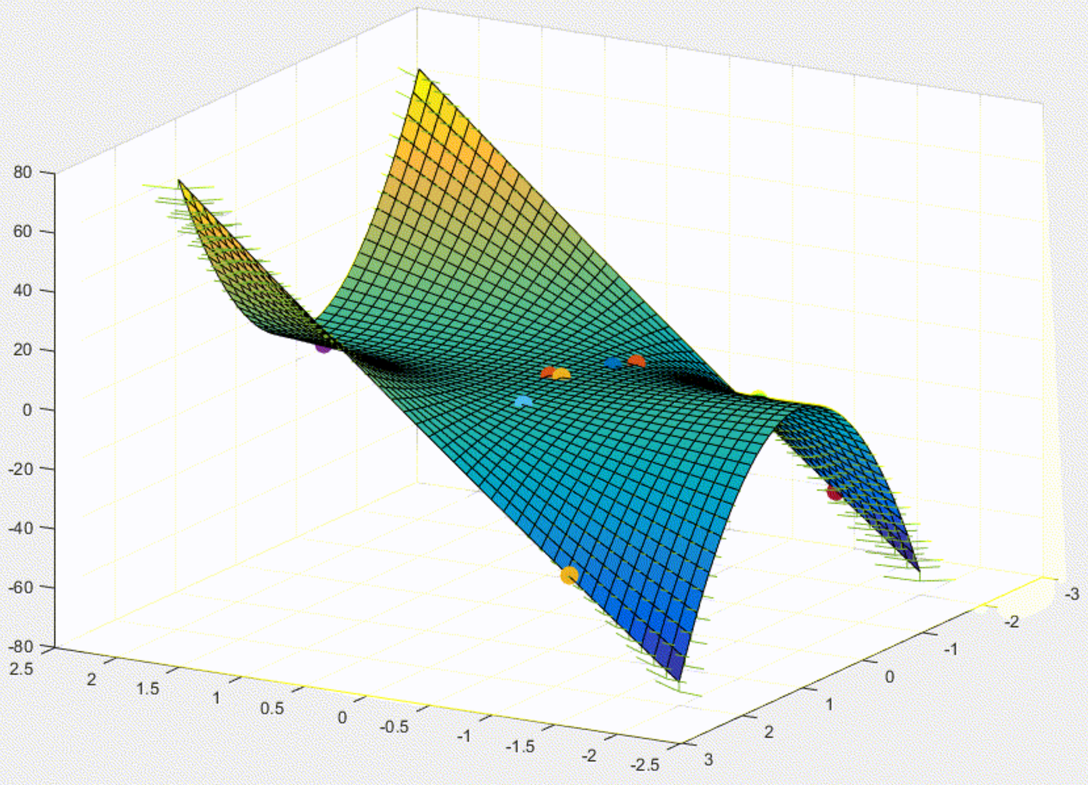
GAN Training Objective [Goodfellow et al., 2014]



- Generator tries to fool discriminator (i.e. generate realistic samples)
- Discriminator tries to distinguish fake from real samples
- Saddle-point problem

$$\min_\theta \max_\omega \mathbb{E}_{x \sim P_\theta}[\log D_\omega(x)] + \mathbb{E}_{x \sim Q}[\log(1 - D_\omega(x))]$$


8
6
4
2
0
-2
-4
-6
-8
3
2
1
0
-1
-2
-3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5


80
60
40
20
0
-20
-40
-60
-80
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
3
2
1
0
-1
-2
-3

Progress of GANs, 2013-2018

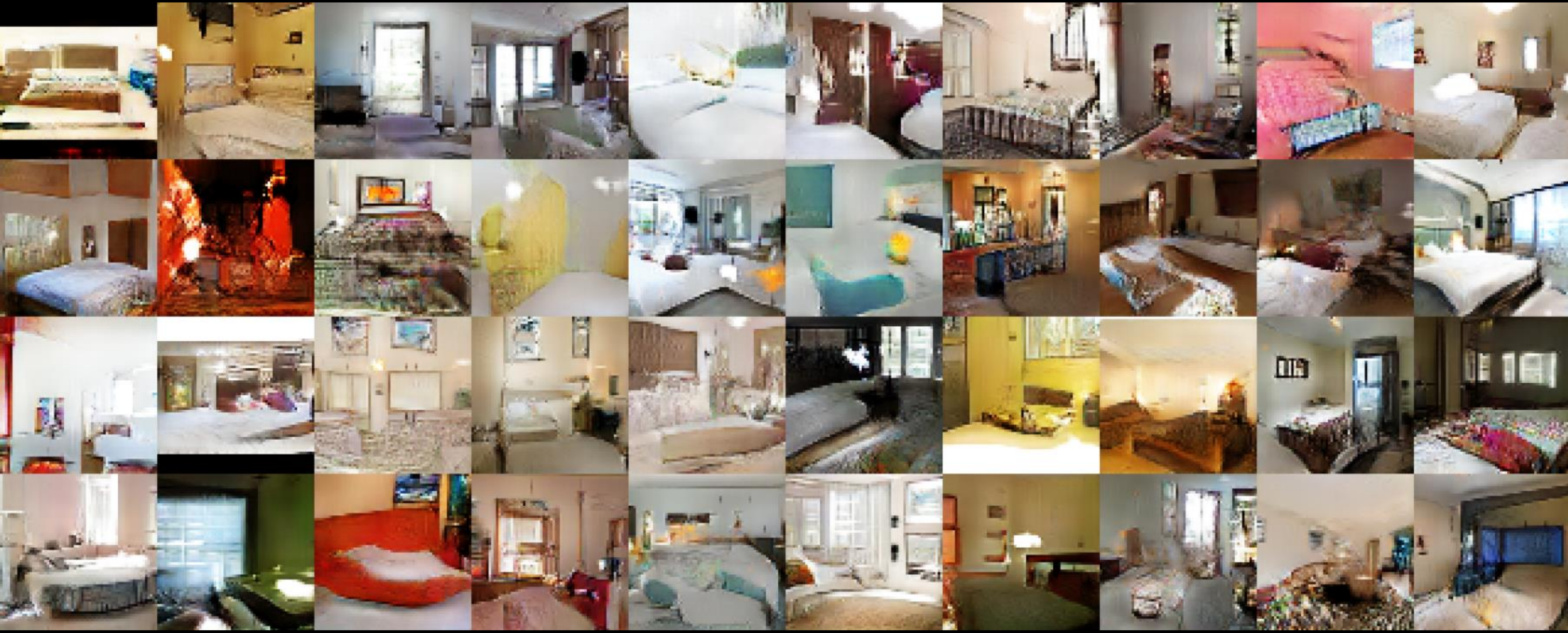
[Goodfellow et al., 2013]
University of Montreal

[Radford et al., 2015]
Facebook AI Research

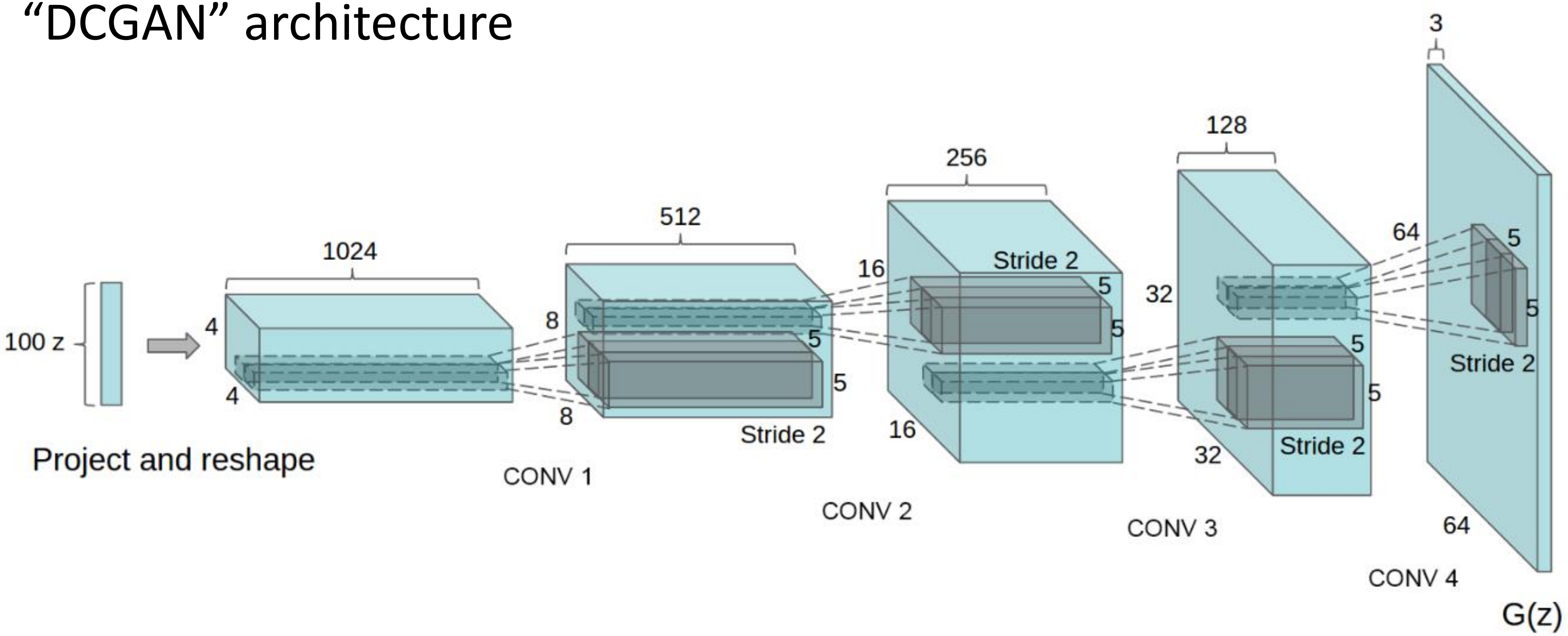
[Roth et al., 2017]
Microsoft and ETHZ

[Karras et al., 2018]
NVIDIA

Natural Images (Radford et al., 2015, arXiv:1511.06434)



“DCGAN” architecture



Linear interpolation in latent space [Radford et al., 2015]

Estimating f-divergences from samples

[Nguyen, Wainwright, Jordan, 2010]

- Divergence between two distributions

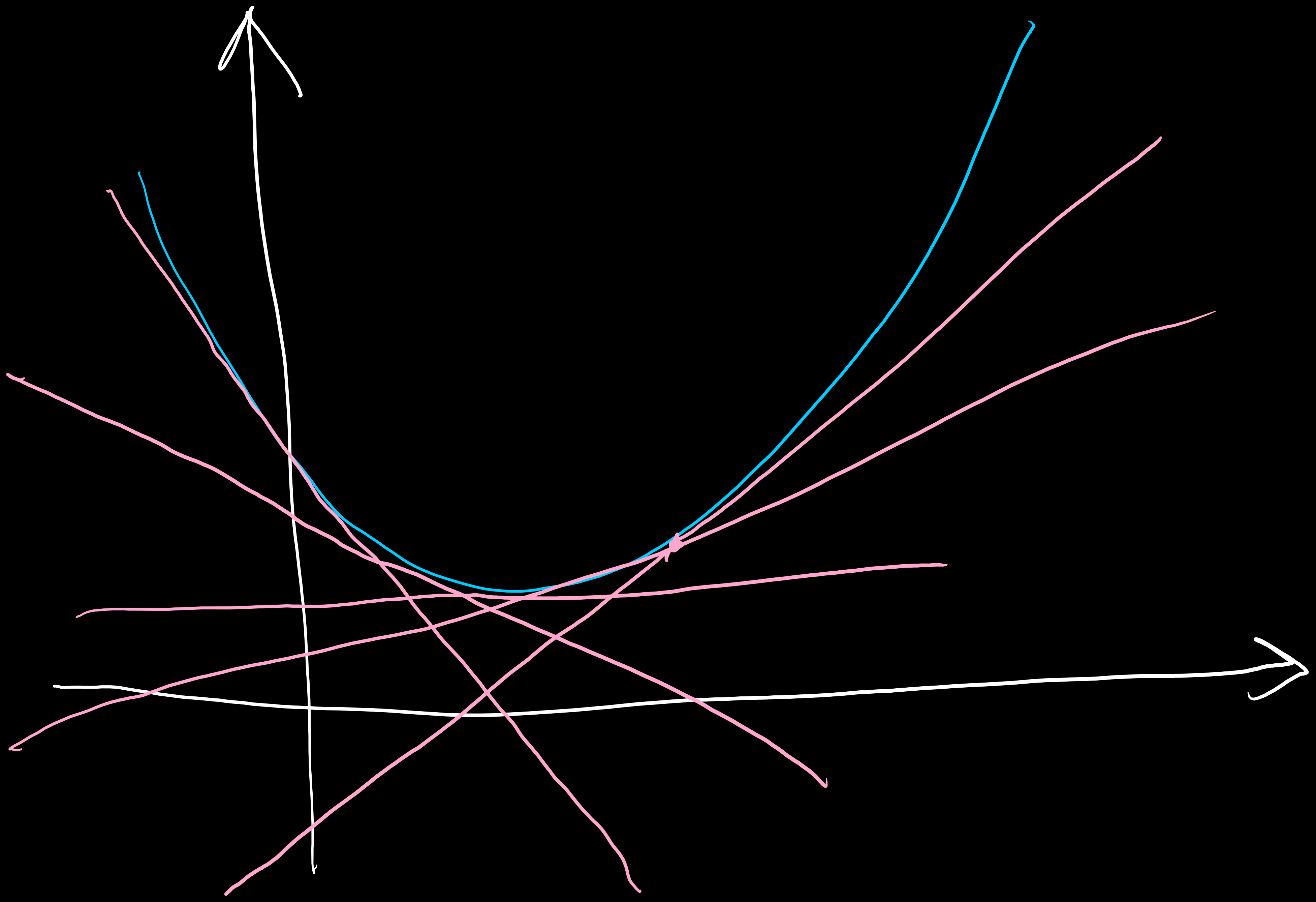
$$D_f(P \parallel Q) = \int_{\mathcal{X}} q(x)\, f\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x$$

f : generator function (convex & $f(1)=0$)

- Every convex function f has a *Fenchel conjugate* f^* so that

$$f(u) = \sup_{t \in \mathrm{dom}_{f^*}} \{tu - f^*(t)\}$$

"any convex f can be represented as point-wise max of *linear* functions"



Estimating f-divergences from samples (cont)

[Nguyen, Wainwright, Jordan, 2010]

$$\begin{aligned}
D_f(P \parallel Q) &= \int_{\mathcal{X}} q(x)\, f\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x \\
&= \int_{\mathcal{X}} q(x) \sup_{t \in \mathrm{dom}_{f^*}} \left\{ t \frac{p(x)}{q(x)} - f^*(t) \right\} \mathrm{d}x \\
&\geq \sup_{T \in \mathcal{T}} \left(\int_{\mathcal{X}} p(x)\, T(x)\, \mathrm{d}x - \int_{\mathcal{X}} q(x)\, f^*\big(T(x)\big)\, \mathrm{d}x \right) \\
&= \sup_{T \in \mathcal{T}} \big(\underbrace{\mathbb{E}_{x \sim P}[T(x)]} - \underbrace{\mathbb{E}_{x \sim Q}[f^*(T(x))]} \big)
\end{aligned}$$

Approximate using: samples from P samples from Q

f-GAN and GAN objectives

- GAN

$$\min_{\theta} \max_{\omega} \mathbb{E}_{x\sim P_\theta}[\log D_\omega(x)] + \mathbb{E}_{x\sim Q}[\log(1 - D_\omega(x))]$$

- f-GAN

$$\min_{\theta} \max_{\omega} \left(\mathbb{E}_{x\sim P_\theta}[T_\omega(x)] - \mathbb{E}_{x\sim Q}[f^*(T_\omega(x))]\right)$$

  - GAN discriminator-variational function correspondence: $\log D_\omega(x) = T_\omega(x)$
  - GAN minimizes the Jensen-Shannon divergence (which was also pointed out in Goodfellow et al., 2014)

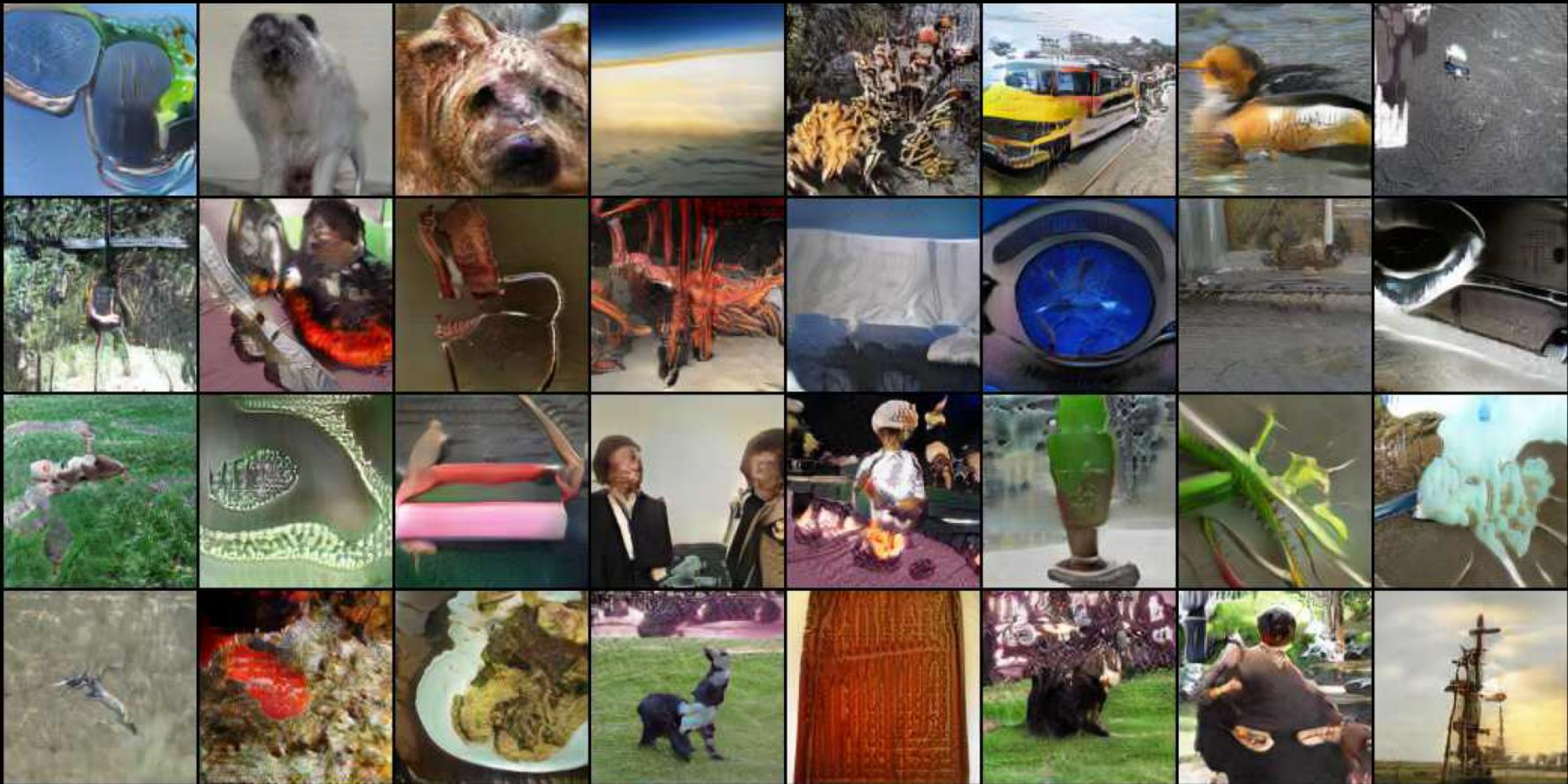
f-divergences

Name	$D_f(P\|Q)$	Generator $f(u)$
Total variation	$\frac{1}{2}\int \lvert p(x)-q(x)\rvert \,\mathrm{d}x$	$\frac{1}{2}\lvert u-1\rvert$
Kullback-Leibler	$\int p(x)\log\frac{p(x)}{q(x)}\,\mathrm{d}x$	$u\log u$
Reverse Kullback-Leibler	$\int q(x)\log\frac{q(x)}{p(x)}\,\mathrm{d}x$	$-\log u$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)}\,\mathrm{d}x$	$(u-1)^2$
Neyman χ^2	$\int \frac{(p(x)-q(x))^2}{q(x)}\,\mathrm{d}x$	$\frac{(1-u)^2}{u}$
Squared Hellinger	$\int \left(\sqrt{p(x)}-\sqrt{q(x)}\right)^2\,\mathrm{d}x$	$(\sqrt{u}-1)^2$
Jeffrey	$\int (p(x)-q(x))\log\left(\frac{p(x)}{q(x)}\right)\,\mathrm{d}x$	$(u-1)\log u$
Jensen-Shannon	$\frac{1}{2}\int p(x)\log\frac{2p(x)}{p(x)+q(x)}+q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x$	$-(u+1)\log\frac{1+u}{2}+u\log u$
Jensen-Shannon-weighted	$\int p(x)\pi\log\frac{p(x)}{\pi p(x)+(1-\pi)q(x)}+(1-\pi)q(x)\log\frac{q(x)}{\pi p(x)+(1-\pi)q(x)}\,\mathrm{d}x$	$\pi u\log u-(1-\pi+\pi u)\log(1-\pi+\pi u)$
GAN	$\int p(x)\log\frac{2p(x)}{p(x)+q(x)}+q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x-\log(4)$	$u\log u-(u+1)\log(u+1)$
α-divergence ($\alpha\notin\{0,1\}$)	$\frac{1}{\alpha(\alpha-1)}\int\left(p(x)\left[\left(\frac{q(x)}{p(x)}\right)^{\alpha}-1\right]-\alpha(q(x)-p(x))\right)\,\mathrm{d}x$	$\frac{1}{\alpha(\alpha-1)}\left(u^{\alpha}-1-\alpha(u-1)\right)$

Difficul

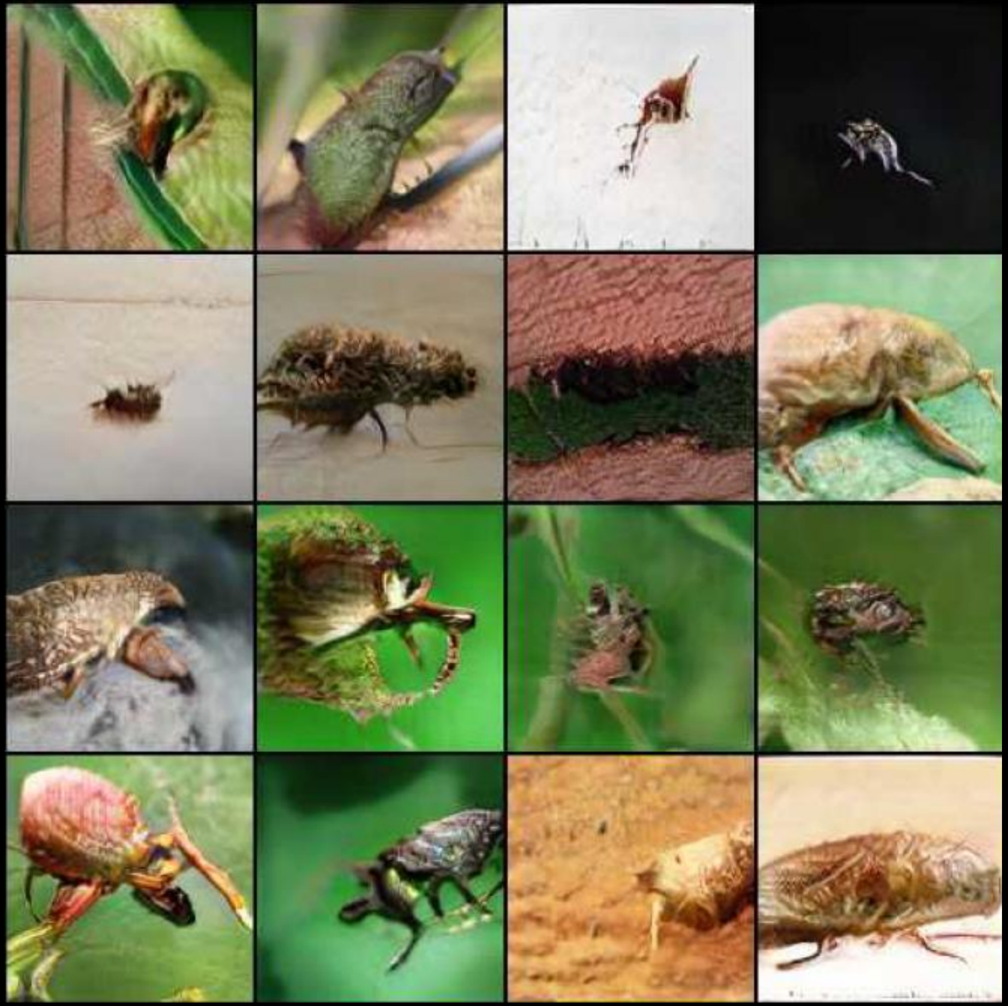
- GANs hav
  - "mode
- 2018: larg
  - Stabiliz
  - Conser

ImageNet 1000 classes, 128x128, unconditional generation, ResNet 55 [Mescheder et al., ICML 2018], arXiv:1801.04406



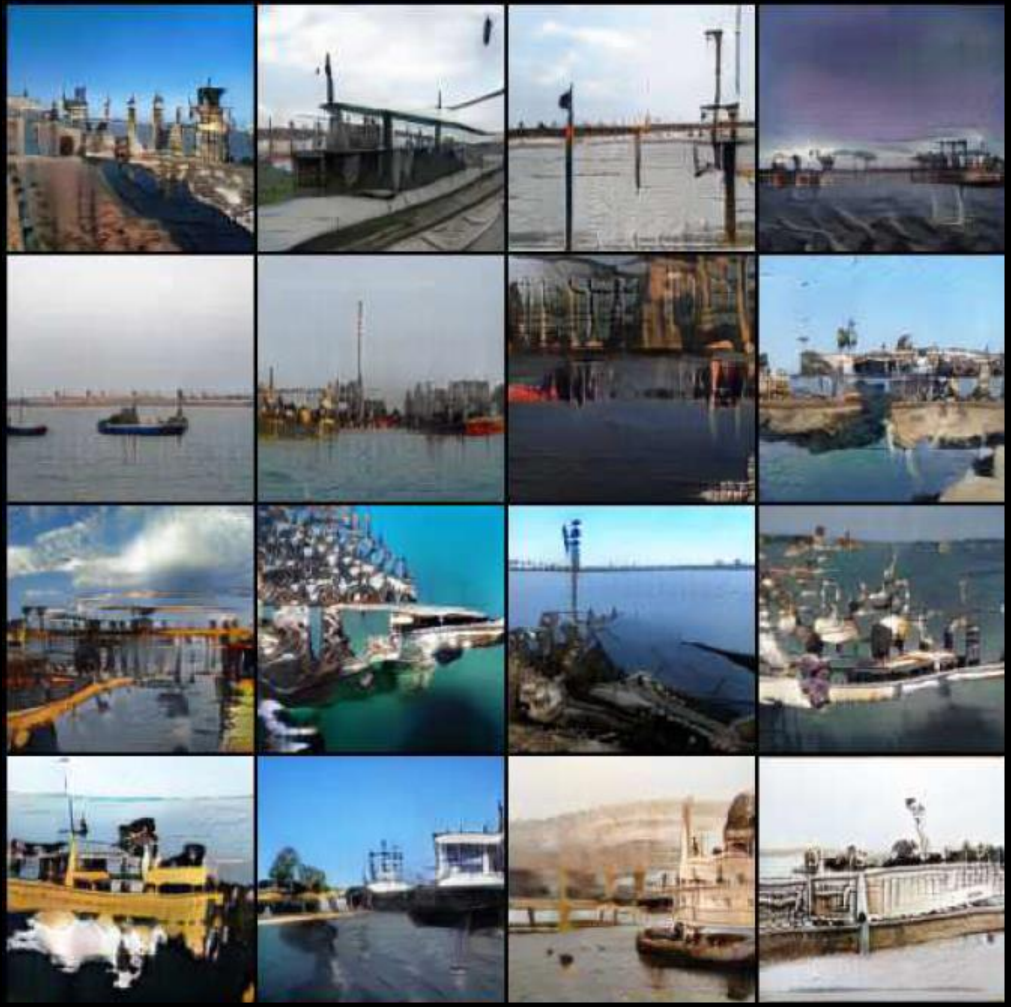
ImageNet conditional generation [Mescheder et al., ICML 2018], arXiv:1801.04406

papillon



weevil

ImageNet conditional generation [Mescheder et al., ICML 2018], arXiv:1801.04406



dock

pizza

Playing around with GANs

- https://github.com/rothk/Stabilizing_GANs
- Paper [Roth et al., NIPS 2017] https://arxiv.org/abs/1705.09367
- NIPS 2017 state of the art GAN models
  - Allows 150 layer ResNet GANs
  - TensorFlow implementation
- New state of the art: [Mescheder et al., ICML 2018], code https://github.com/LMescheder/GAN_stability

Thanks!

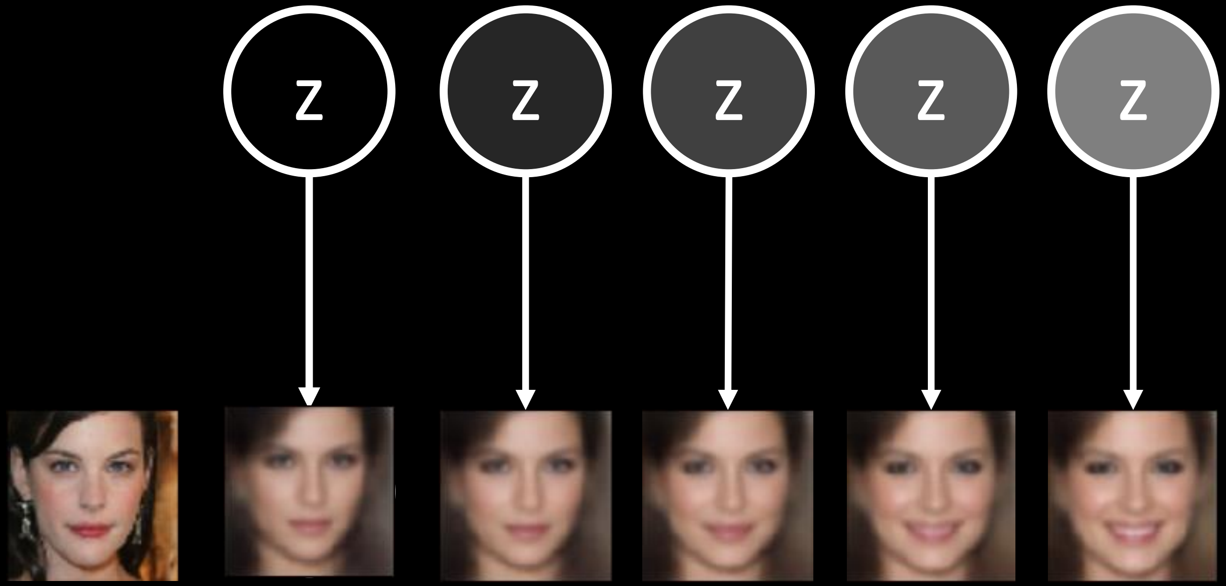
Sebastian.Nowozin@microsoft.com

Additional Material

Maximum Likelihood

- Fisher, 1928:
  Estimate model parameters by maximizing the likelihood function
  $$p(X;\theta) = \prod_i p(x_i;\theta)$$
- Equivalently, maximize the log-likelihood (more convenient)
  $$\log p(X;\theta) = \sum_i \log p(x_i;\theta)$$

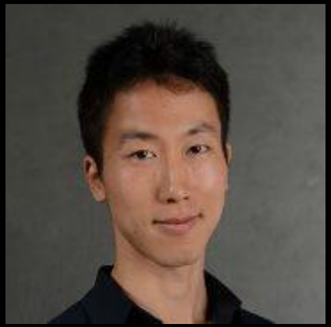
Representation Learning

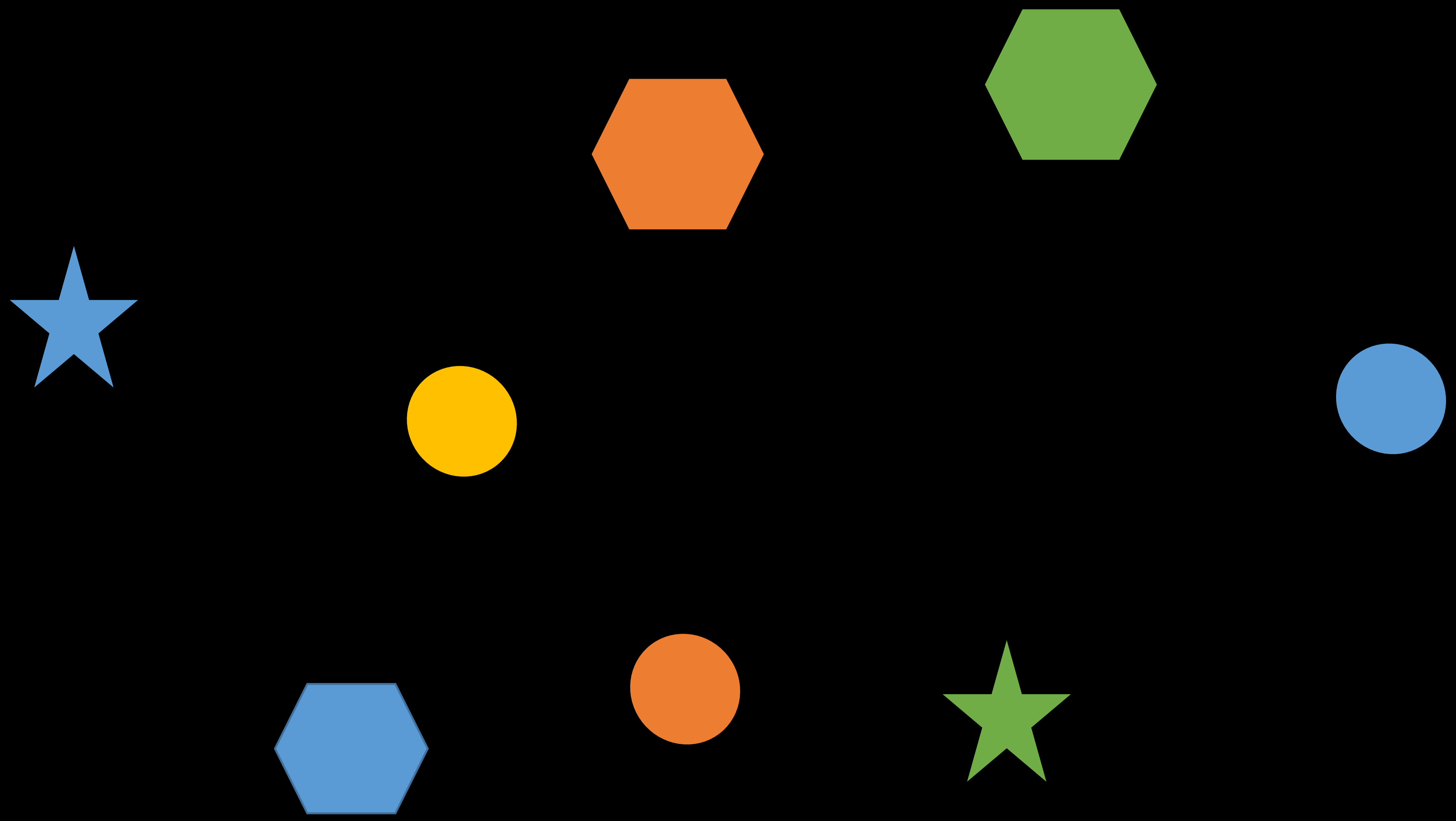


VAEs for Representation Learning

Diane Bouchacourt, Ryota Tomioka, Sebastian Nowozin

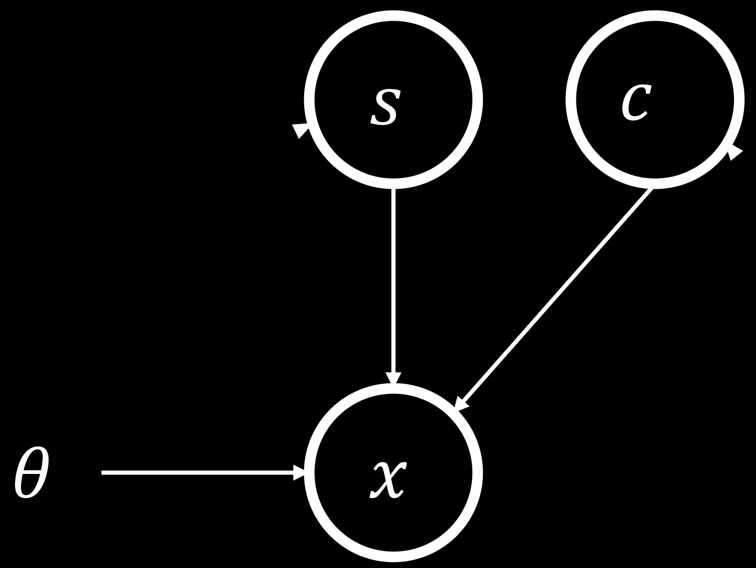
arXiv:1705.08841, NIPS 2017

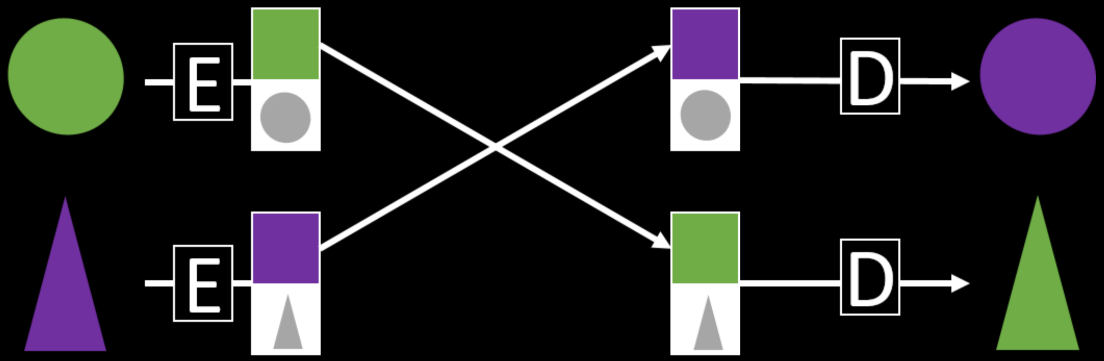




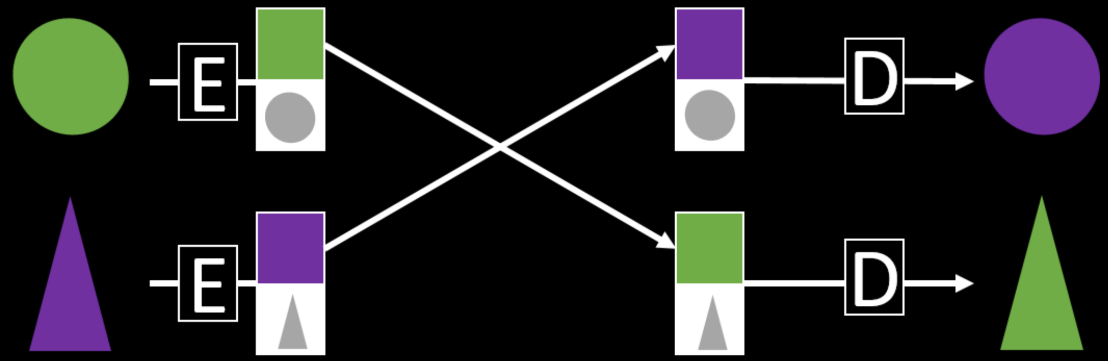
Two parts latent code

- Style s
- Content c




E
E
D
D

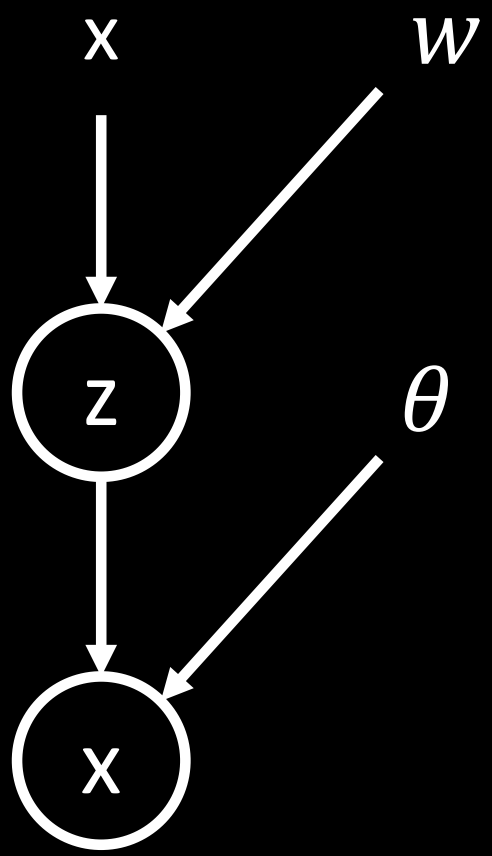

FIXED CONTENT
DATA SAMPLE
REC. SAMPLE
DATA SAMPLE
REC. SAMPLE
FIXED STYLE



Control over the latent space

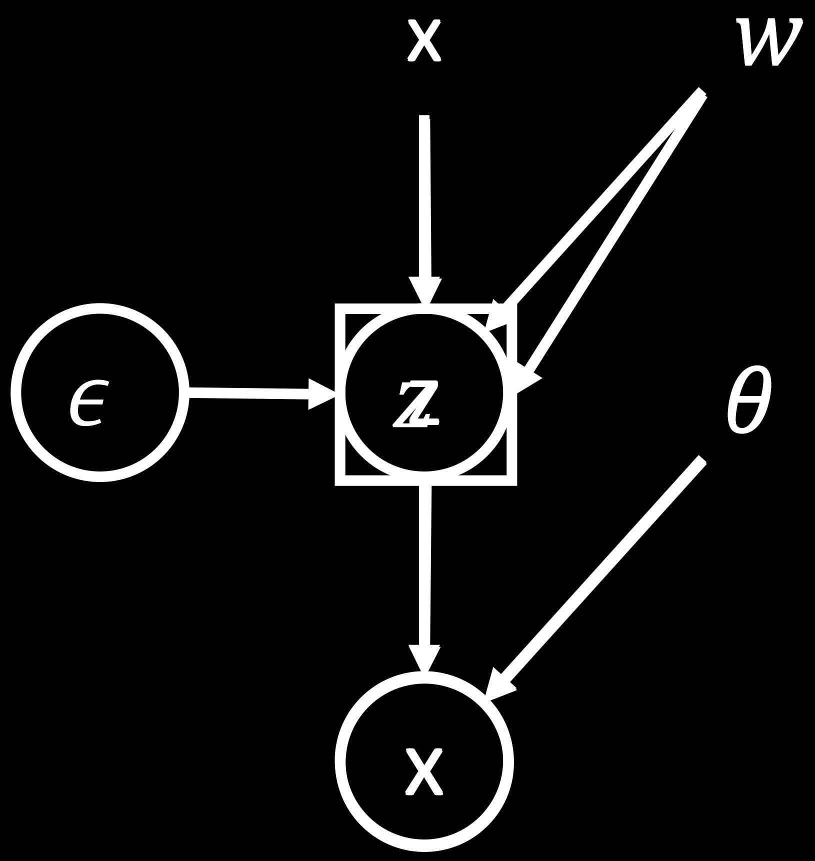
Reparametrization Trick

- [Rezende et al., 2014]
  [Kingma and Welling, 2014]



Reparametrization Trick

- [Rezende et al., 2014]
  [Kingma and Welling, 2014]
- Stochastic computation graphs
  [Schulman et al., 2015]



From highly-recommended tutorial: [Doersch, “Tutorial on Variational Autoencoders”, arXiv:1606.05908]