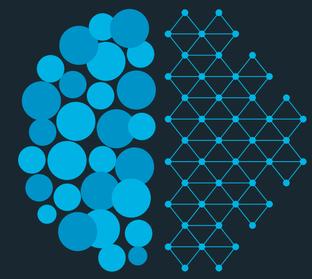
Institut
des algorithmes
d'apprentissage
de Montréal



MILA

Generative Models II

Aaron Courville
CIFAR Fellow, Université de Montréal

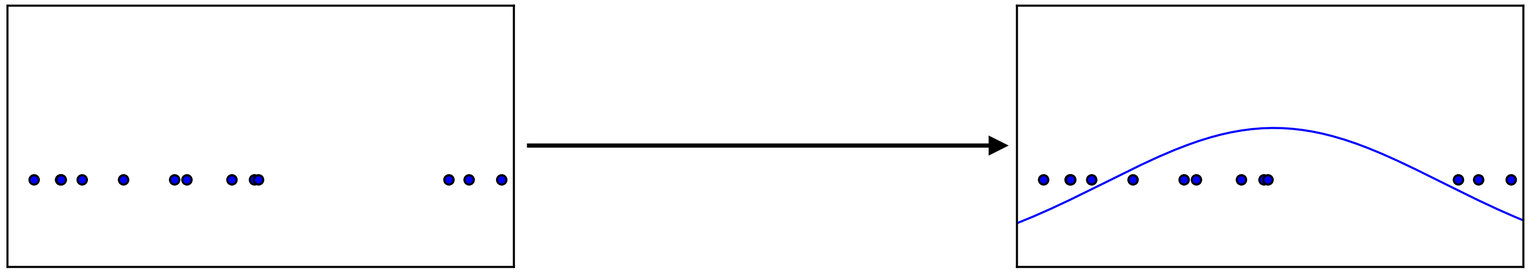
CIFAR-CRM Deep Learning Summer School

Université de Montréal, June 29th, 2017

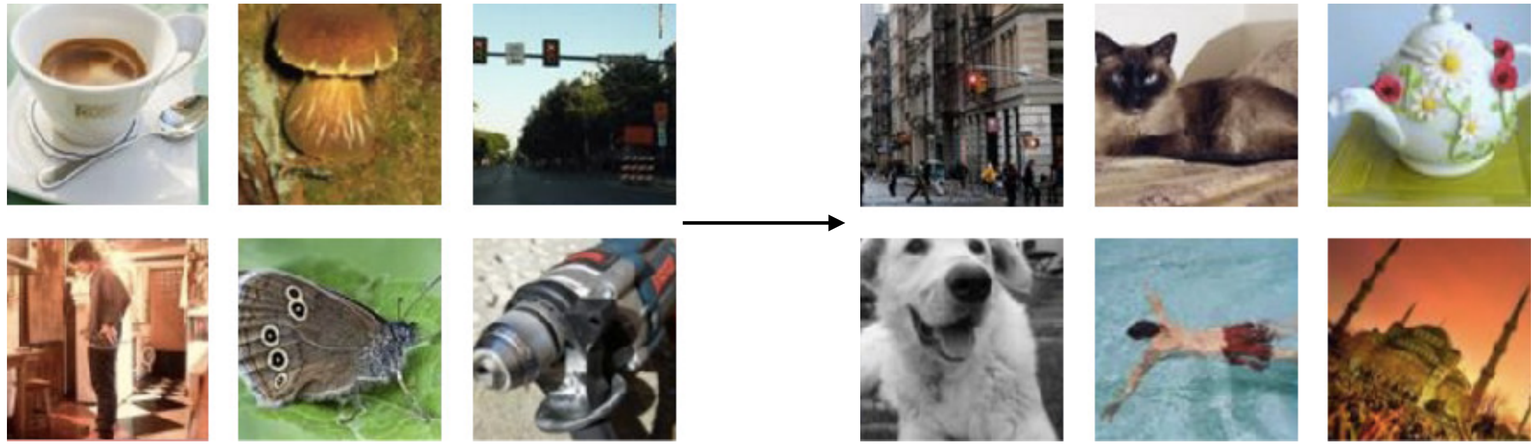
Generative modeling

- Generative models take training samples from some data distribution and learn a model that represents that distribution.

- Density estimation:



- Sample generation:



Training examples Model samples

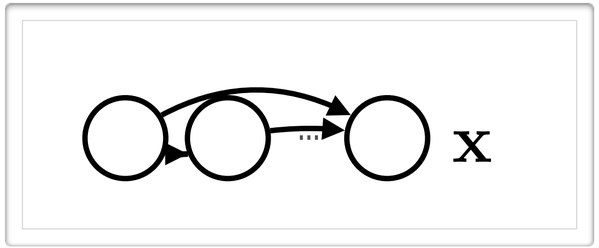
images taken from Goodfellow (2017)

Generative models II: Outline

- Autoregressive models
  - PixelCNN
- Latent variable models
  - Variational Autoencoders
    - VAE
    - Improved strategies for inference
  - Generative Adversarial Networks
    - GAN
    - Wasserstein GAN
    - ALI

Autoregressive generative models

- Autoregressive generative models are well known for sequence data (language modeling, time series, etc.)
- Less obviously applicable to arbitrary (non-sequential) observations
- Some history:



logistic regression for the conditionals (Frey et al., 1996)

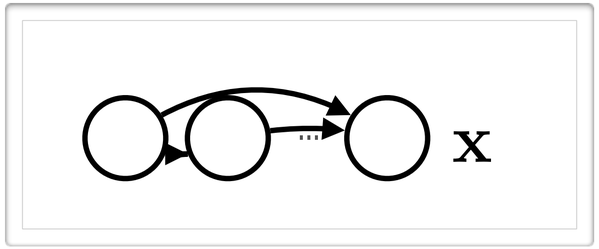
neural networks for the conditionals (Bengio and Bengio, 2000)

idem, with new weight sharing (NADE) (Larochelle and Murray, Gregor and Lecun, 2011)

Deep NADE, PixelRNN, PixelCNN, WaveNet, Video Pixel Network, etc.

Autoregressive generative models

- Choose an ordering of the dimensions in x.
- Define the conditionals in the product rule expression of p(x).

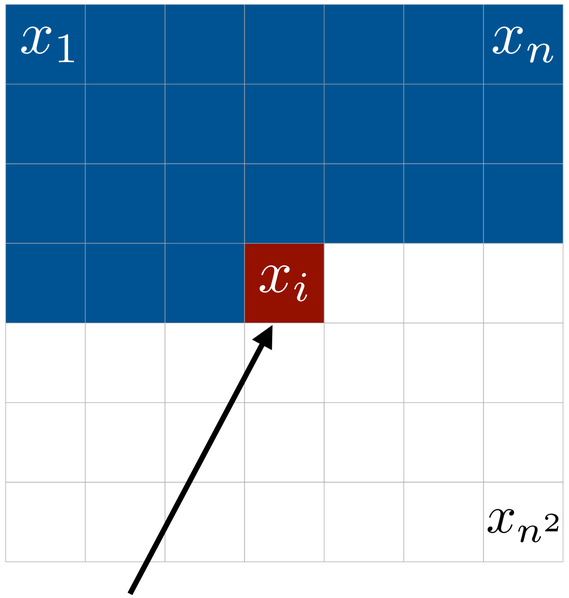


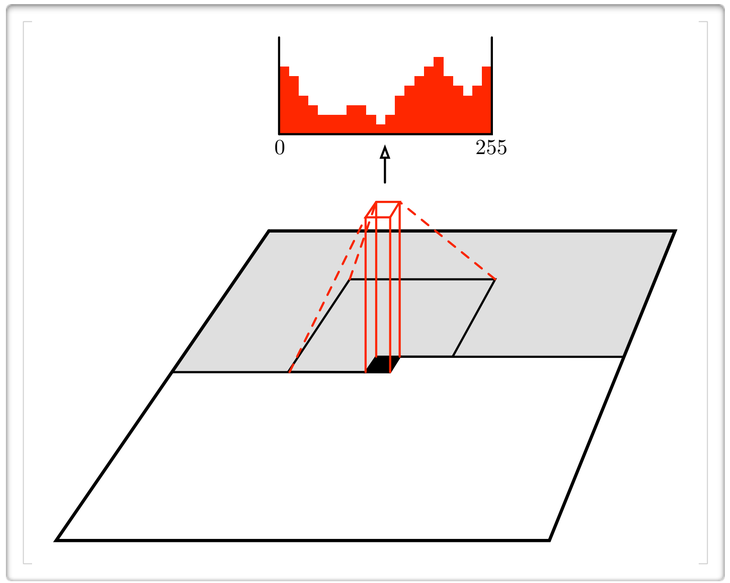
$$p(\mathbf{x}) = \prod_{k=1}^{D} p(x_k | \mathbf{x}_{<k})$$

- Properties
  - Pros: p(x) is tractable, so easy to train, easy to sample (though slower)
  - Cons: doesn't have a natural latent representation

PixelCNN

Idea: use masked convolutions to enforce the autoregressive relationship





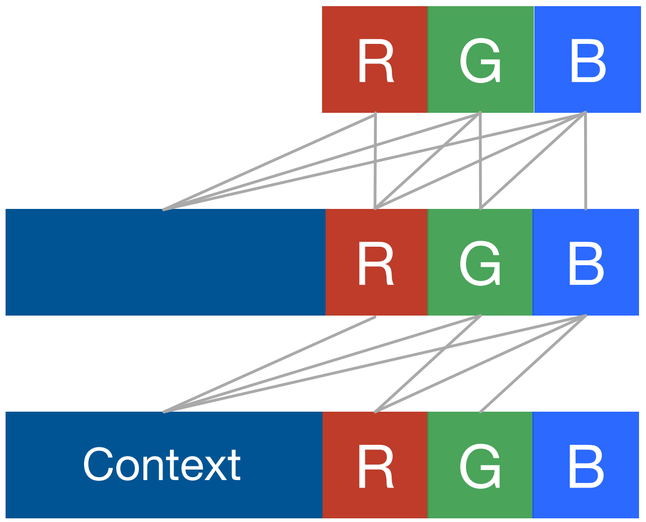
$$p(x_i \mid \mathbf{x}_{<i})$$

Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks." arXiv preprint arXiv:1601.06759 (2016).

PixelCNN

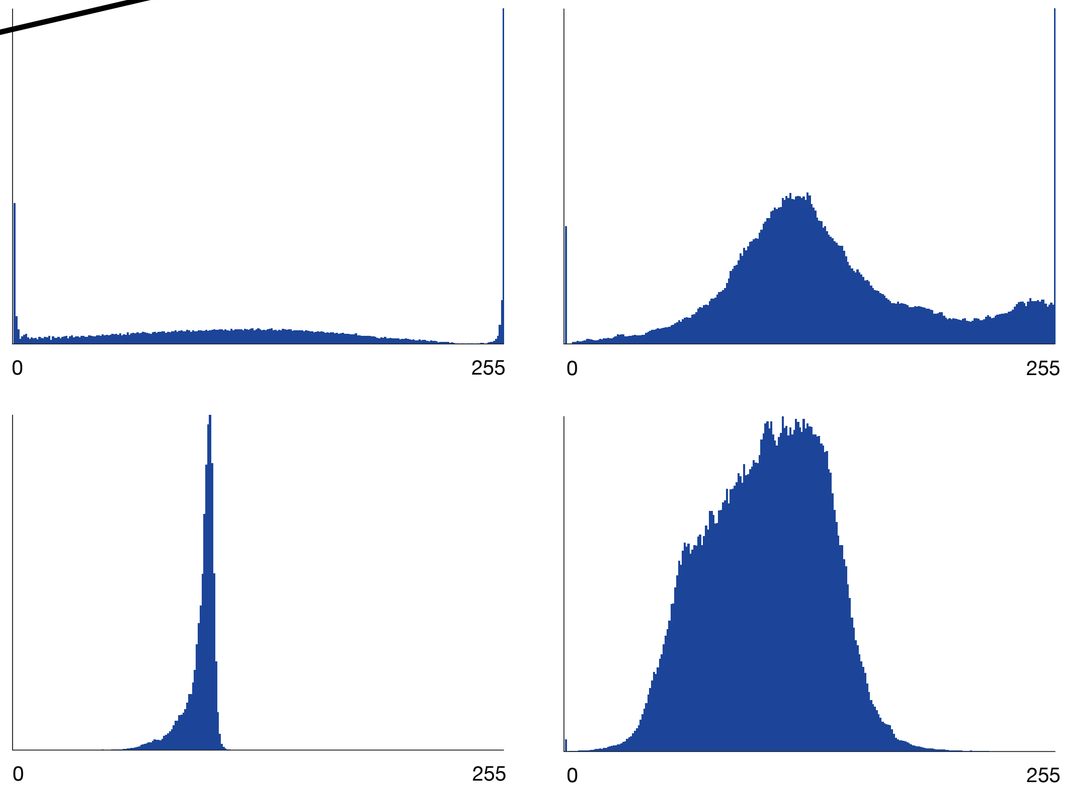
$$p(x_i \mid \mathbf{x}_{<i}) = p(x_{i,R} \mid \mathbf{x}_{<i})p(x_{i,G} \mid x_{i,R}, \mathbf{x}_{<i})p(x_{i,B} \mid x_{i,R}, x_{i,G}, \mathbf{x}_{<i})$$

autoregressive over color channels



Mask B

Mask A

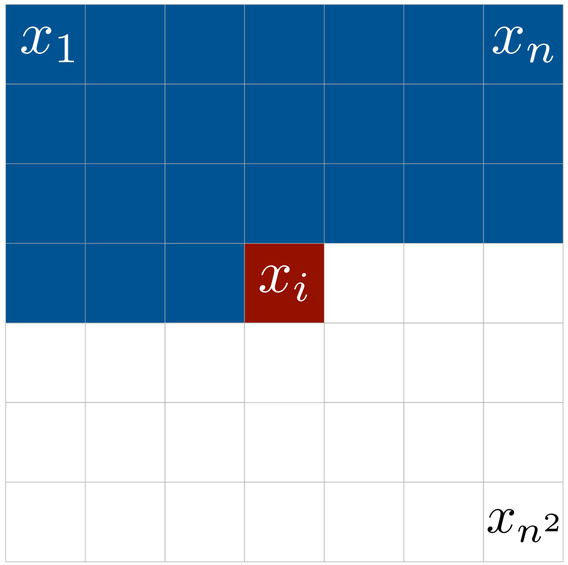


8-bits pixel values (multinoulli distribution)

Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks."arXiv preprint arXiv:1601.06759 (2016).

PixelCNN

How can convolutions make this raster scan faster?



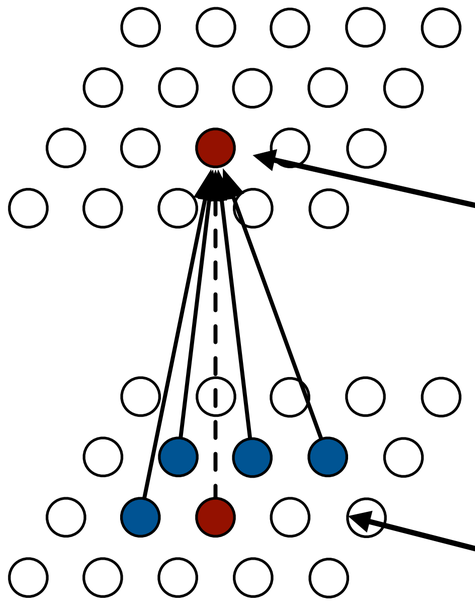
Use a stack of masked convolutions

1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

Training can be parallelized, though generation is still a sequential operation over pixels

van den Oord, Aaron, et al. "Conditional image generation with PixelCNN decoders." Advances in Neural Information Processing Systems. 2016.

PixelCNN



composing multiple layers increases the context size

only depends on pixel above and to the left

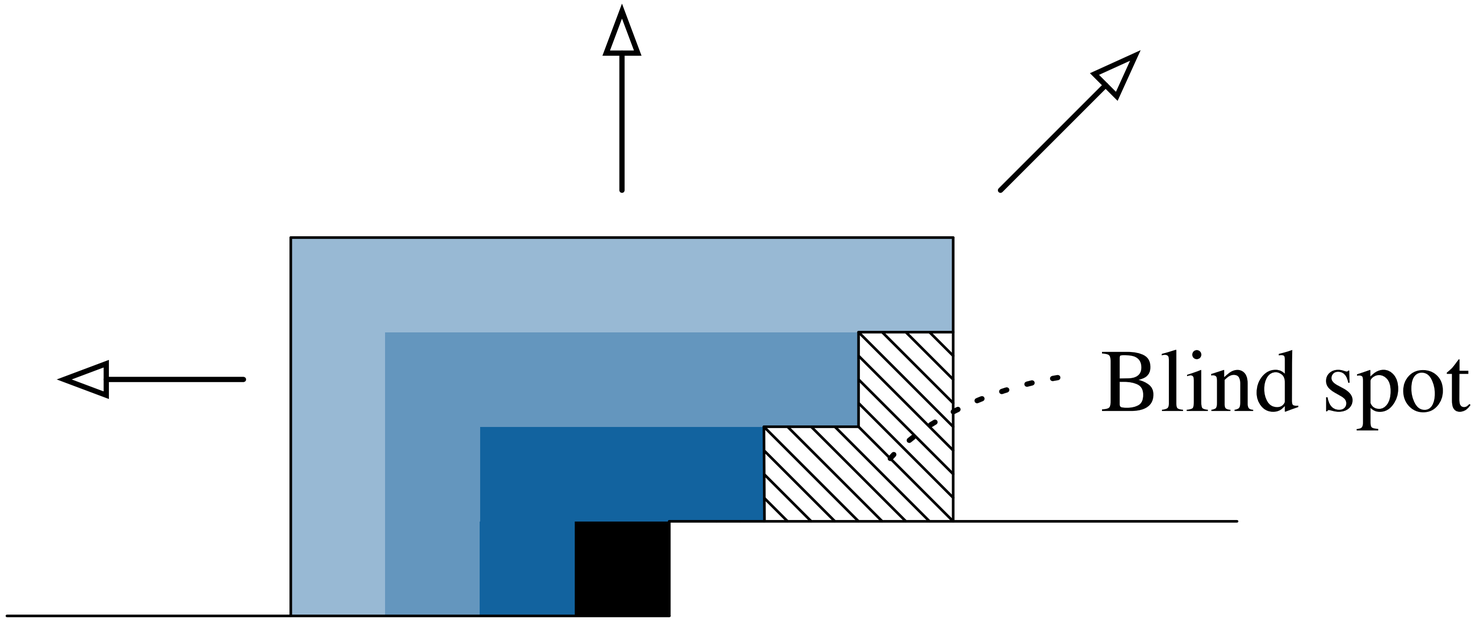
masked convolution

Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks." arXiv preprint arXiv:1601.06759 (2016).

Improving PixelCNN

There is a problem with this form of masked convolution.

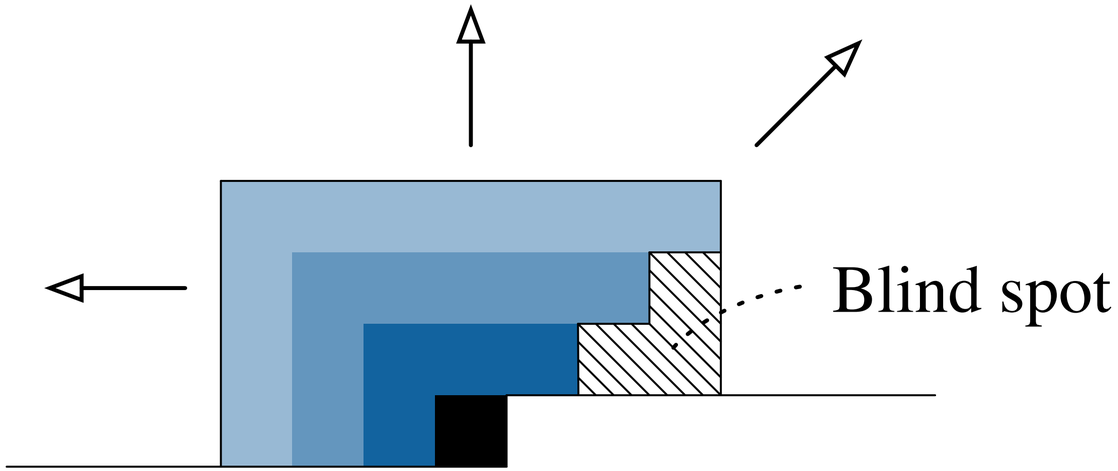
1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0



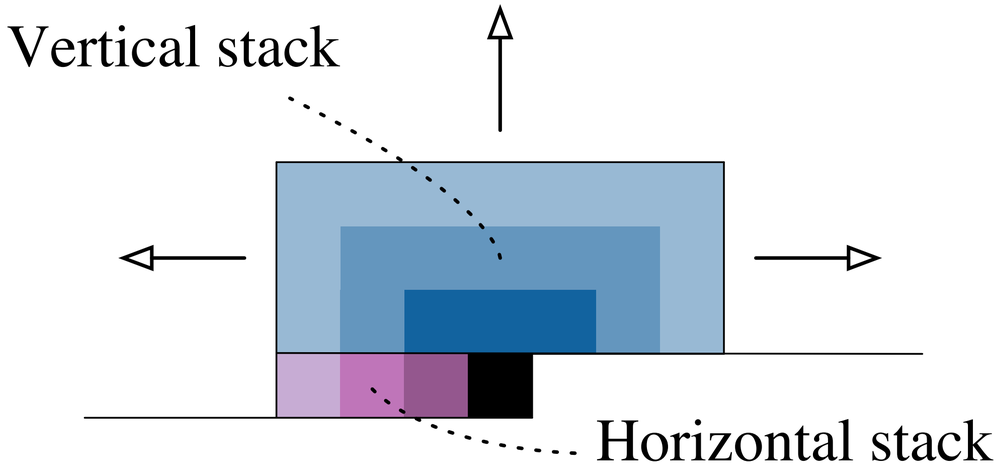
Stacking layers of masked convolution creates a blindspot

van den Oord, Aaron, et al. "Conditional image generation with PixelCNN decoders." Advances in Neural Information Processing Systems. 2016.

Improving PixelCNN I



Stacking layers of masked convolution creates a blindspot



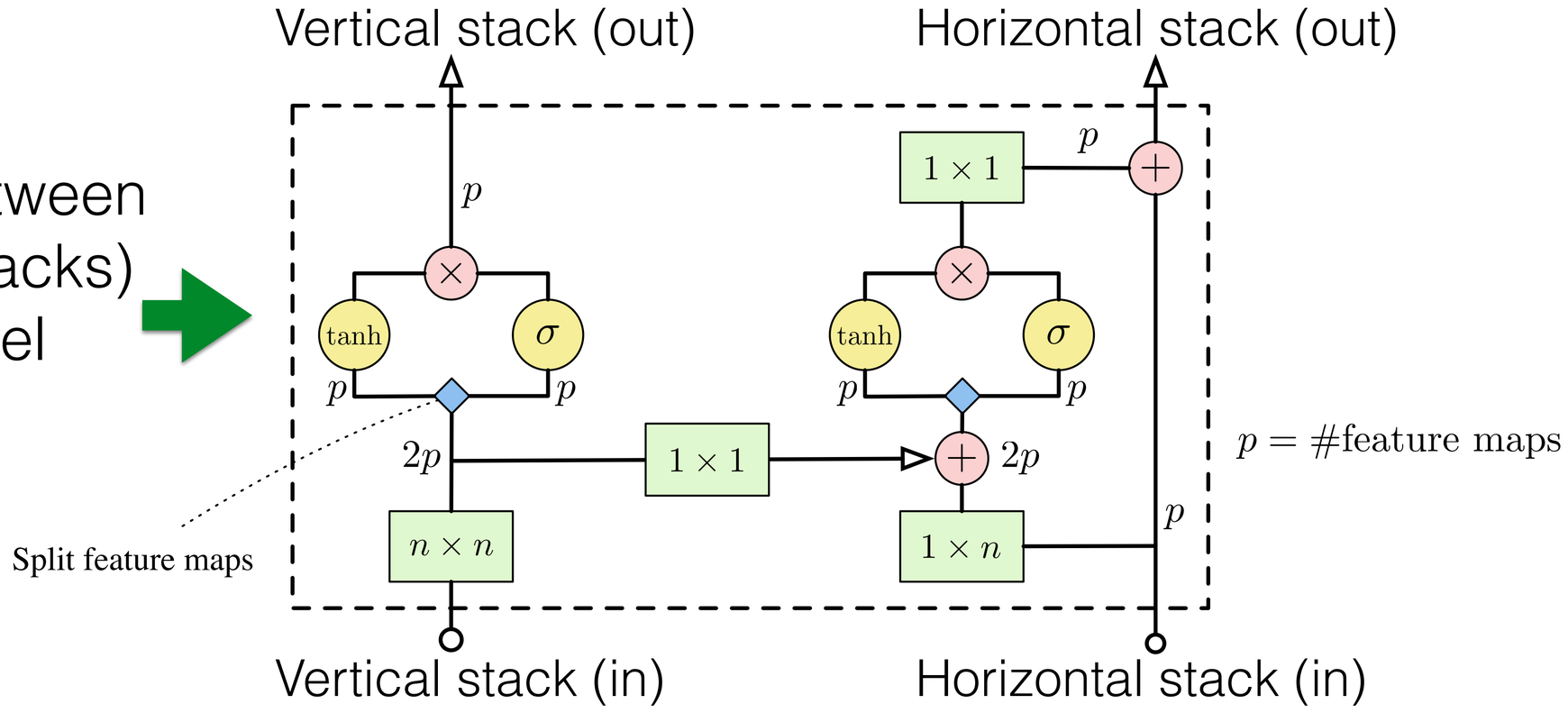
Solution: *use two stacks of convolution, a vertical stack and a horizontal stack*

van den Oord, Aaron, et al. "Conditional image generation with PixelCNN decoders." Advances in Neural Information Processing Systems. 2016.

Improving PixelCNN II

Use more expressive nonlinearity: $\mathbf{h}_{k+1} = \tanh(W_{k,f} * \mathbf{h}_k) \odot \sigma(W_{k,g} * \mathbf{h}_k)$

This information flow (between vertical and horizontal stacks) preserves the correct pixel dependencies



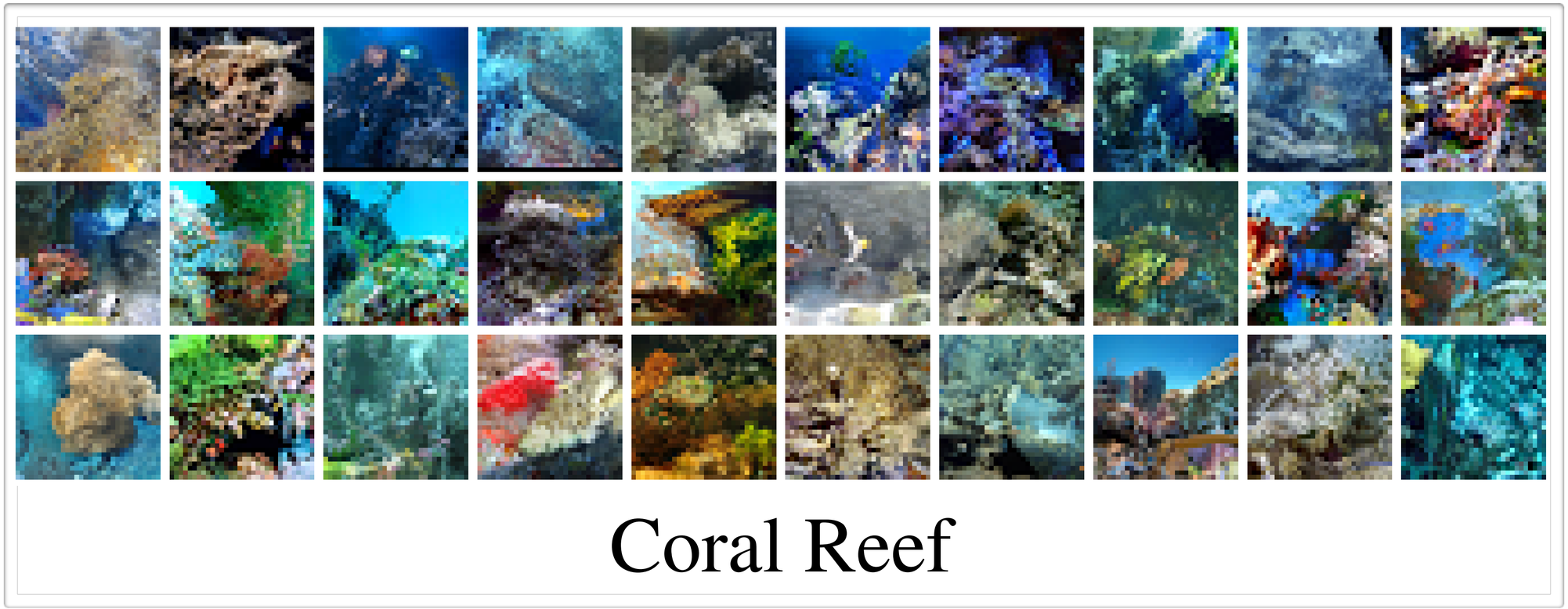
van den Oord, Aaron, et al. "Conditional image generation with PixelCNN decoders." NIPS 2016

PixelCNN: Experimental Results

Conditional Image Generation with PixelCNN Decoders
van den Oord, Kalchbrenner, Vinyals, Espeholt, Graves, Kavukcuoglu, NIPS 2016

Topics: CIFAR-10

- Samples from a class-conditional PixelCNN



PixelCNN: Experimental Results

Conditional Image Generation with PixelCNN Decoders
van den Oord, Kalchbrenner, Vinyals, Espeholt, Graves, Kavukcuoglu, NIPS 2016

Topics: CIFAR-10

- Samples from a class-conditional PixelCNN

PixelCNN: Experimental Results

Conditional Image Generation with PixelCNN Decoders
van den Oord, Kalchbrenner, Vinyals, Espeholt, Graves, Kavukcuoglu, NIPS 2016

Topics: CIFAR-10

- Samples from a class-conditional PixelCNN

PixelCNN: Experimental Results

Topics: CIFAR-10

Conditional Image Generation with PixelCNN Decoders
van den Oord, Kalchbrenner, Vinyals, Espeholt, Graves, Kavukcuoglu, NIPS 2016

- Performance measured in bits/dim

Model	NLL Test (Train)
Uniform Distribution: [30]	8.00
Multivariate Gaussian: [30]	4.70
NICE: [4]	4.48
Deep Diffusion: [24]	4.20
DRAW: [9]	4.13
Deep GMMs: [31, 29]	4.00
Conv DRAW: [8]	3.58 (3.57)
RIDE: [26, 30]	3.47
PixelCNN: [30]	3.14 (3.08)
PixelRNN: [30]	3.00 (2.93)
Gated PixelCNN:	3.03 (2.90)

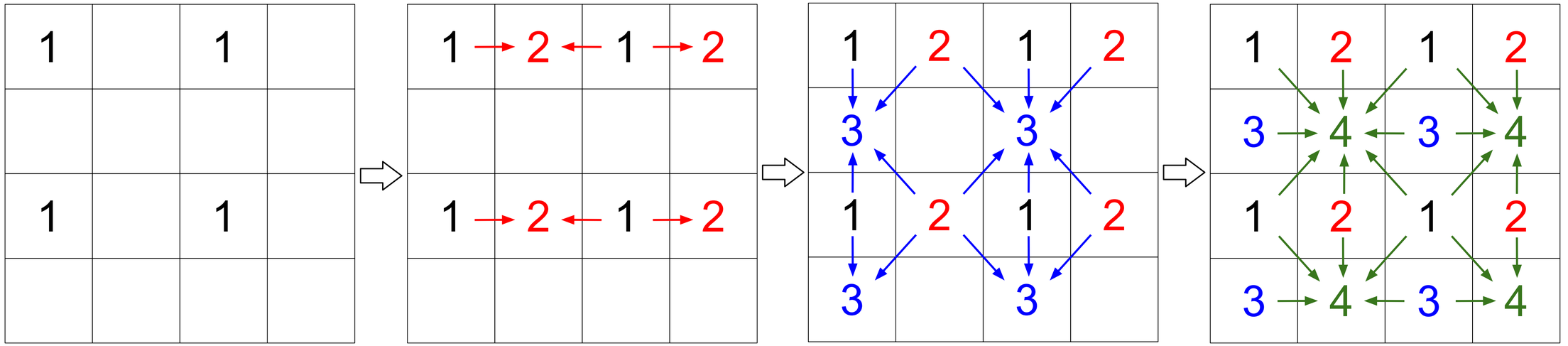
Parallel Multiscale Autoregressive Density Estimation

Scott Reed, Aaron vanden Oord, Nal Kalchbrenner, Sergio Go´mez Colmenarejo, Ziyu Wang, Dan Belov, Nando de Freitas (2017)

MILA

Can we speed up the generation time of PixelCNN?

- Yes, via multiscale generation:



Parallel Multiscale Autoregressive Density Estimation

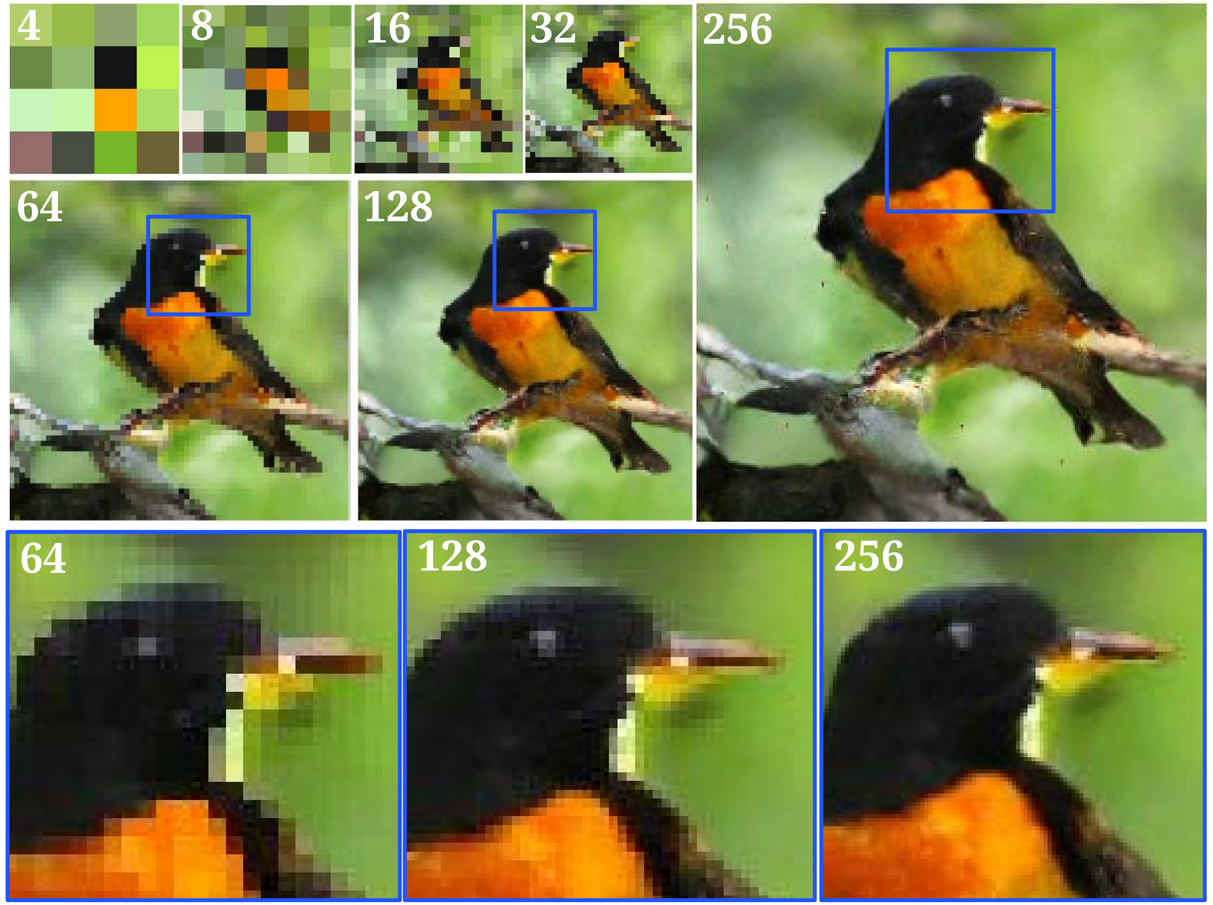
Scott Reed, Aaron vanden Oord, Nal Kalchbrenner, Sergio Go´mez Colmenarejo, Ziyu Wang, Dan Belov, Nando de Freitas (2017)

MILA

Can we speed up the generation time of PixelCNN?

- Yes, via multiscale generation.
- Also seems to help to provide better global structure

"A yellow bird with a black head, orange eyes and an orange bill."



Latent Variable Models

- The Variational Autoencoder model:
  - Kingma and Welling, *Auto-Encoding Variational Bayes*, *International Conference on Learning Representations (ICLR)* 2014.
  - Rezende, Mohamed and Wierstra, *Stochastic back-propagation and variational inference in deep latent Gaussian models*. ICML 2014.

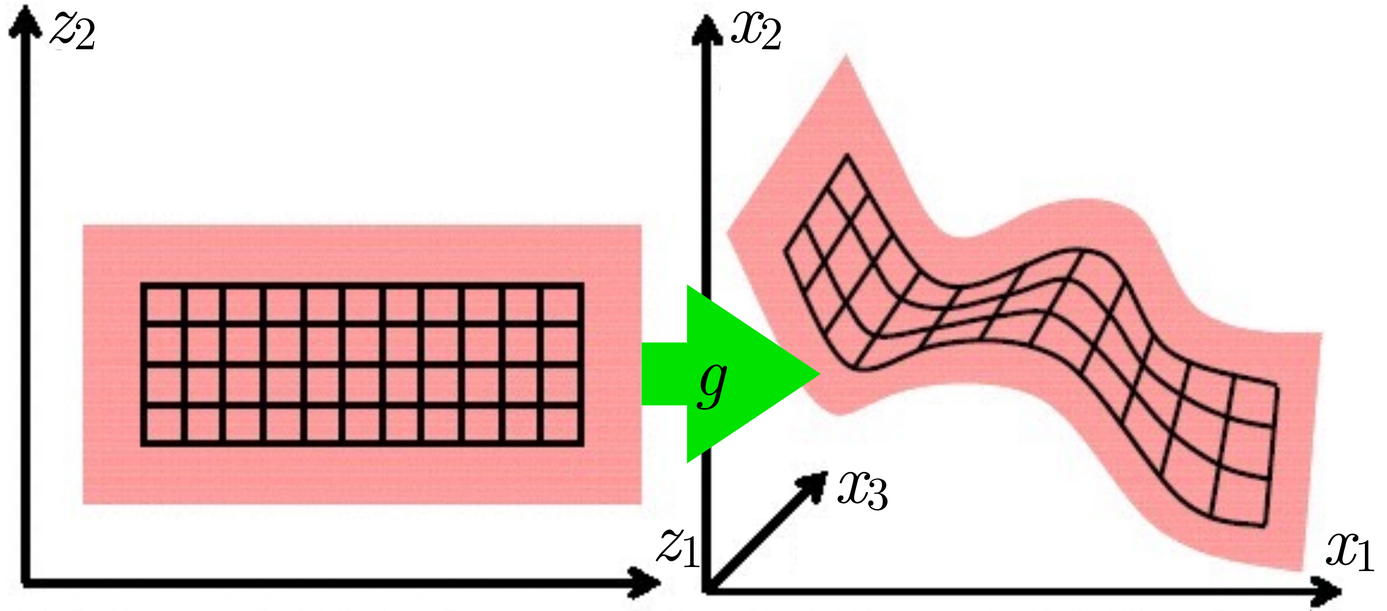
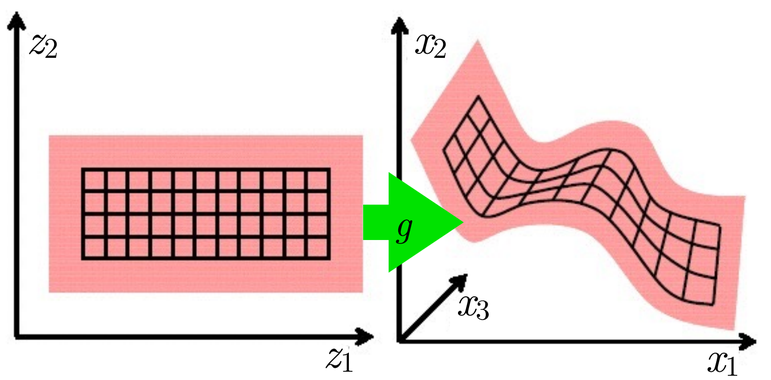
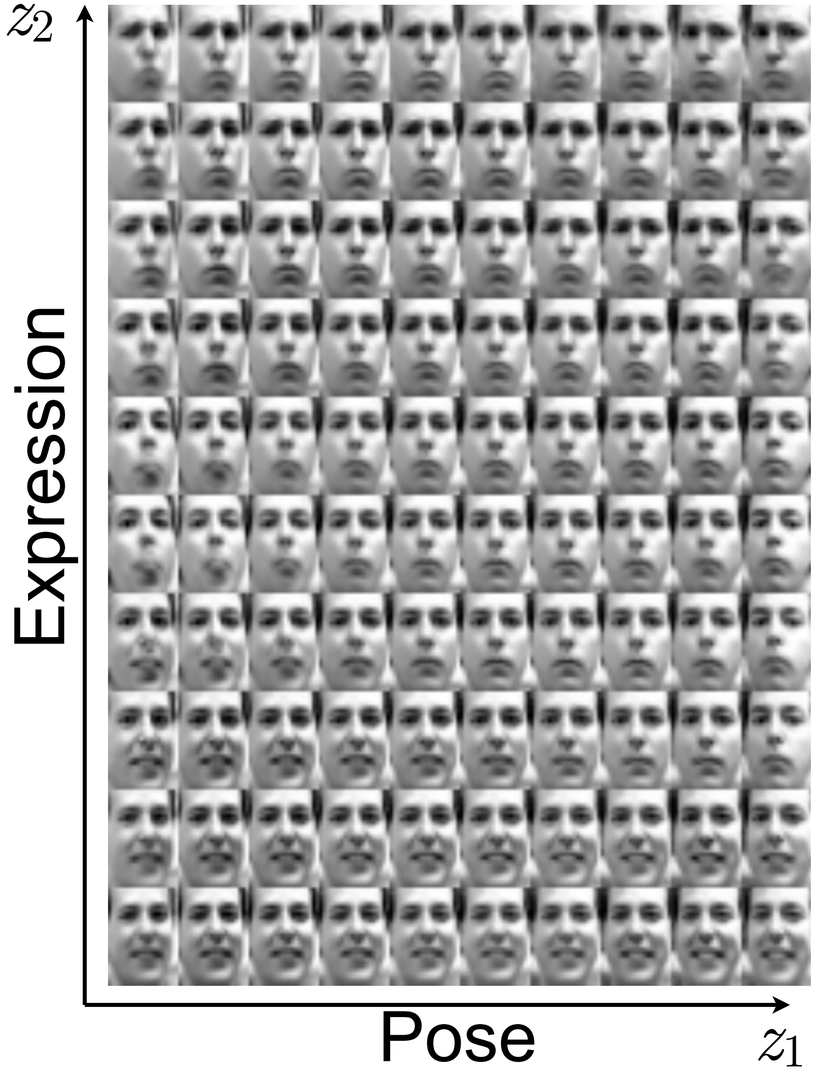


Image from: Ward, A. D., Hamarneh, G.: **3D Surface Parameterization Using Manifold Learning for Medial Shape Representation**, *Conference on Image Processing, Proc. of SPIE Medical Imaging*, 2007

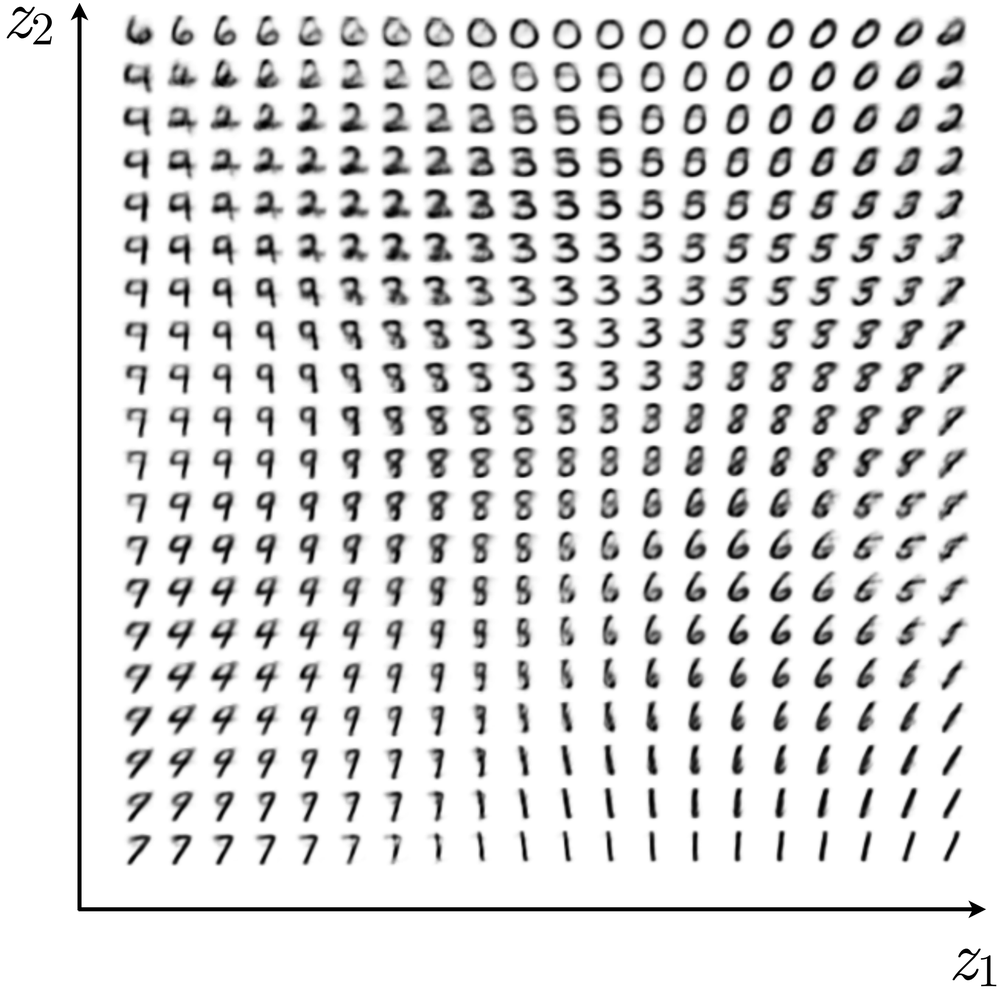
Latent Variable Models



Frey Face dataset:



MNIST:



Latent Variable Models

- latent variable model: learn a mapping from some latent variable z to a complicated distribution on x.

$$p(x) = \int p(x, z)\ dz \qquad \text{where} \quad p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{z})p(\boldsymbol{z})$$

$$p(\boldsymbol{z}) = \text{something simple} \qquad p(\boldsymbol{x} \mid \boldsymbol{z}) = g(\boldsymbol{z})$$

- Can we learn to decouple the true explanatory factors underlying the data distribution? E.g. separate identity and expression in face images

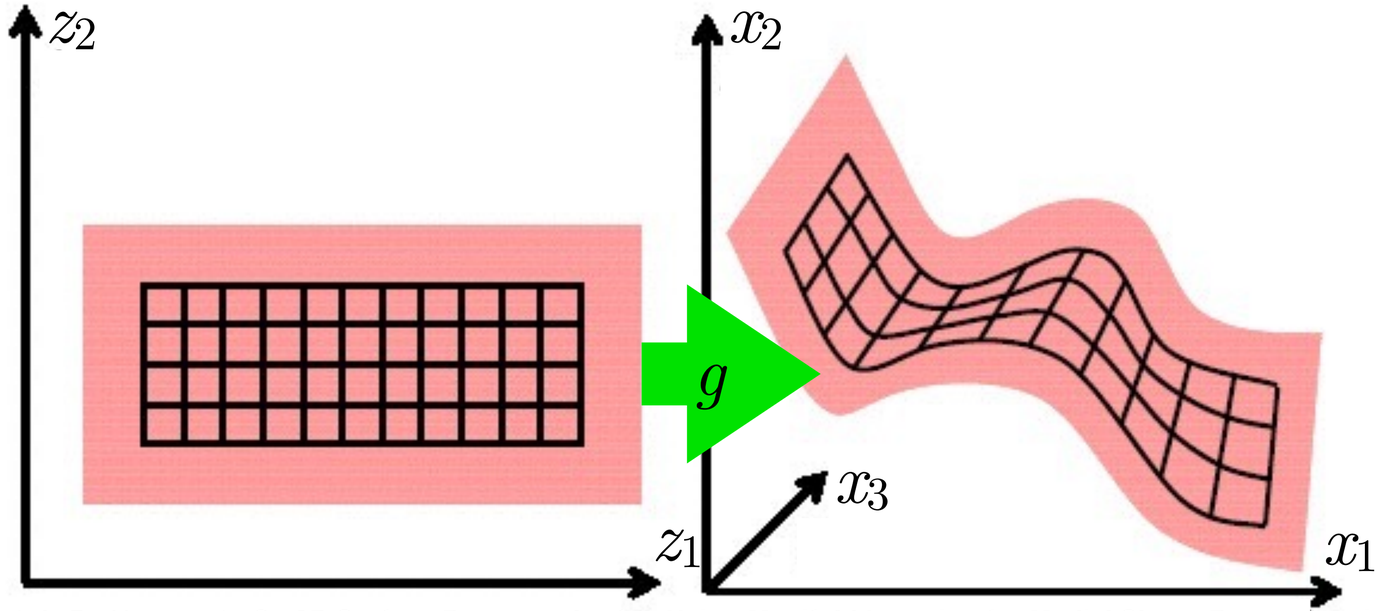


Image from: Ward, A. D., Hamarneh, G.: **3D Surface Parameterization Using Manifold Learning for Medial Shape Representation**, *Conference on Image Processing, Proc. of SPIE Medical Imaging*, 2007

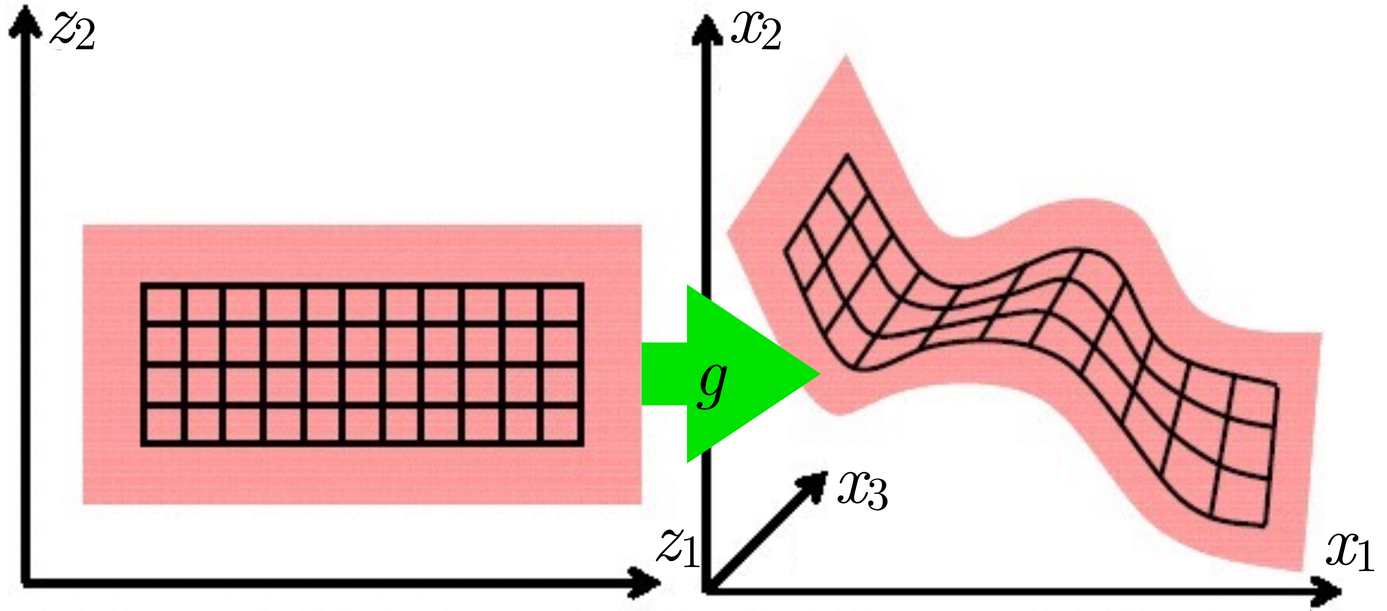
Latent Variable Models

- latent variable model: learn a mapping from some latent variable z to a complicated distribution on x.

$$p(x) = \int p(x, z)\ dz \qquad \text{where} \quad p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{z})p(\boldsymbol{z})$$

$$p(\boldsymbol{z}) = \text{something simple} \qquad p(\boldsymbol{x} \mid \boldsymbol{z}) = g(\boldsymbol{z})$$

- Can we learn to decouple the true explanatory factors underlying the data distribution? E.g. separate identity and expression in face images



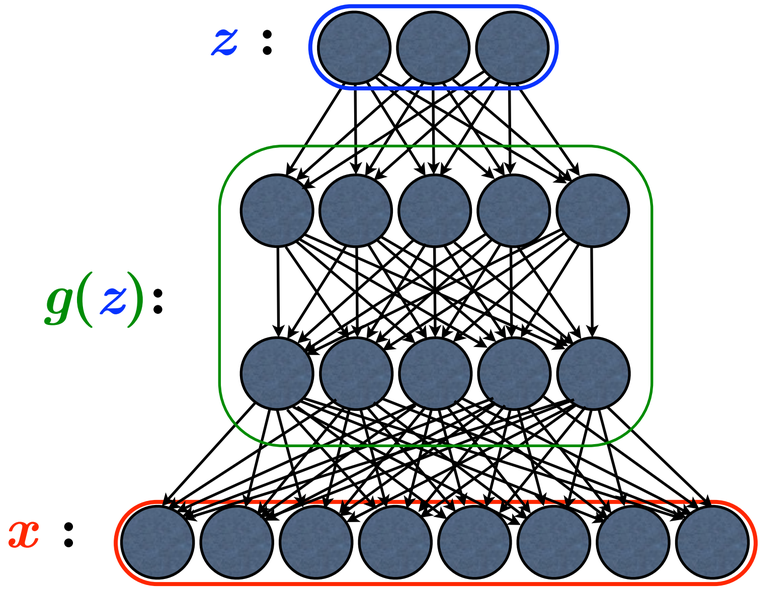


Image from: Ward, A. D., Hamarneh, G.: **3D Surface Parameterization Using Manifold Learning for Medial Shape Representation**, *Conference on Image Processing, Proc. of SPIE Medical Imaging*, 2007

Variational Auto-Encoder (VAE)

- Where does z come from? — The classic DAG problem.
- The VAE approach: introduce an inference machine $q_\phi(z \mid x)$ that learns to approximate the posterior $p_\theta(z \mid x)$.
- Define a variational lower bound on the data likelihood: $p_\theta(x) \geq \mathcal{L}(\theta, \phi, x)$

$$
\begin{aligned}
\mathcal{L}(\theta, \phi, x) &= \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x, z) - \log q_\phi(z \mid x) \right] \\
&= \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z) + \log p_\theta(z) - \log q_\phi(z \mid x) \right] \\
&= -D_{\mathrm{KL}} \left(q_\phi(z \mid x) \| p_\theta(z) \right) + \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z) \right]
\end{aligned}
$$

***regularization* term** ***reconstruction* term**

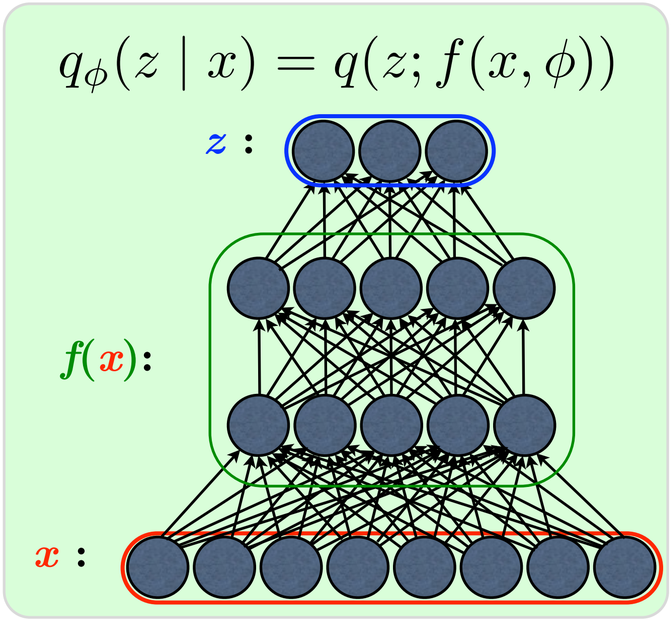
- What is $q_\phi(z \mid x)$?

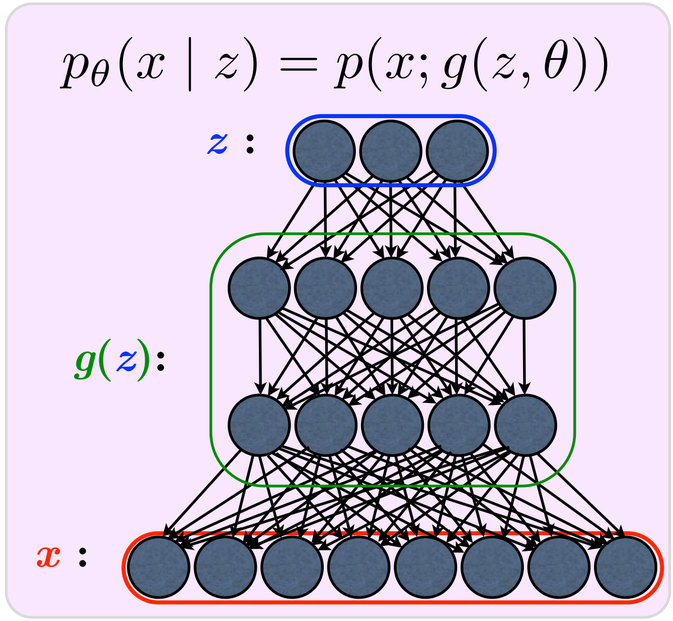
VAE Inference model

- The VAE approach: introduce an inference model $q_\phi(z \mid x)$ that learns to approximates the intractable posterior $p_\theta(z \mid x)$ by optimizing the variational lower bound:

$$\mathcal{L}(\theta, \phi, x) = -D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| p_\theta(z)\right) + \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right]$$

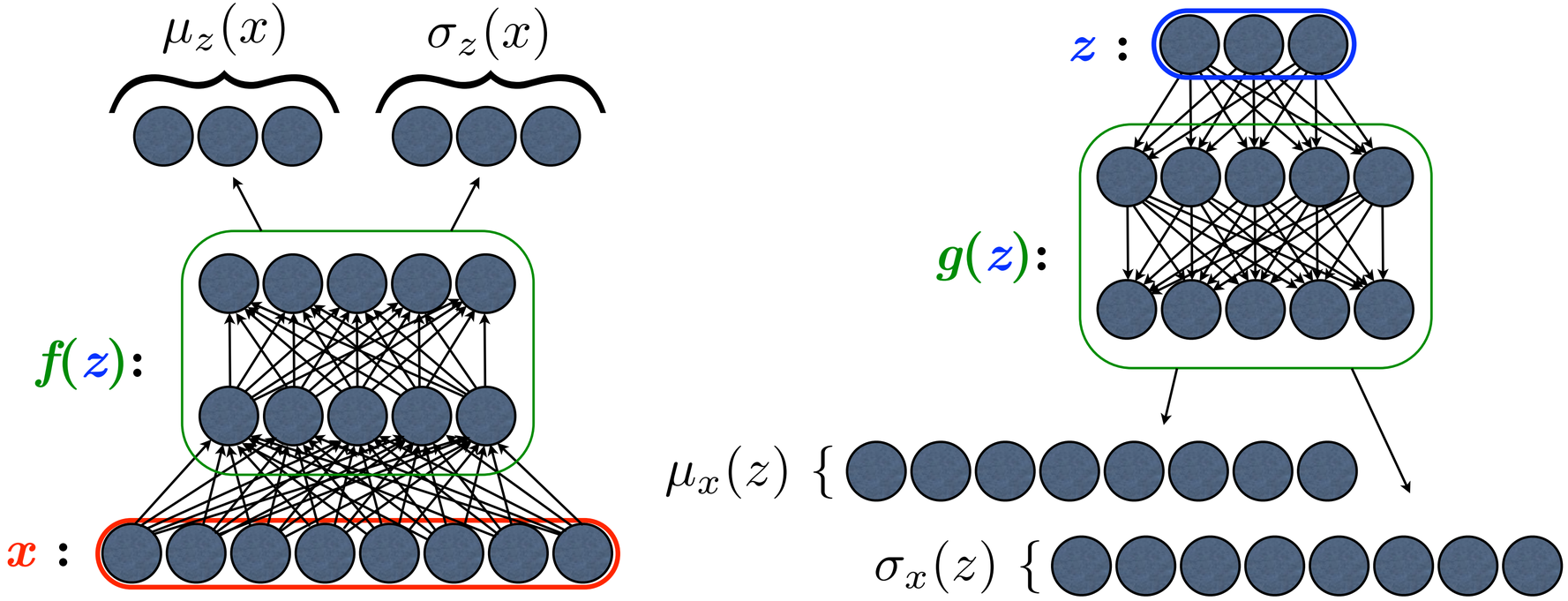
- We parameterize $q_\phi(z \mid x)$ with another neural network:





Reparametrization trick

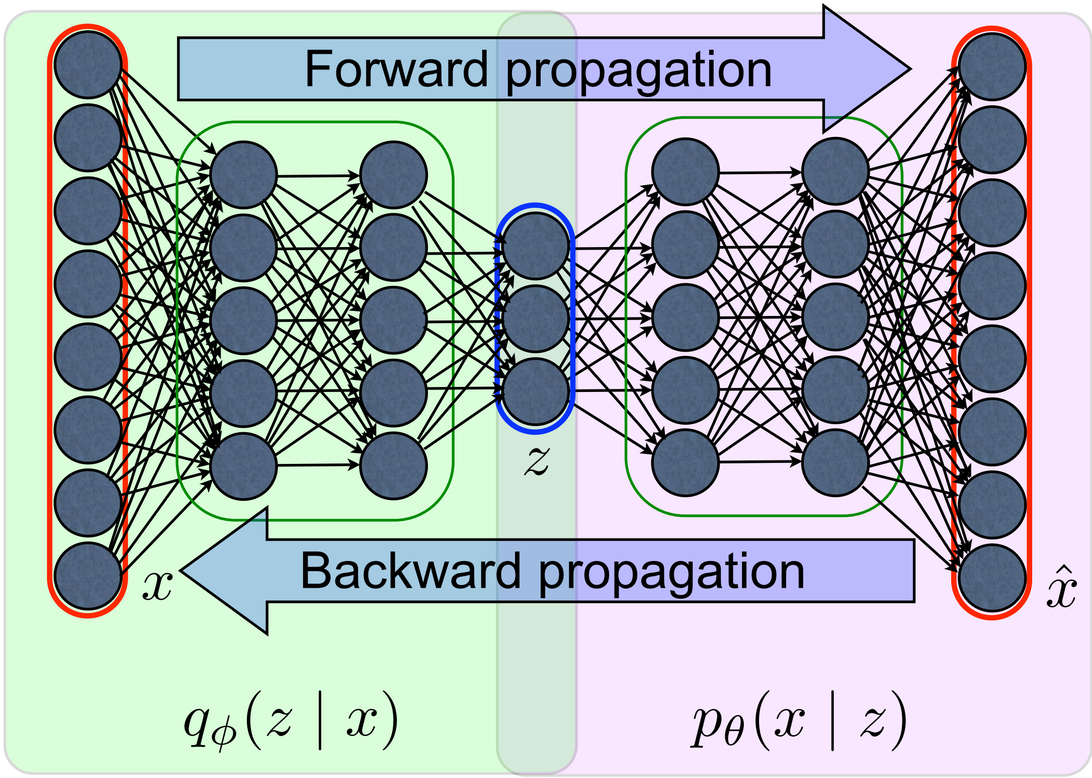
- Adding a few details + one really important trick
- Let's consider z to be real and $q_\phi(z \mid x) = \mathcal{N}(z; \mu_z(x), \sigma_z(x))$
- Parametrize z as $z = \mu_z(x) + \sigma_z(x)\epsilon_z$ where $\epsilon_z = \mathcal{N}(0, 1)$
- (optional) Parametrize x as $x = \mu_x(z) + \sigma_x(z)\epsilon_x$ where $\epsilon_x = \mathcal{N}(0, 1)$



Training with backpropagation!

- Due to a reparametrization trick, we can simultaneously train both the generative model $p_\theta(x \mid z)$ and the inference model $q_\phi(z \mid x)$ by optimizing the variational bound using gradient backpropagation.

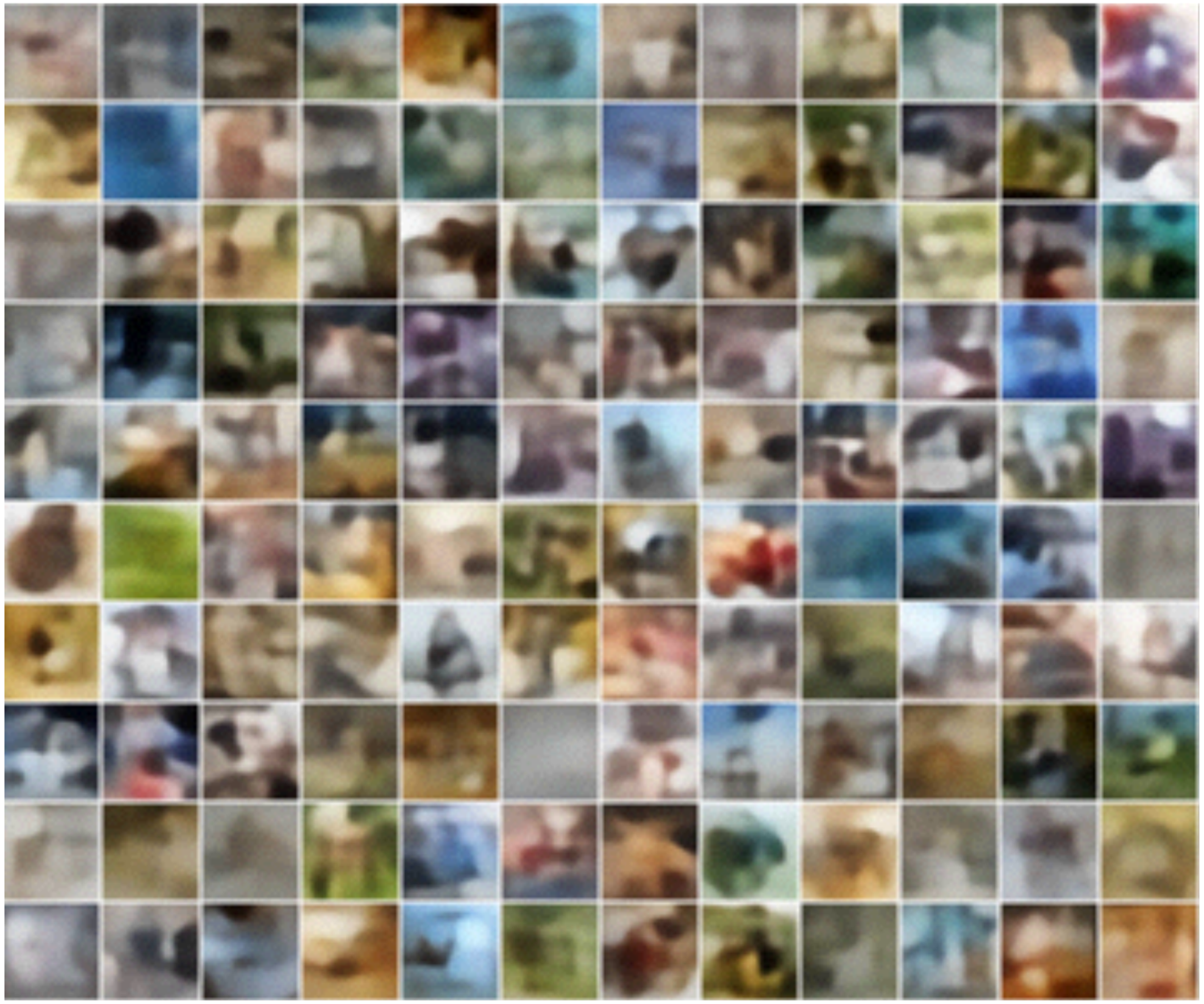
Objective function: $\mathcal{L}(\theta, \phi, x) = -D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| p_\theta(z)\right) + \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right]$



vanilla VAE samples

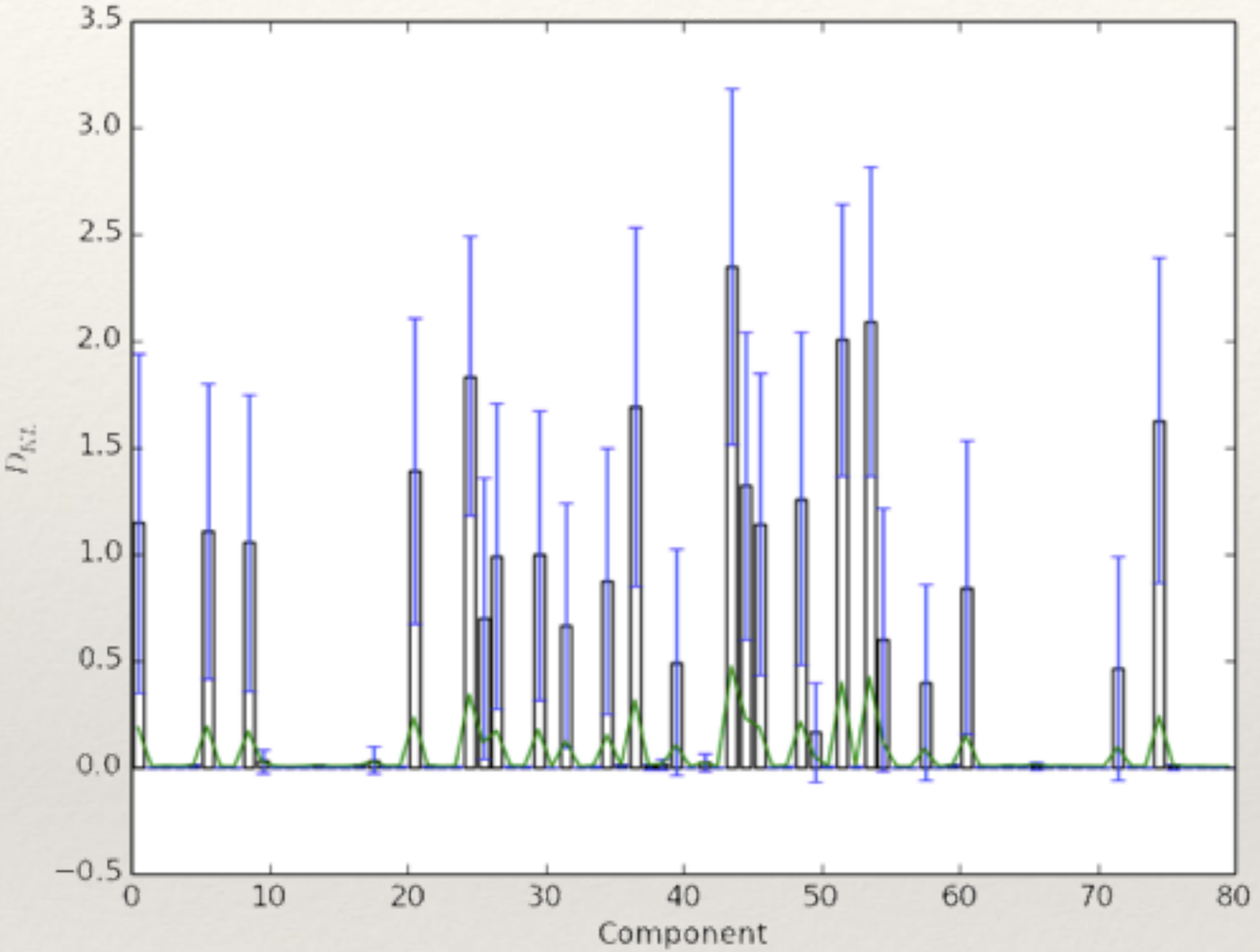
Impressive …
… at the time

Labelled Faces in the Wild (LFW)



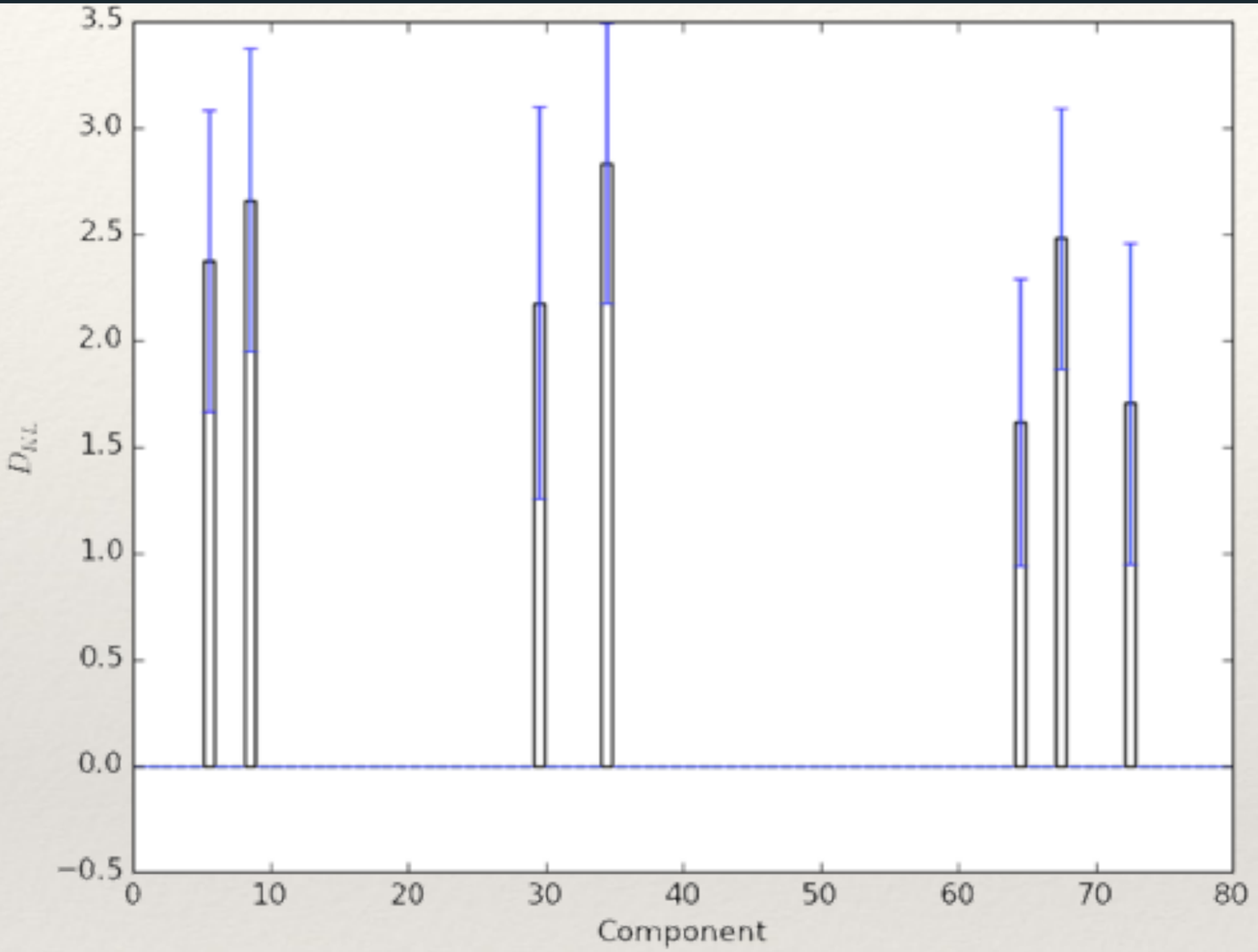
ImageNet (small)

deep encoder/decoder: Some component collapse



Figures from Laurent Dinh & Vincent Dumoulin

deeper encoder/decoder: more component collapse



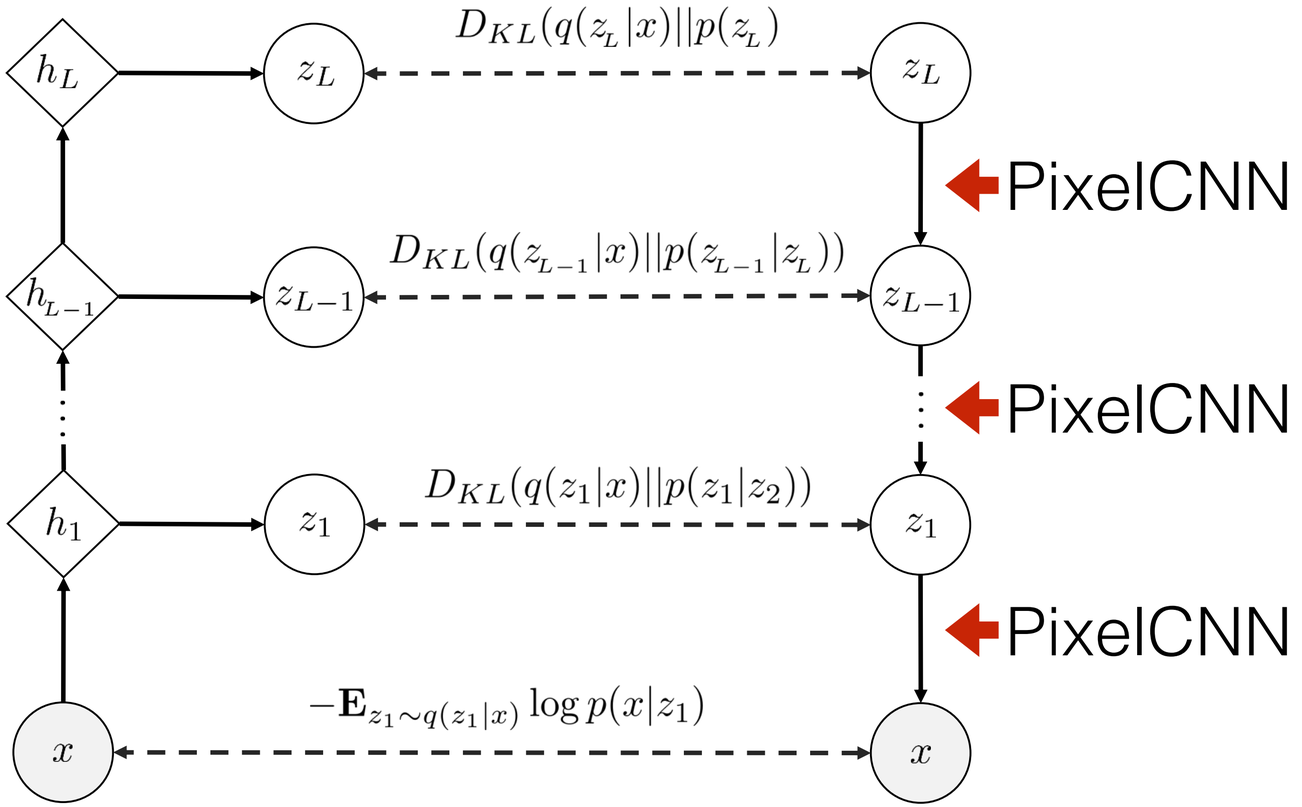
Figures from Laurent Dinh & Vincent Dumoulin

PixelVAE

Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed Adrien Ali Taiga,
Francesco Visin, David Vazquez, Aaron Courville. ICLR 2017

MILA

- Uses a PixelCNN in the VAE decoder to help avoid the blurring caused by the standard VAE assumption of independent pixels.



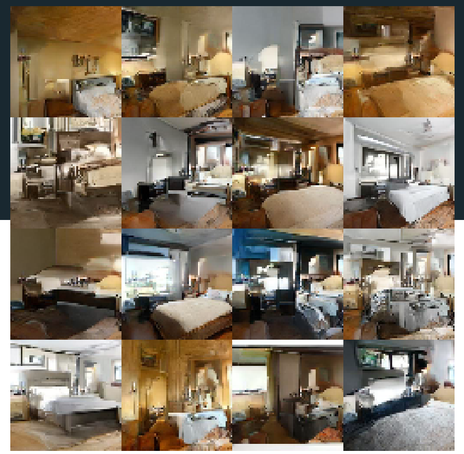
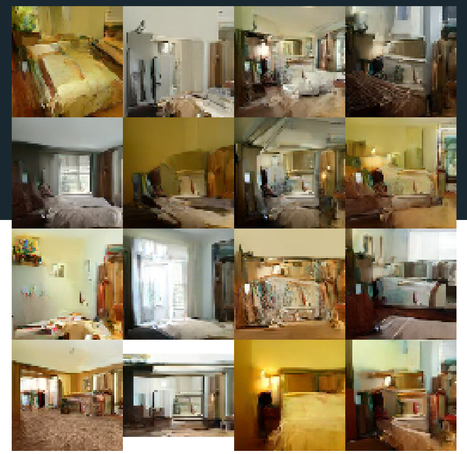
PixelVAE Samples (Gulrajani et al. 2017)

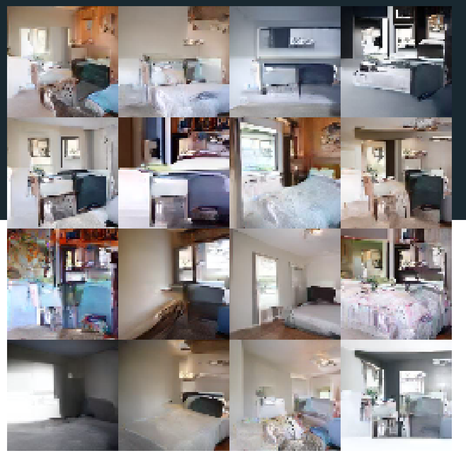
LSUN bedroom scenes (64x64)

ImageNet (64x64)

PixelVAE

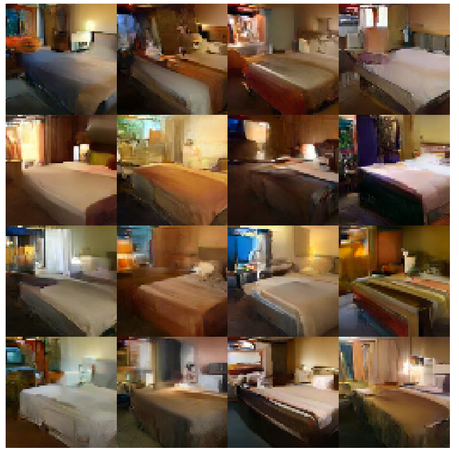
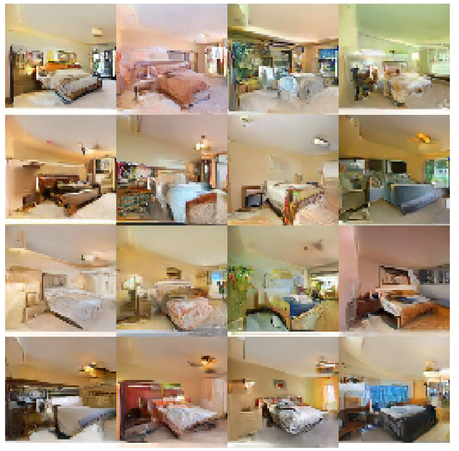
varying only the top-level latent variables

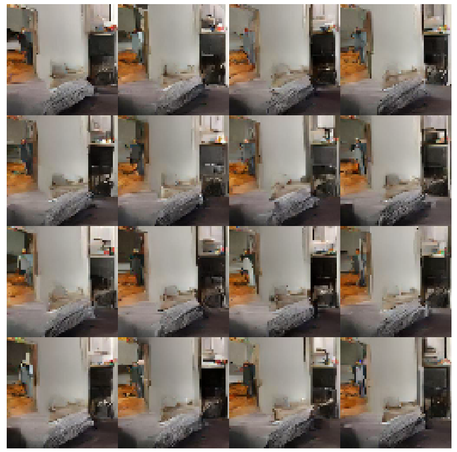


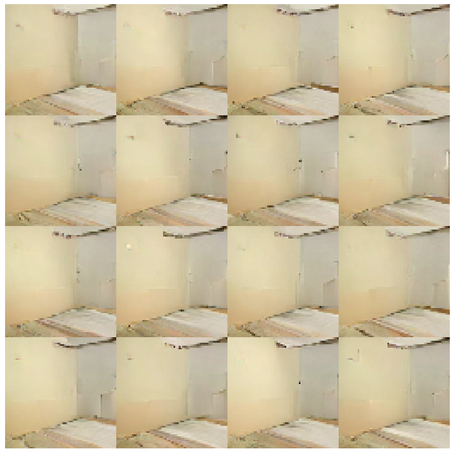
varying only the bottom-level latent variables



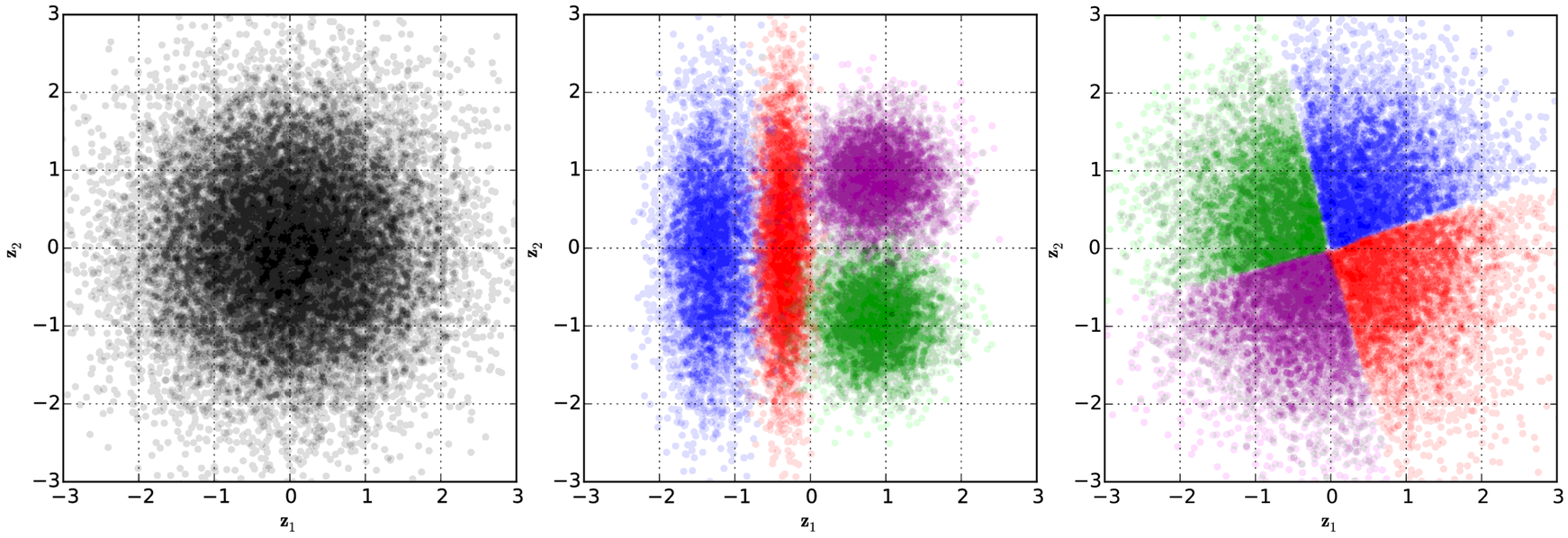



varying only the pixel-level noise



Inverse Autoregressive Flow (Kingma et al., NIPS 2016)



(a) Prior distribution (b) Posteriors in standard VAE (c) Posteriors in VAE with IAF

- Standard VAE posteriors are factorized - limiting how well they can (marginally) fit the prior.
- IAF greatly improves the flexibility of the posterior distributions, and allows for a much better fit between the posteriors and the prior.

Normalizing Flows (Rezende and Mohamed, 2015)

- **Normalizing flows**: the transformation of a probability density through a sequence of invertible mappings.

- Transformation of random variables: $\boldsymbol{z}' = \boldsymbol{f}(\boldsymbol{z})$, $\boldsymbol{f}^{-1}(\boldsymbol{z}') = \boldsymbol{z}$
  For invertible functions:

$$q(\boldsymbol{z}') = q(\boldsymbol{z}) \left| \det \frac{\partial \boldsymbol{f}^{-1}}{\partial \boldsymbol{z}'} \right| = q(\boldsymbol{z}) \left| \det \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}} \right|^{-1}$$

by the Inverse Function Theorem

- Chaining together a sequence: $\boldsymbol{z}_K = \boldsymbol{f}_K \circ \boldsymbol{f}_{K-1} \circ \cdots \circ \boldsymbol{f}_2 \circ \boldsymbol{f}_1(\boldsymbol{z}_0)$

$$\log q_K(\boldsymbol{z}_K) = \log q_0(\boldsymbol{z}_0) - \sum_{k=1}^{K} \log \left| \det \frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{z}_k} \right|$$

Normalizing Flows (Rezende and Mohamed, 2015)

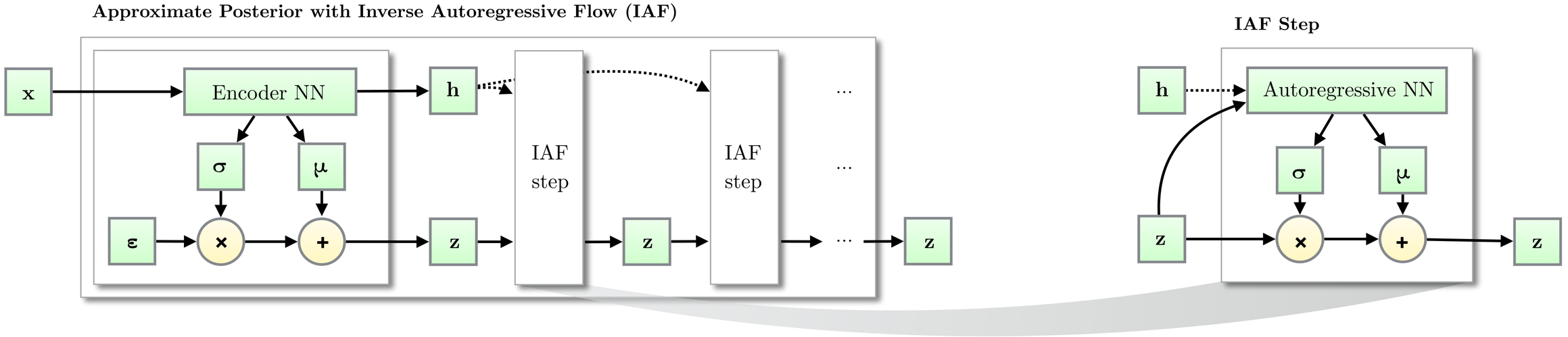
- **Law of the unconscious statistician**: expectations w.r.t. the transformed density $q_K(\boldsymbol{z}_K)$ can be written as expectations w.r.t. the original $q_0(\boldsymbol{z}_0)$. For $\boldsymbol{z}_K = \boldsymbol{f}_K \circ \boldsymbol{f}_{K-1} \circ \cdots \circ \boldsymbol{f}_2 \circ \boldsymbol{f}_1(\boldsymbol{z}_0)$,

$$\mathbb{E}_{q_K}\left[g(\boldsymbol{z}_K)\right] = \mathbb{E}_{q_0}\left[g(\boldsymbol{f}_K \circ \boldsymbol{f}_{K-1} \circ \cdots \circ \boldsymbol{f}_2 \circ \boldsymbol{f}_1(\boldsymbol{z}_0))\right]$$

- The variational lower bound:

$$\begin{aligned}
\mathcal{L}(\theta, \phi, \boldsymbol{x}) &= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}, \boldsymbol{z}) - \log q_\phi(\boldsymbol{z} \mid \boldsymbol{x})\right] \\
&= \mathbb{E}_{q_K(\boldsymbol{z}_K)}\left[\log p(\boldsymbol{x}, \boldsymbol{z}_K) - \log q_K(\boldsymbol{z}_K)\right] \\
&= \mathbb{E}_{q_0(\boldsymbol{z}_0)}\left[\log p(\boldsymbol{x}, \boldsymbol{z}_K) - \log q_0(\boldsymbol{z}_0) + \sum_{k=1}^{K} \log \left|\det \frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{z}_{k-1}}\right|\right]
\end{aligned}$$

VAE-IAF: Building rich posteriors via a sequence of autoregressive latent variables



$$\mathbf{z}_0 = \boldsymbol{\mu}_0 + \boldsymbol{\sigma}_0 \odot \boldsymbol{\epsilon} \qquad \text{random sample } \boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$$

$$\mathbf{z}_t = \boldsymbol{\mu}_t + \boldsymbol{\sigma}_t \odot \mathbf{z}_{t-1}$$

- For IAF, if $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are autoregressive functions of $\boldsymbol{z}_{t-1}$:

$$\log q(\mathbf{z}_T|\mathbf{x}) = \log q(\mathbf{z}_0|\mathbf{x}) - \sum_{t=1}^{T} \log \det \left| \frac{d\mathbf{z}_t}{d\mathbf{z}_{t-1}} \right| \quad \Rightarrow \quad \log q(\mathbf{z}_T|\mathbf{x}) = -\sum_{i=1}^{D} \left(\tfrac{1}{2}\epsilon_i^2 + \tfrac{1}{2}\log(2\pi) + \sum_{t=0}^{T} \log \sigma_{t,i} \right)$$

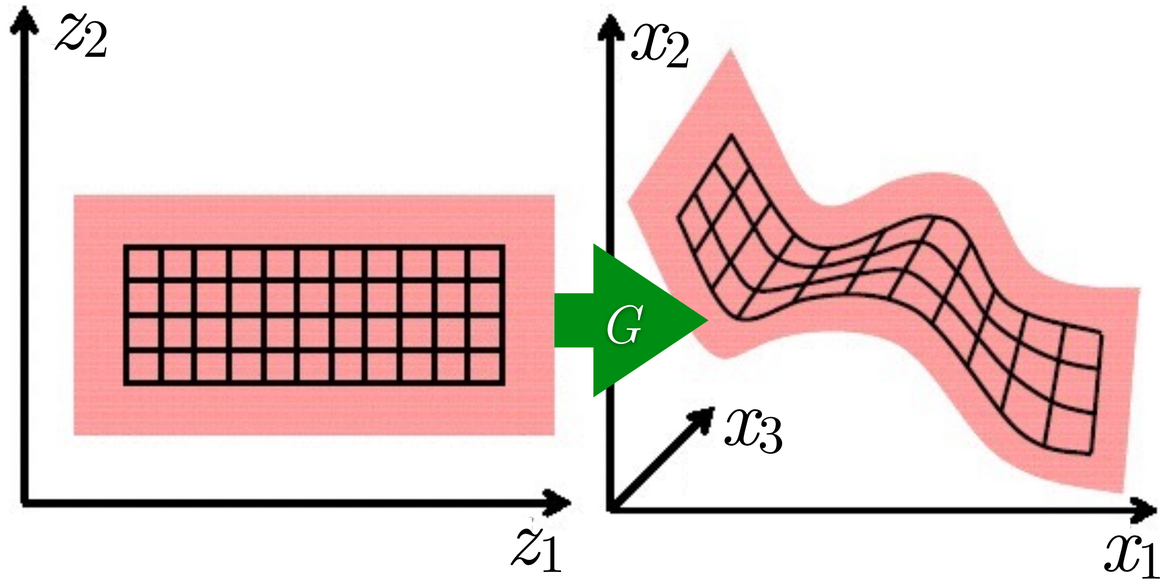
VAE-IAF: MNIST log likelihood

Table 1: Generative modeling results on the dynamically sampled binarized MNIST version used in previous publications (Burda et al., 2015). Shown are averages; the number between brackets are standard deviations across 5 optimization runs. The right column shows an importance sampled estimate of the marginal likelihood for each model with 128 samples. Best previous results are reproduced in the first segment: [1]: (Salimans et al., 2014) [2]: (Burda et al., 2015) [3]: (Kaae Sønderby et al., 2016) [4]: (Tran et al., 2015)

Model	VLB	$\log p(\mathbf{x}) \approx$
Convolutional VAE + HVI [1]	-83.49	-81.94
DLGM 2hl + IWAE [2]		-82.90
LVAE [3]		-81.74
DRAW + VGP [4]	-79.88	
Diagonal covariance	-84.08 ($\pm$ 0.10)	-81.08 ($\pm$ 0.08)
IAF (Depth $= 2$, Width $= 320$)	-82.02 ($\pm$ 0.08)	-79.77 ($\pm$ 0.06)
IAF (Depth $= 2$, Width $= 1920$)	-81.17 ($\pm$ 0.08)	-79.30 ($\pm$ 0.08)
IAF (Depth $= 4$, Width $= 1920$)	-80.93 ($\pm$ 0.09)	-79.17 ($\pm$ 0.08)
IAF (Depth $= 8$, Width $= 1920$)	-80.80 ($\pm$ 0.07)	**-79.10** ($\pm$ 0.07)

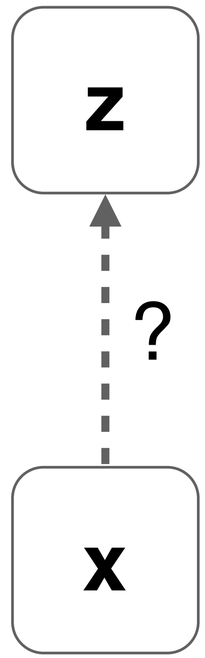
Another way to train a latent variable model

Latent variables

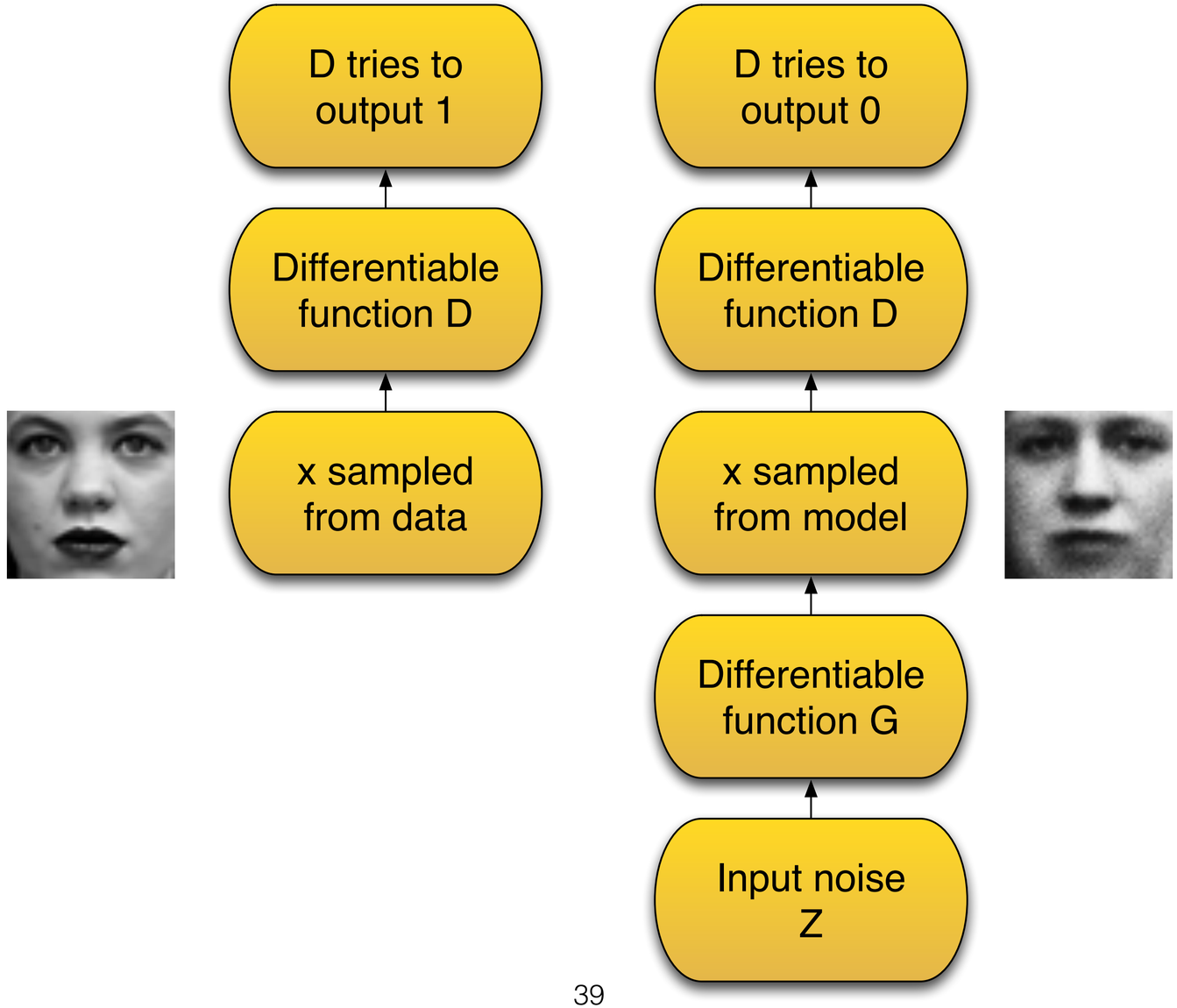


Observed variables

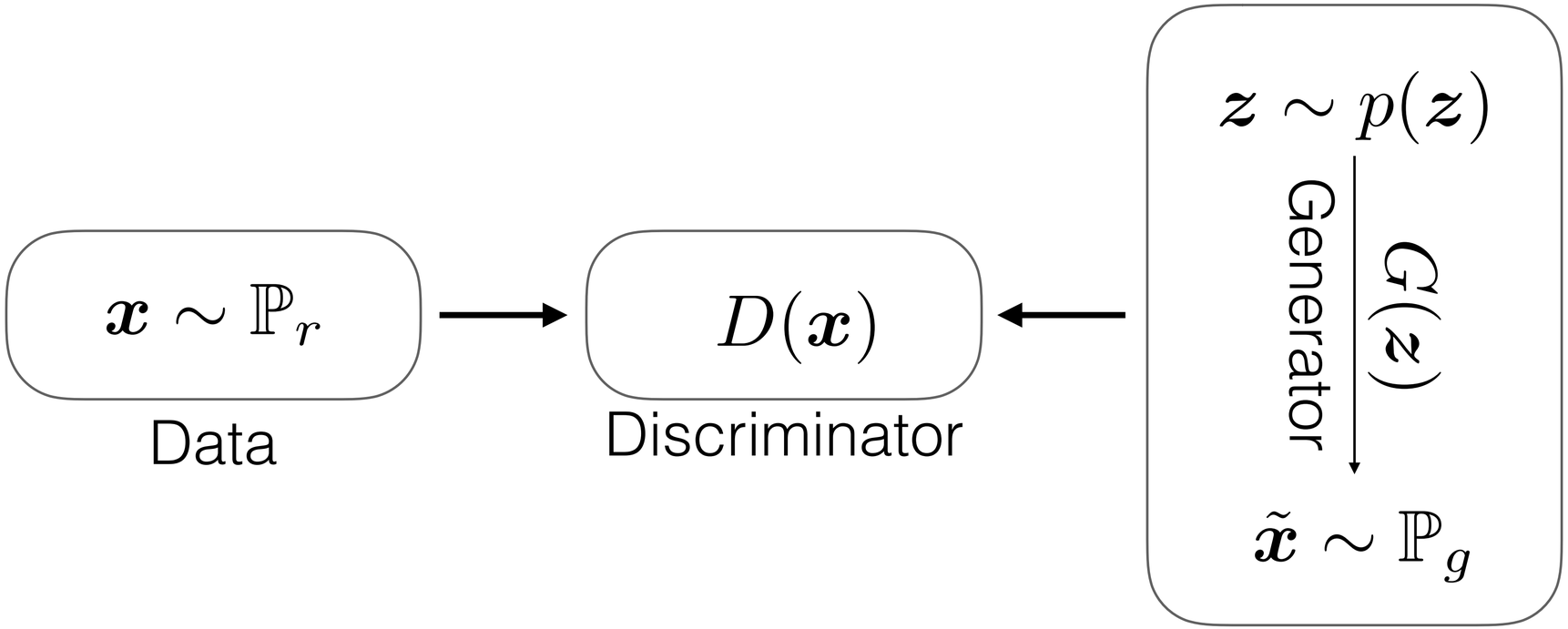
inference



Generative Adversarial Networks



Generative Adversarial Networks



GAN Objective

- Formally, express the game between discriminator D and generator G with the minimax objective:

$$\min_G \max_D \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} [\log(D(\boldsymbol{x}))] + \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} [\log(1 - D(\tilde{\boldsymbol{x}}))].$$

  where:

  - $\mathbb{P}_r$ is the data distribution
  - $\mathbb{P}_g$ is the model distribution implicitly defined by:

$$\tilde{\boldsymbol{x}} = G(\boldsymbol{z}), \quad \boldsymbol{z} \sim p(\boldsymbol{z})$$

  - the generator input $\boldsymbol{z}$ is sampled from some simple noise distribution, (e.g. uniform or Gaussian).

GAN Theory

- Optimal (nonparametric) discriminator:

$$D^*(\boldsymbol{x}) = \frac{p_r(\boldsymbol{x})}{p_r(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

- Under an ideal discriminator, the generator minimizes the Jensen-Shannon divergence between $\mathbb{P}_r$ and $\mathbb{P}_g$.

$$\mathrm{JS}(\mathbb{P}_r \| \mathbb{P}_g) = \mathrm{KL}\left(\mathbb{P}_r \middle\| \frac{\mathbb{P}_r + \mathbb{P}_g}{2}\right) + \mathrm{KL}\left(\mathbb{P}_g \middle\| \frac{\mathbb{P}_r + \mathbb{P}_g}{2}\right)$$

$$\text{where } \mathrm{KL}(\mathbb{P}_r \| \mathbb{P}_g) = \int \log\left(\frac{p_r(x)}{p_g(x)}\right) p_r(x) d\mu(x)$$

GAN Theory … in practice

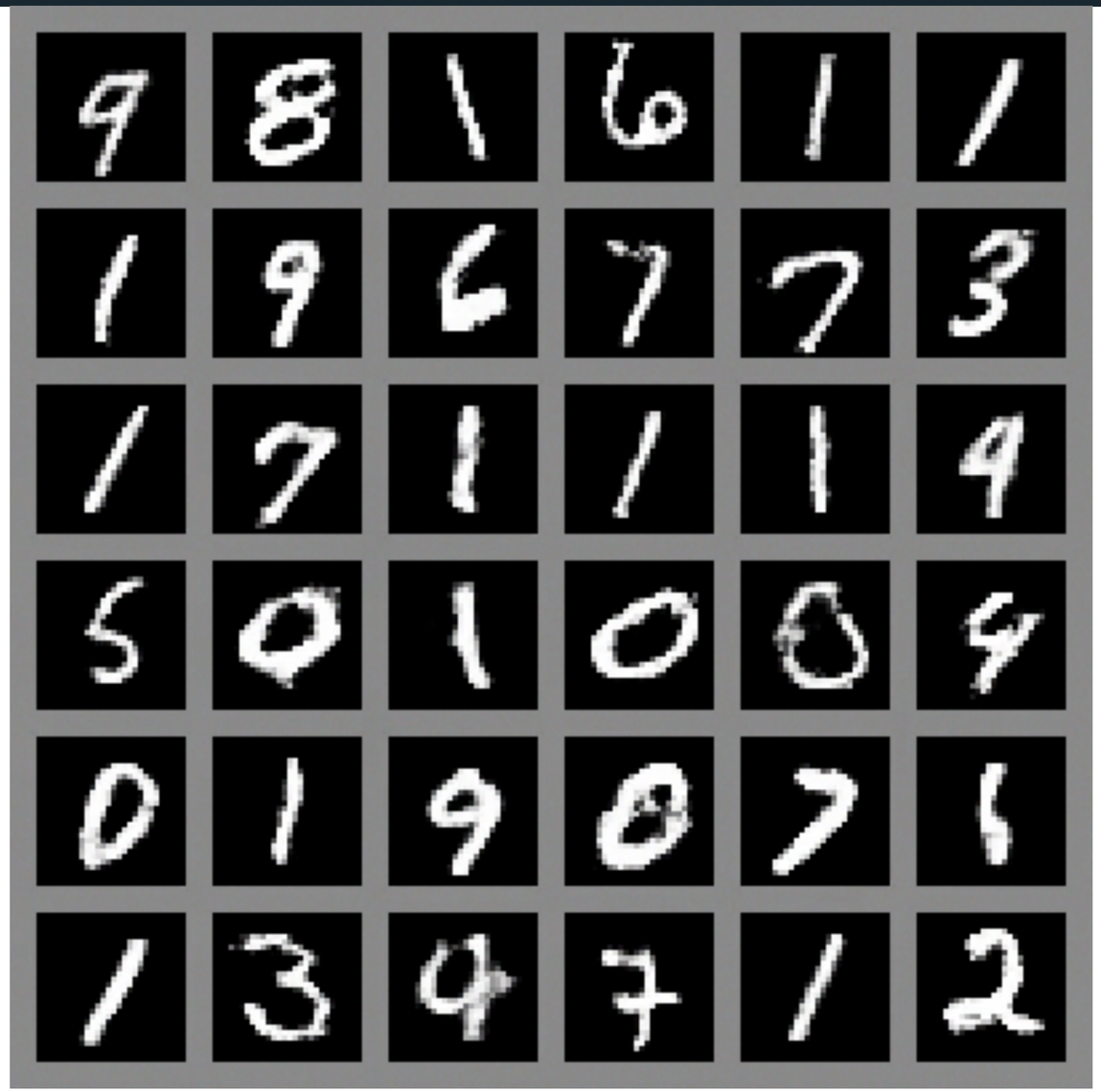
- The minimax objective leads to vanishing gradients as the discriminator saturates.
- In practice, Goodfellow et al (2014) advocate the heuristic training objective:

$$\max_D \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}[\log(D(\boldsymbol{x}))] + \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[\log(1-D(\tilde{\boldsymbol{x}}))].$$

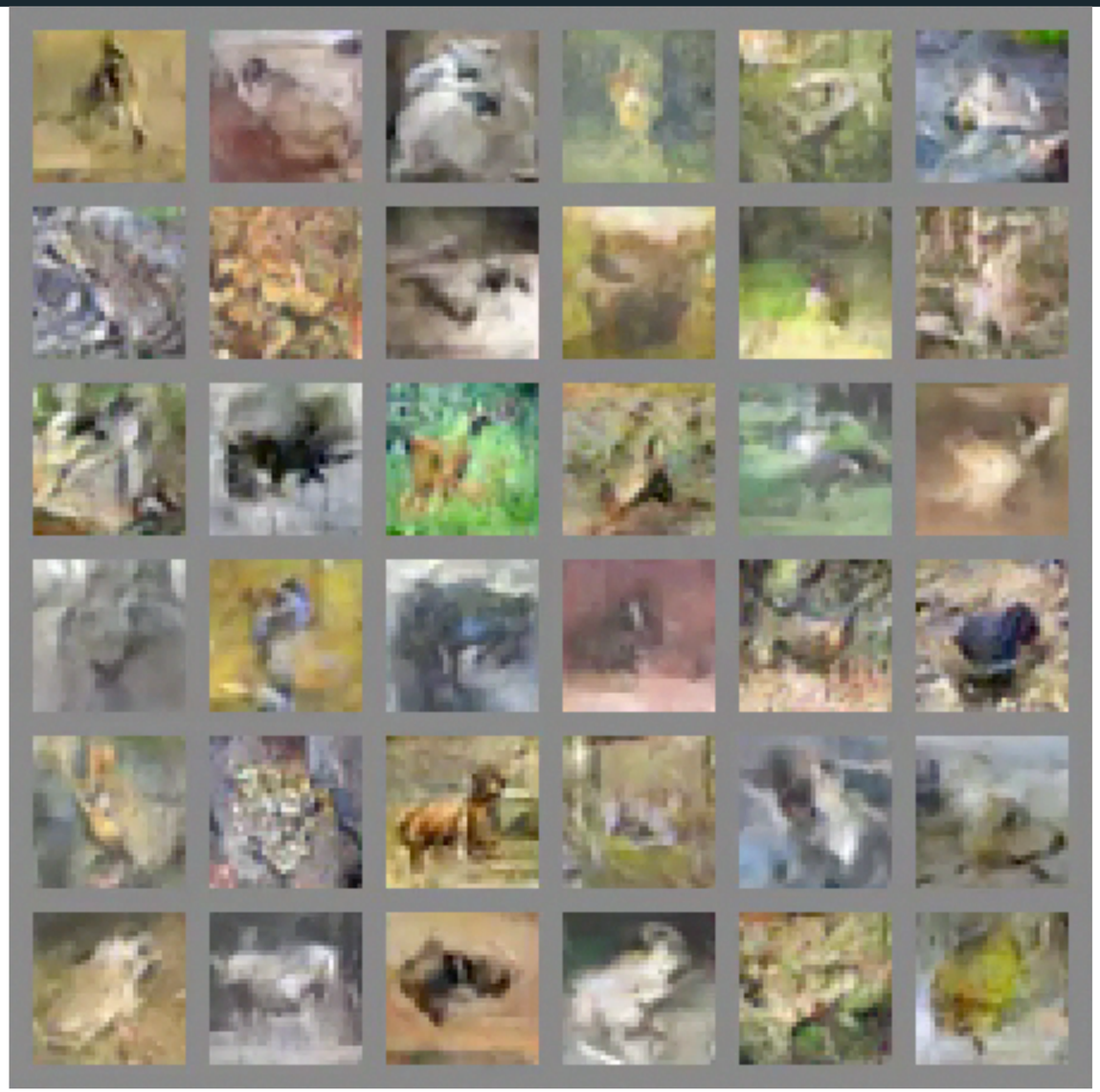
$$\max_G \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[\log(D(\tilde{\boldsymbol{x}}))].$$

- However, this modified loss function can still misbehave in the presence of a good discriminator.

GAN samples



MNIST

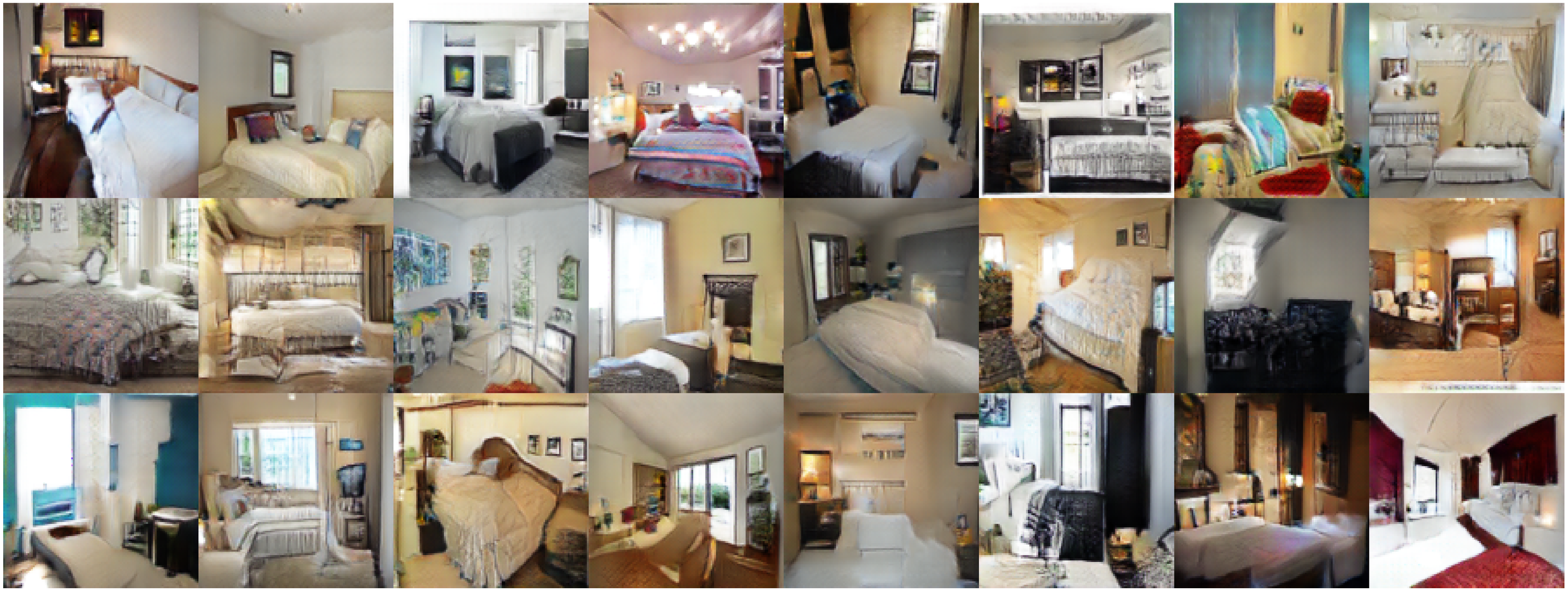


CIFAR-10

Least-Squares GAN

Xudong Mao, Qing Li†, Haoran Xie, Raymond Y.K. Lau and Zhen Wang, ArXiv, Feb. 2017

MILA



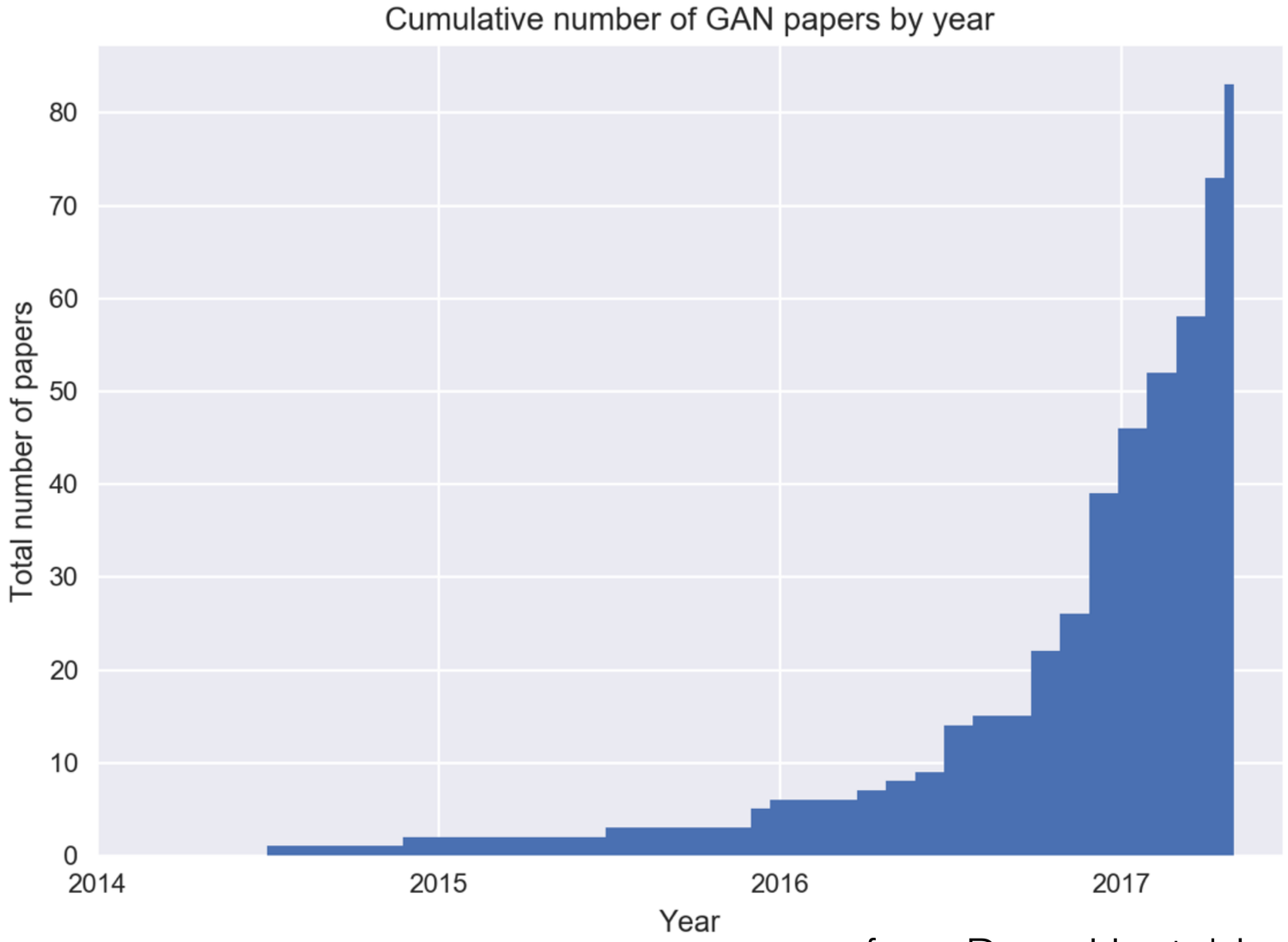
128x128 LSUN bedroom scenes

GAN Zoo

Deep Hunt, blog by Avinash Hindupur

GAN — Generative Adversarial Networks
3D-GAN — Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
acGAN — Face Aging With Conditional Generative Adversarial Networks
AC-GAN — Conditional Image Synthesis With Auxiliary Classifier GANs
AdaGAN — AdaGAN: Boosting Generative Models
AEGAN — Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
AffGAN — Amortised MAP Inference for Image Super-resolution
AL-CGAN — Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
ALI — Adversarially Learned Inference
AMGAN — Generative Adversarial Nets with Labeled Data by Activation Maximization
AnoGAN — Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
ArtGAN — ArtGAN: Artwork Synthesis with Conditional Categorial GANs
b-GAN — b-GAN: Unified Framework of Generative Adversarial Networks
Bayesian GAN — Deep and Hierarchical Implicit Models
BEGAN — BEGAN: Boundary Equilibrium Generative Adversarial Networks
BiGAN — Adversarial Feature Learning
BS-GAN — Boundary-Seeking Generative Adversarial Networks
CGAN — Conditional Generative Adversarial Nets
CCGAN — Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
CatGAN — Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
CoGAN — Coupled Generative Adversarial Networks
Context-RNN-GAN — Contextual RNN-GANs for Abstract Reasoning Diagram Generation
C-RNN-GAN — C-RNN-GAN: Continuous recurrent neural networks with adversarial training
CS-GAN — Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
CVAE-GAN — CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
CycleGAN — Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
DTN — Unsupervised Cross-Domain Image Generation
DCGAN — Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
DiscoGAN — Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
DR-GAN — Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
DualGAN — DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
EBGAN — Energy-based Generative Adversarial Network
f-GAN — f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
GAWWN — Learning What and Where to Draw
GoGAN — Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
GP-GAN — GP-GAN: Towards Realistic High-Resolution Image Blending
IAN — Neural Photo Editing with Introspective Adversarial Networks
iGAN — Generative Visual Manipulation on the Natural Image Manifold
IcGAN — Invertible Conditional GANs for image editing
ID-CGAN- Image De-raining Using a Conditional Generative Adversarial Network
Improved GAN — Improved Techniques for Training GANs
InfoGAN — InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
LAGAN — Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
LAPGAN — Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks
LR-GAN — LR-GAN: Layered Recursive Generative Adversarial Networks for Image Generation
LSGAN — Least Squares Generative Adversarial Networks
LS-GAN — Loss-Sensitive Generative Adversarial Networks on Lipschitz Densities
MGAN — Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks
MAGAN — MAGAN: Margin Adaptation for Generative Adversarial Networks
MAD-GAN — Multi-Agent Diverse Generative Adversarial Networks
MalGAN — Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN
MaliGAN — Maximum-Likelihood Augmented Discrete Generative Adversarial Networks
MARTA-GAN — Deep Unsupervised Representation Learning for Remote Sensing Images
McGAN — McGan: Mean and Covariance Feature Matching GAN
MDGAN — Mode Regularized Generative Adversarial Networks
MedGAN — Generating Multi-label Discrete Electronic Health Records using Generative Adversarial Networks
MIX+GAN — Generalization and Equilibrium in Generative Adversarial Nets (GANs)
MPM-GAN — Message Passing Multi-Agent GANs
MV-BiGAN — Multi-view Generative Adversarial Networks
pix2pix — Image-to-Image Translation with Conditional Adversarial Networks
PPGN — Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space
PrGAN — 3D Shape Induction from 2D Views of Multiple Objects
RenderGAN — RenderGAN: Generating Realistic Labeled Data
RTT-GAN — Recurrent Topic-Transition GAN for Visual Paragraph Generation
SGAN — Stacked Generative Adversarial Networks
SGAN — Texture Synthesis with Spatial Generative Adversarial Networks
SAD-GAN — SAD-GAN: Synthetic Autonomous Driving using Generative Adversarial Networks
SalGAN — SalGAN: Visual Saliency Prediction with Generative Adversarial Networks
SEGAN — SEGAN: Speech Enhancement Generative Adversarial Network
SeGAN — SeGAN: Segmenting and Generating the Invisible
SeqGAN — SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient
SimGAN — Learning from Simulated and Unsupervised Images through Adversarial Training
SketchGAN — Adversarial Training For Sketch Retrieval
SL-GAN — Semi-Latent GAN: Learning to generate and modify facial images from attributes
Softmax-GAN — Softmax GAN
SRGAN — Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network
S^2GAN — Generative Image Modeling using Style and Structure Adversarial Networks
SSL-GAN — Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
StackGAN — StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks
TGAN — Temporal Generative Adversarial Nets
TAC-GAN — TAC-GAN — Text Conditioned Auxiliary Classifier Generative Adversarial Network
TP-GAN — Beyond Face Rotation: Global and Local Perception GAN for Photorealistic and Identity Preserving Frontal View Synthesis
Triple-GAN — Triple Generative Adversarial Nets
Unrolled GAN — Unrolled Generative Adversarial Networks
VGAN — Generating Videos with Scene Dynamics
VGAN — Generative Adversarial Networks as Variational Training of Energy Based Models
VAE-GAN — Autoencoding beyond pixels using a learned similarity metric
VariGAN — Multi-View Image Generation from a Single-View
ViGAN — Image Generation and Editing with Variational Info Generative AdversarialNetworks
WGAN — Wasserstein GAN
WGAN-GP — Improved Training of Wasserstein GANs
WaterGAN — WaterGAN: Unsupervised Generative Network to Enable Real-time Color Correction of Monocular Underwater Images

An explo-GAN of papers



from Deep Hunt, blog by Avinash Hindupur

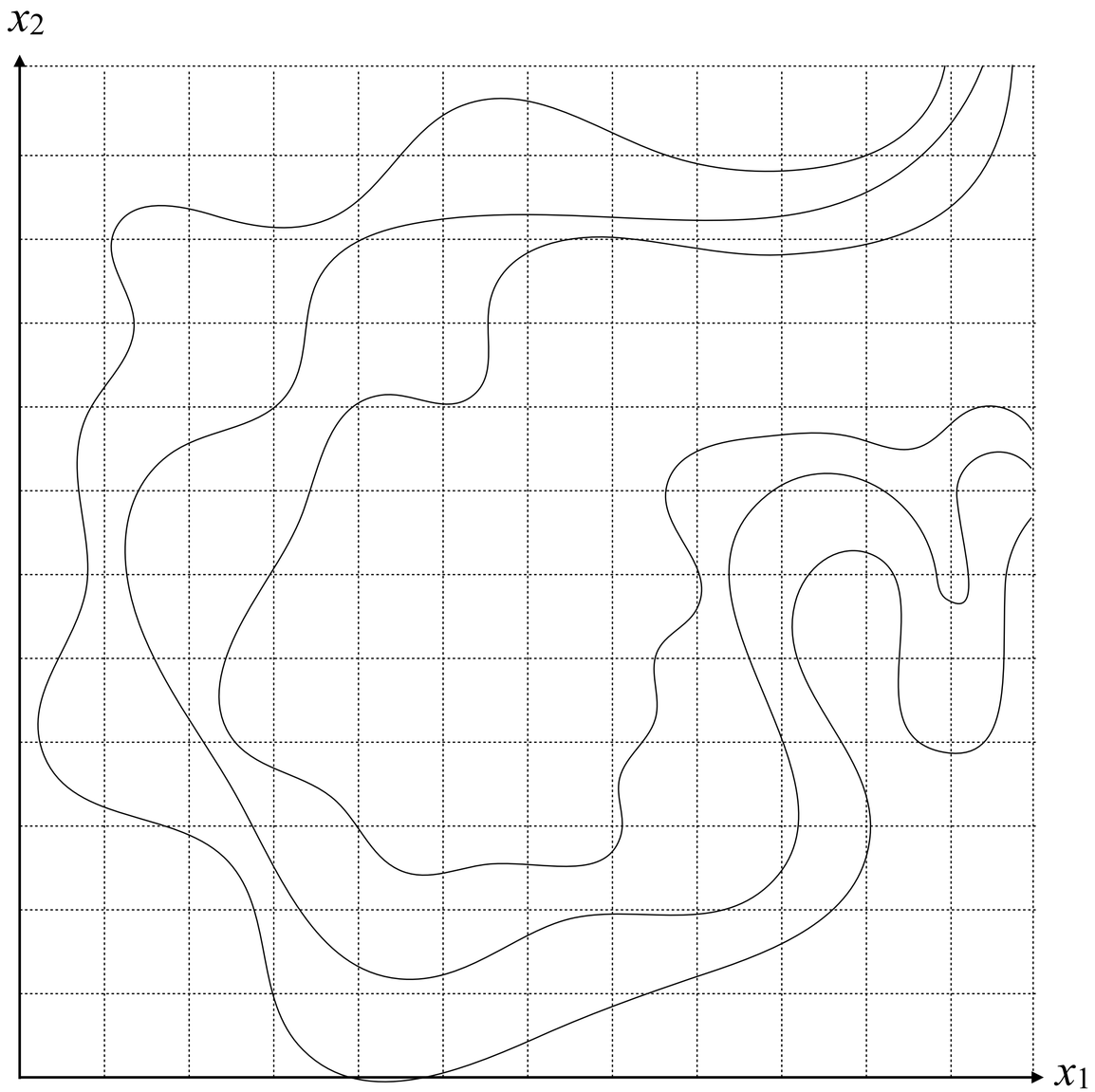
DCGAN samples (Radford, Metz and Chintala; 2016)

MILA

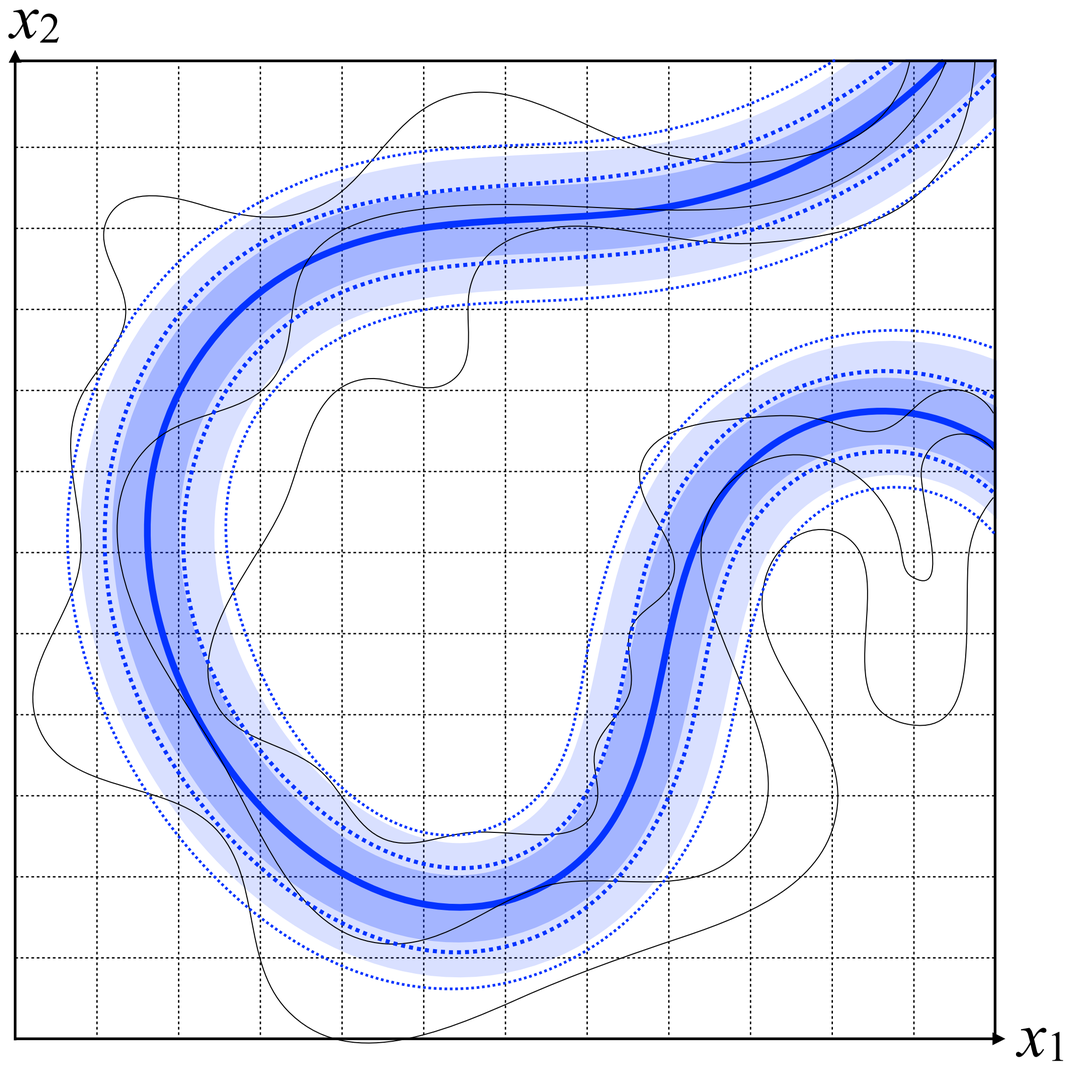
Z-space interpolations

LSUN bedroom scenes

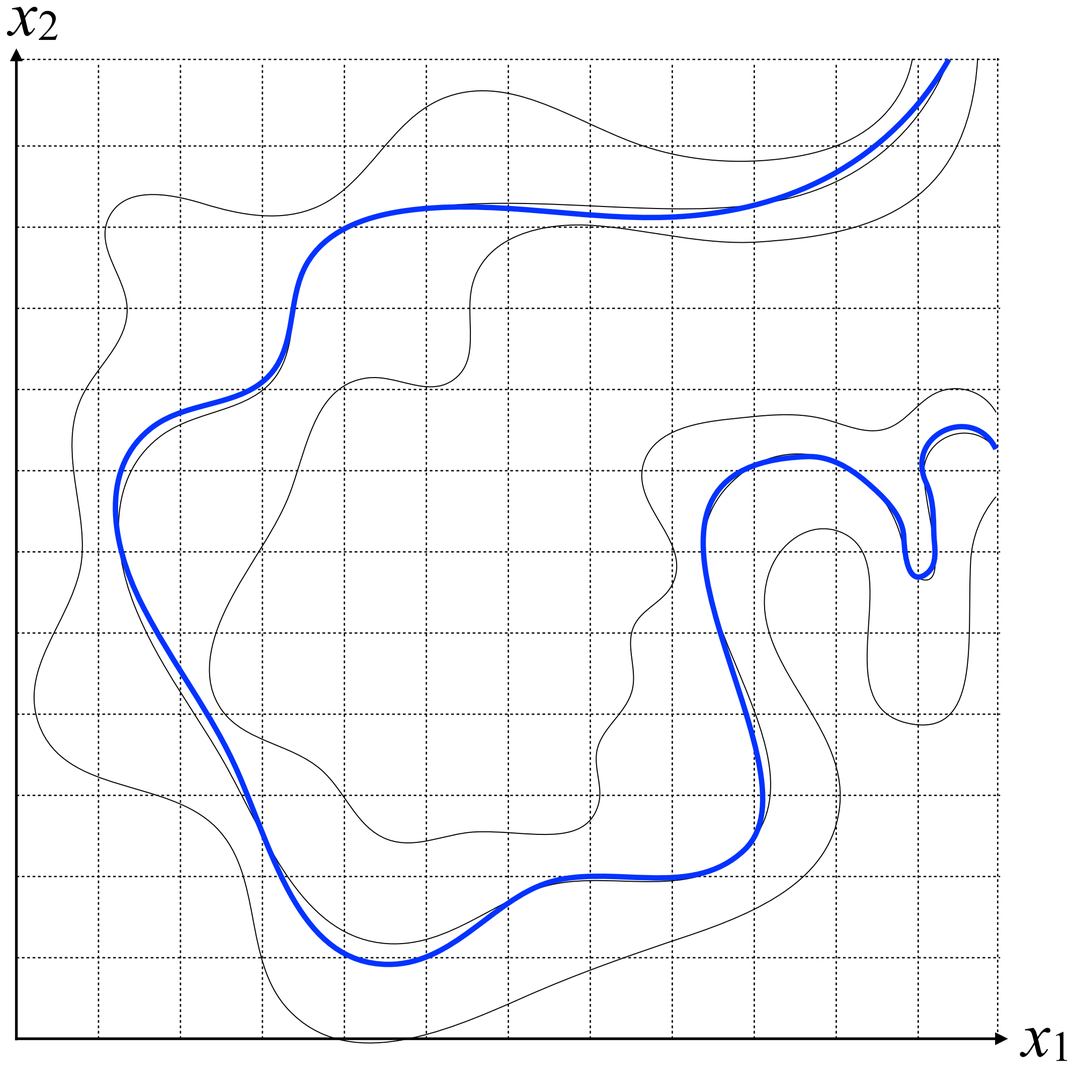
Cartoon of the Image manifold



What makes GANs special?

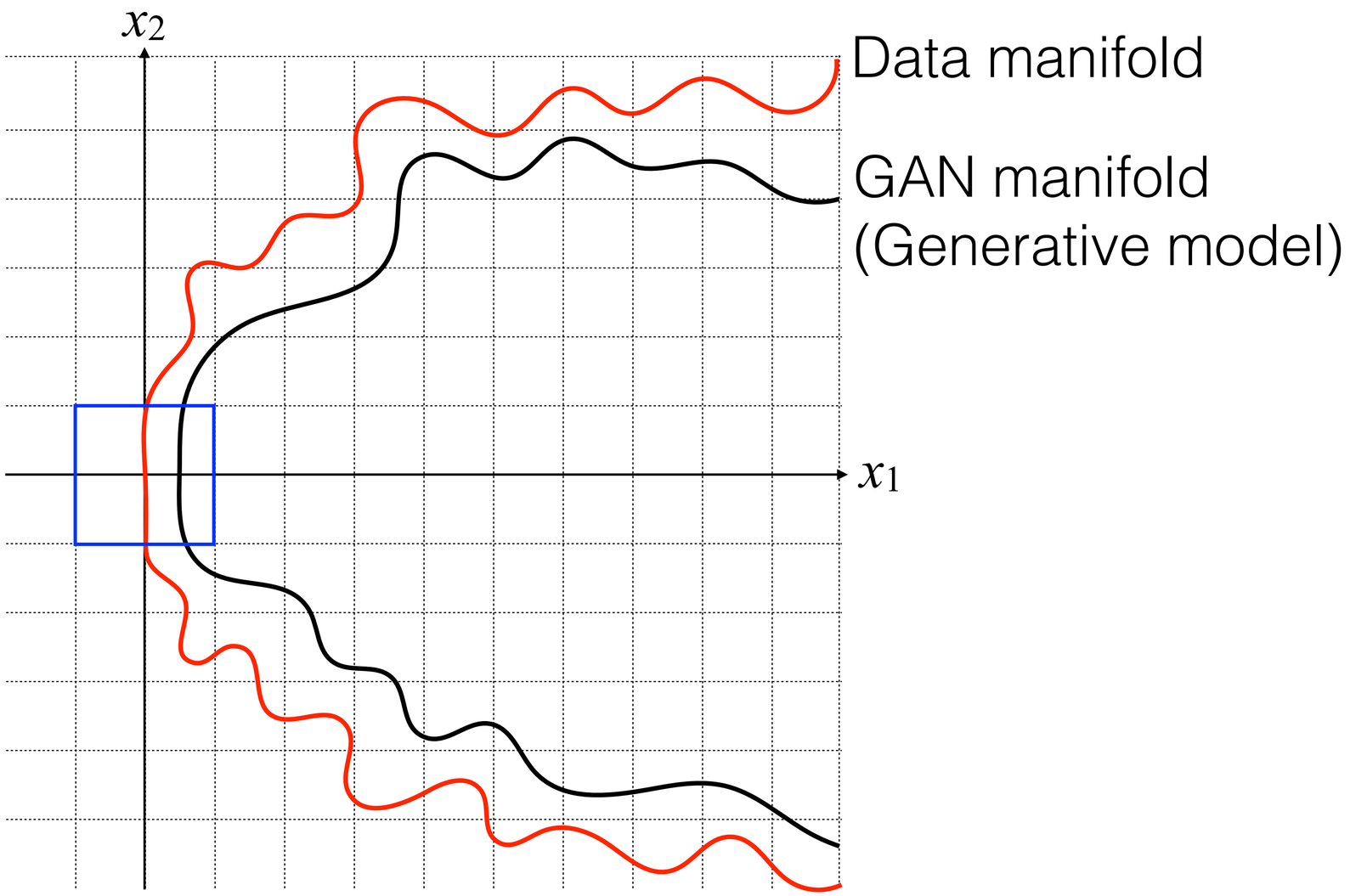


more traditional max-likelihood approach



GAN

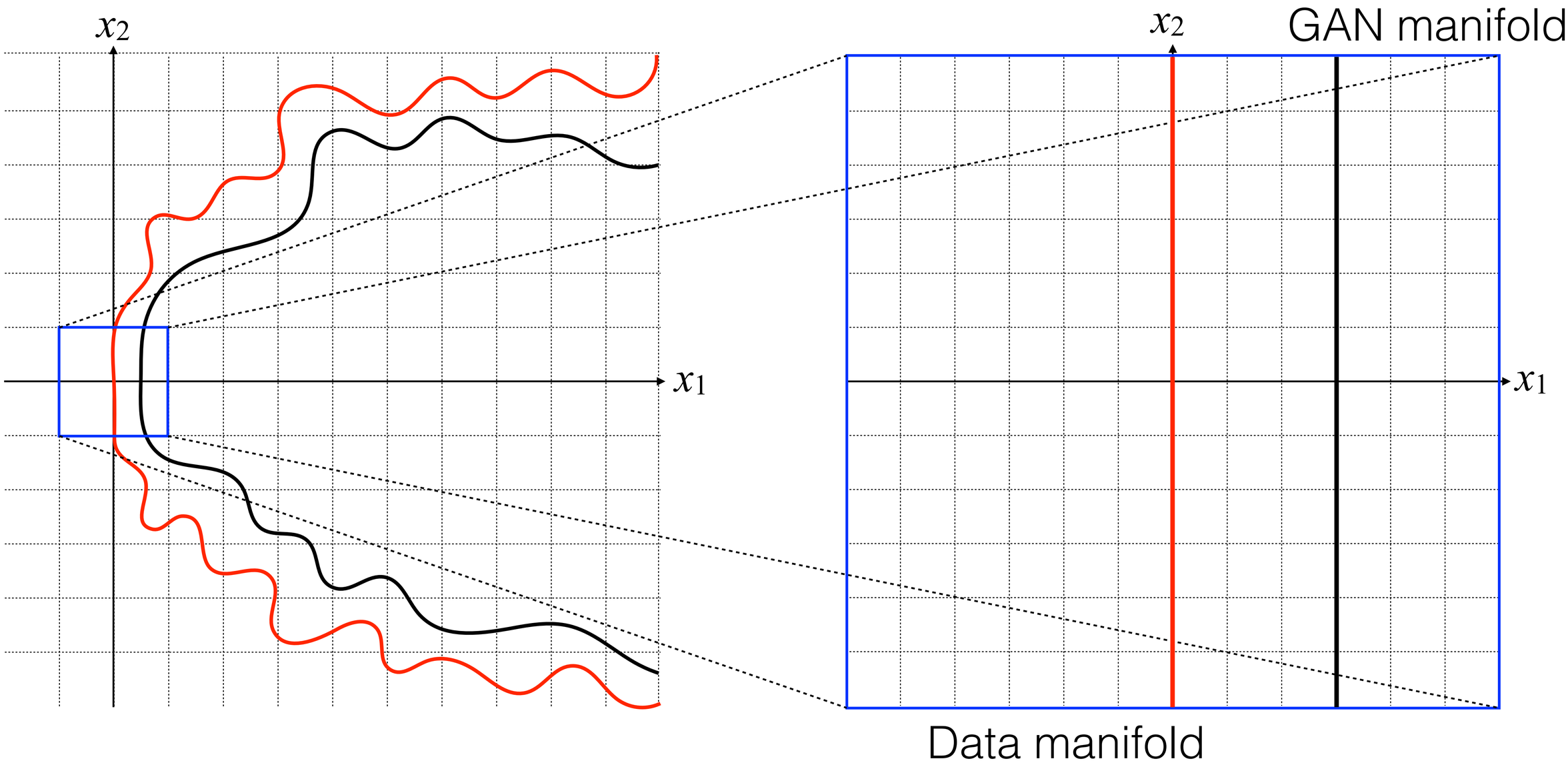
Training a GAN: Distances between Manifolds



Data manifold

GAN manifold
(Generative model)

Training a GAN: Distances between Manifolds

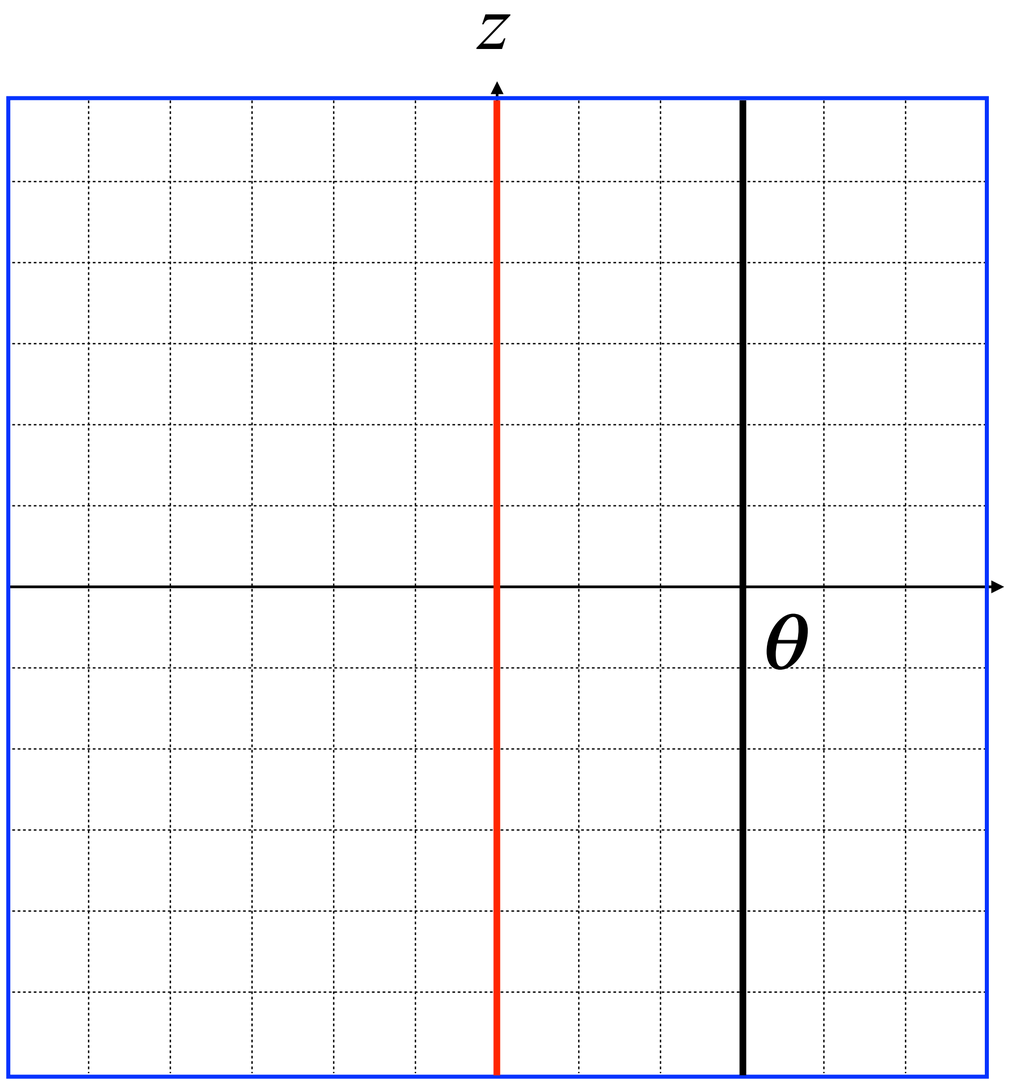


Jensen-Shannon Divergence

$$\mathrm{JS}(\mathbb{P}_r \| \mathbb{P}_g) = \mathrm{KL}\left(\mathbb{P}_r \middle\| \frac{\mathbb{P}_r + \mathbb{P}_g}{2}\right) + \mathrm{KL}\left(\mathbb{P}_g \middle\| \frac{\mathbb{P}_r + \mathbb{P}_g}{2}\right)$$

- What is the JS divergence in this simple case?

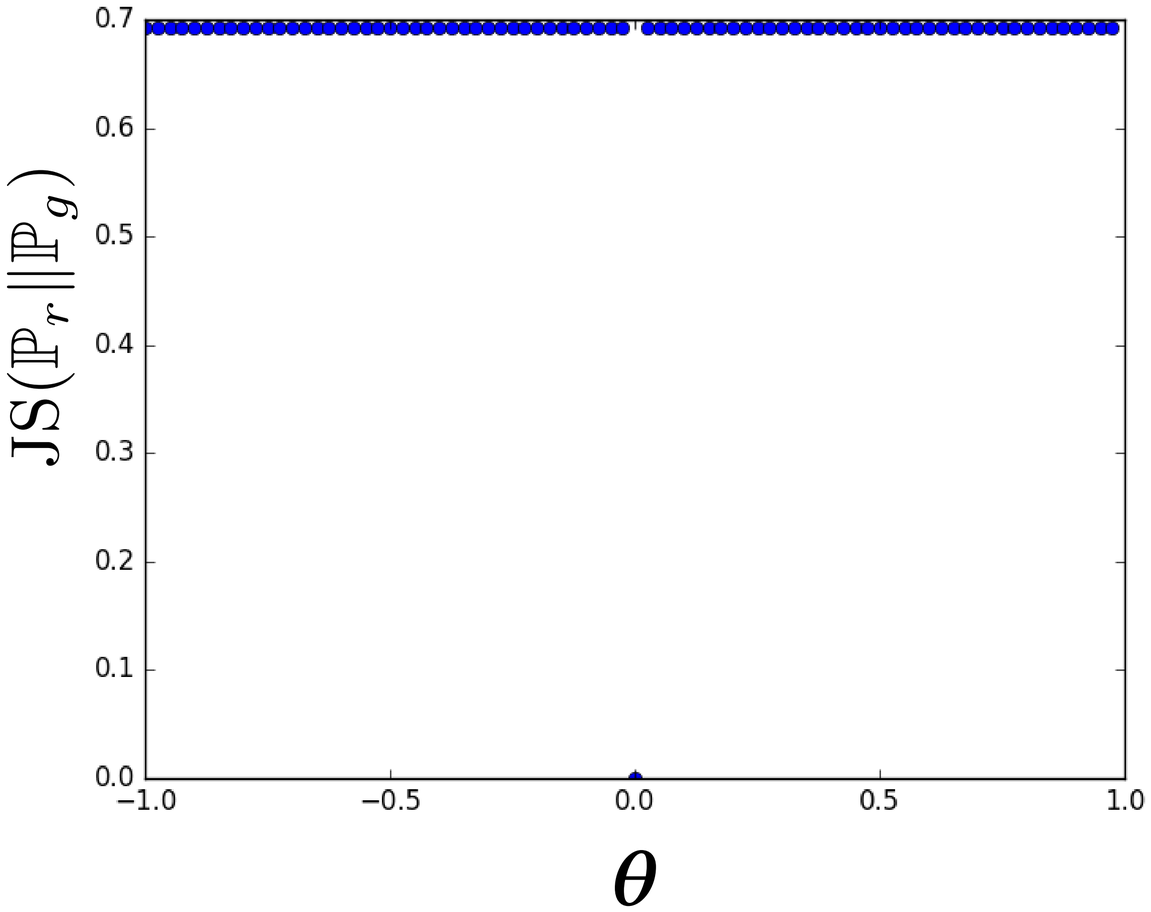
$$\mathrm{JS}(\mathbb{P}_r \| \mathbb{P}_g) = \begin{cases} \log 2 & \text{if } \theta \neq 0, \\ 0 & \text{if } \theta = 0, \end{cases}$$

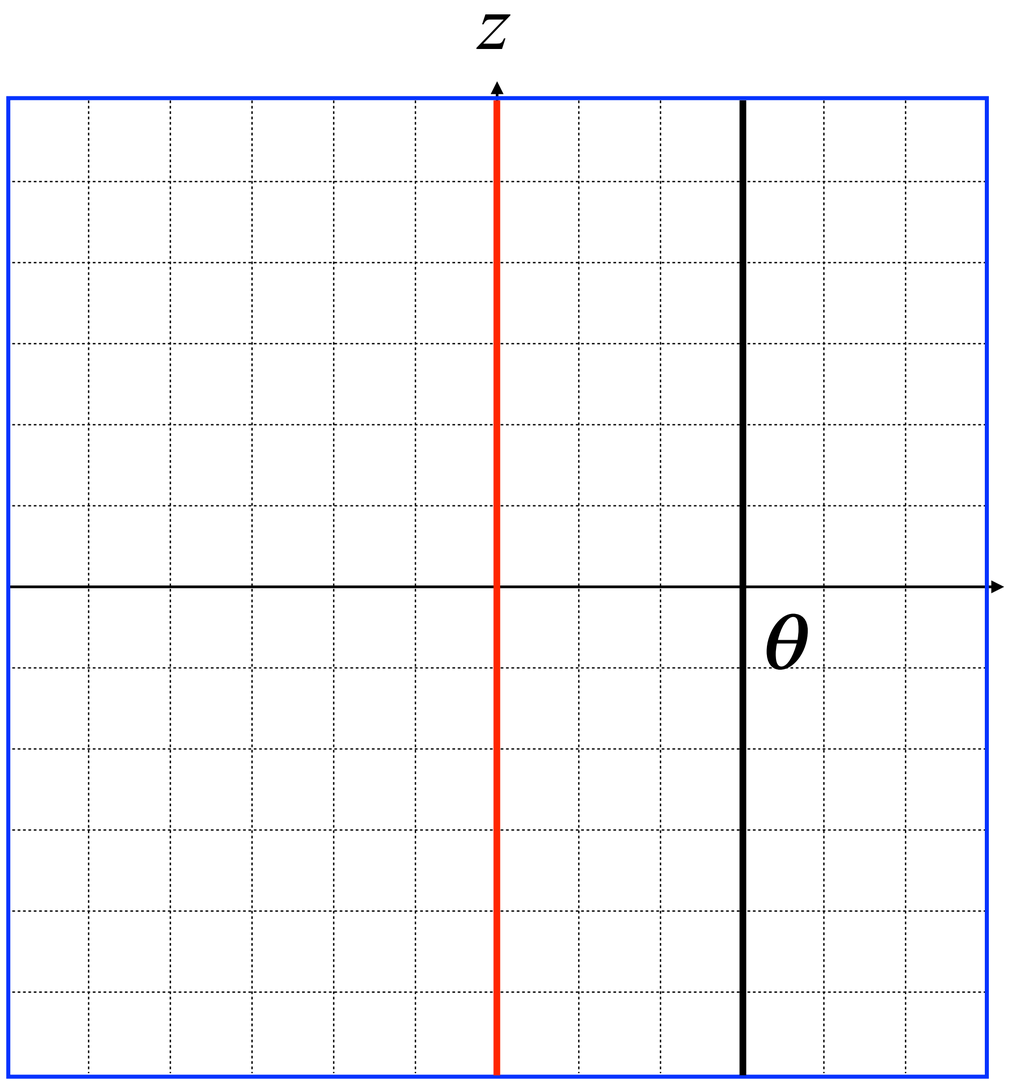


Example from Arjovsky et al. 2017

Jensen-Shannon Divergence

$$\mathrm{JS}(\mathbb{P}_r \| \mathbb{P}_g) = \begin{cases} \log 2 & \text{if } \theta \neq 0, \\ 0 & \text{if } \theta = 0, \end{cases}$$





Example from Arjovsky et al. 2017

Earth-Movers Distance

- JS divergence is not a useful learning signal to train GANs.
- An alternative: Earth-Mover (also called Wasserstein-1) distance.

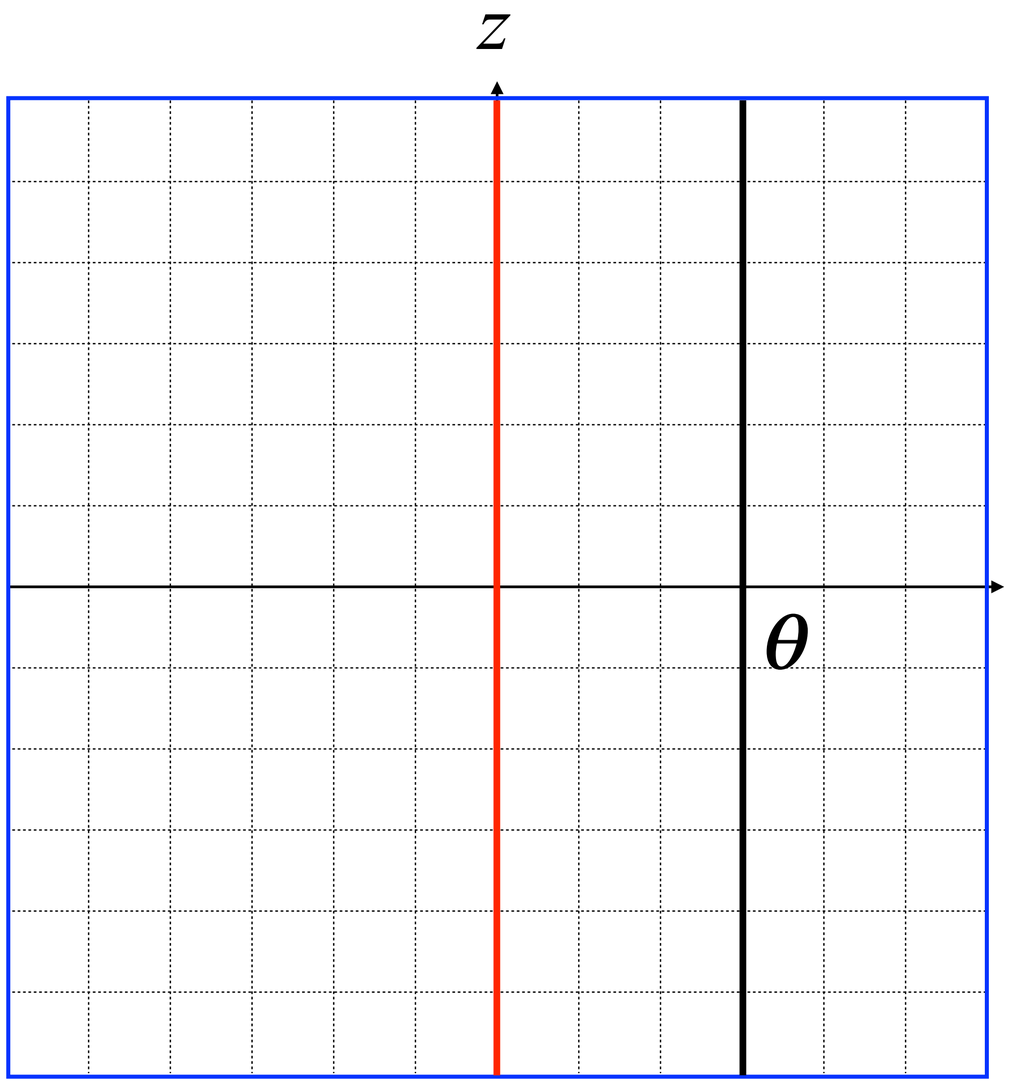
$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma} \left[\|x - y\| \right]$$

  - Minimum cost of transporting mass to transform the distribution $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$.
  - The EM distance is continuous everywhere and differentiable almost everywhere (under mild assumptions).

Wasserstein Distance

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma} \left[\|x - y\| \right]$$

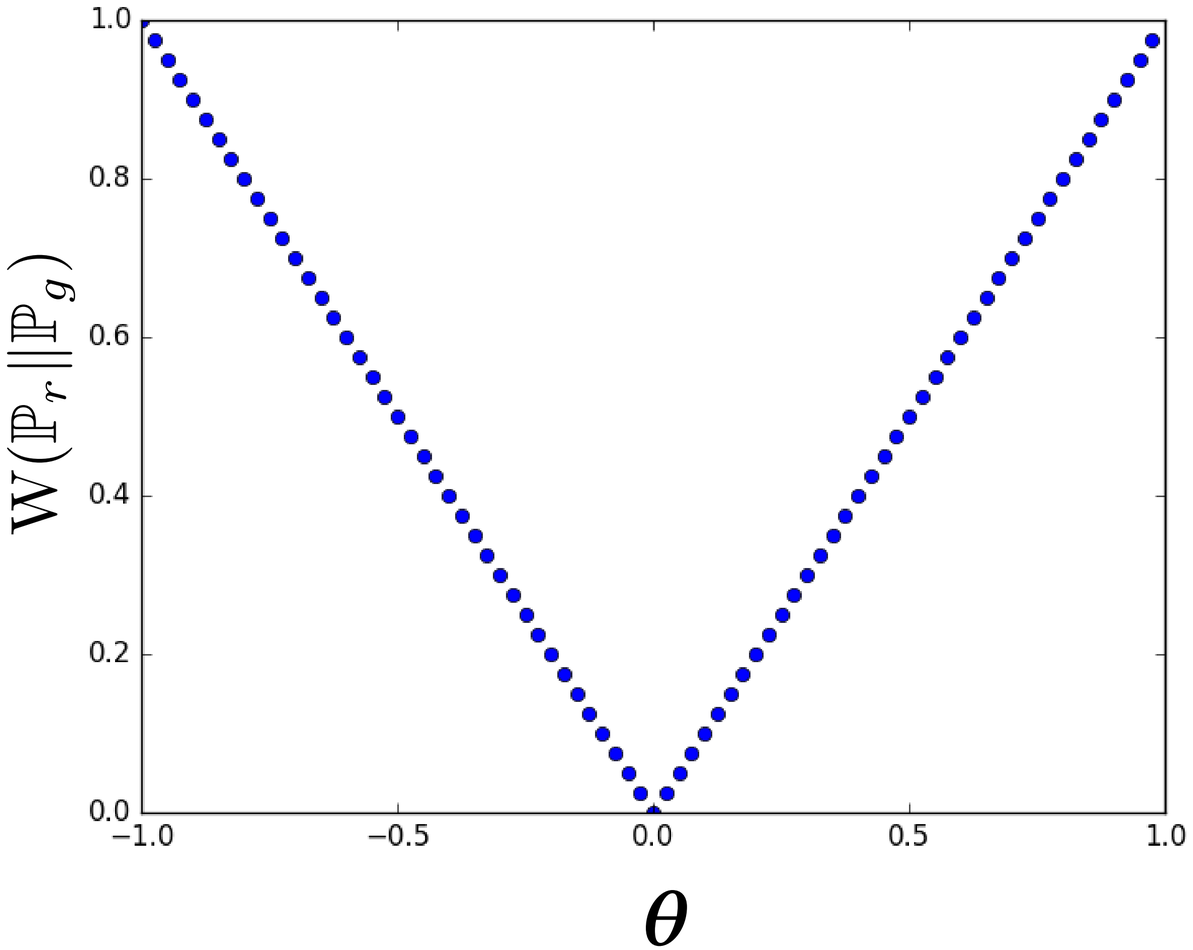
- What is the EM (or Wasserstein) distance in this simple case?

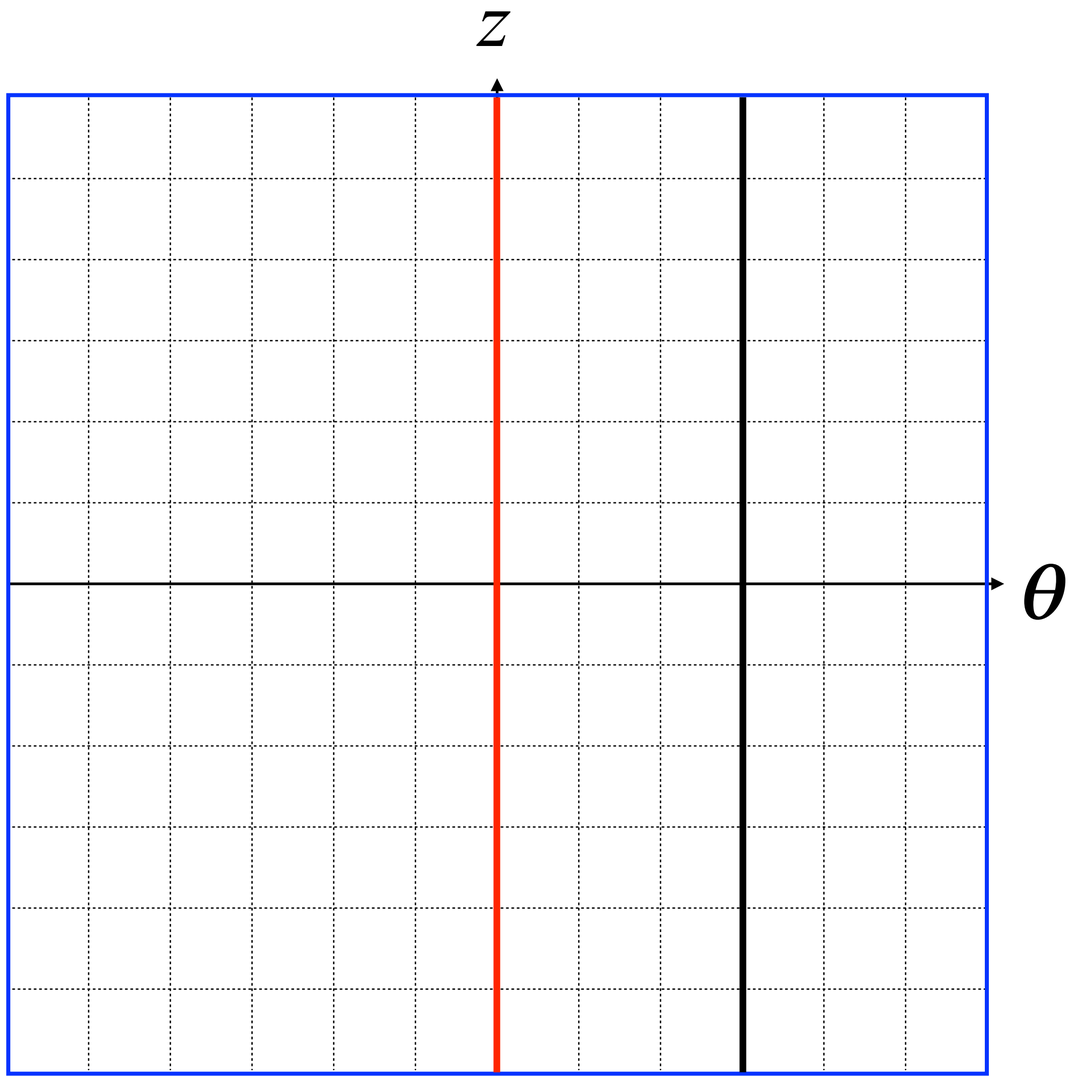


Example from Arjovsky et al. 2017

Wasserstein Distance

$$\mathrm{W}(\mathbb{P}_r \| \mathbb{P}_g) = |\theta|$$

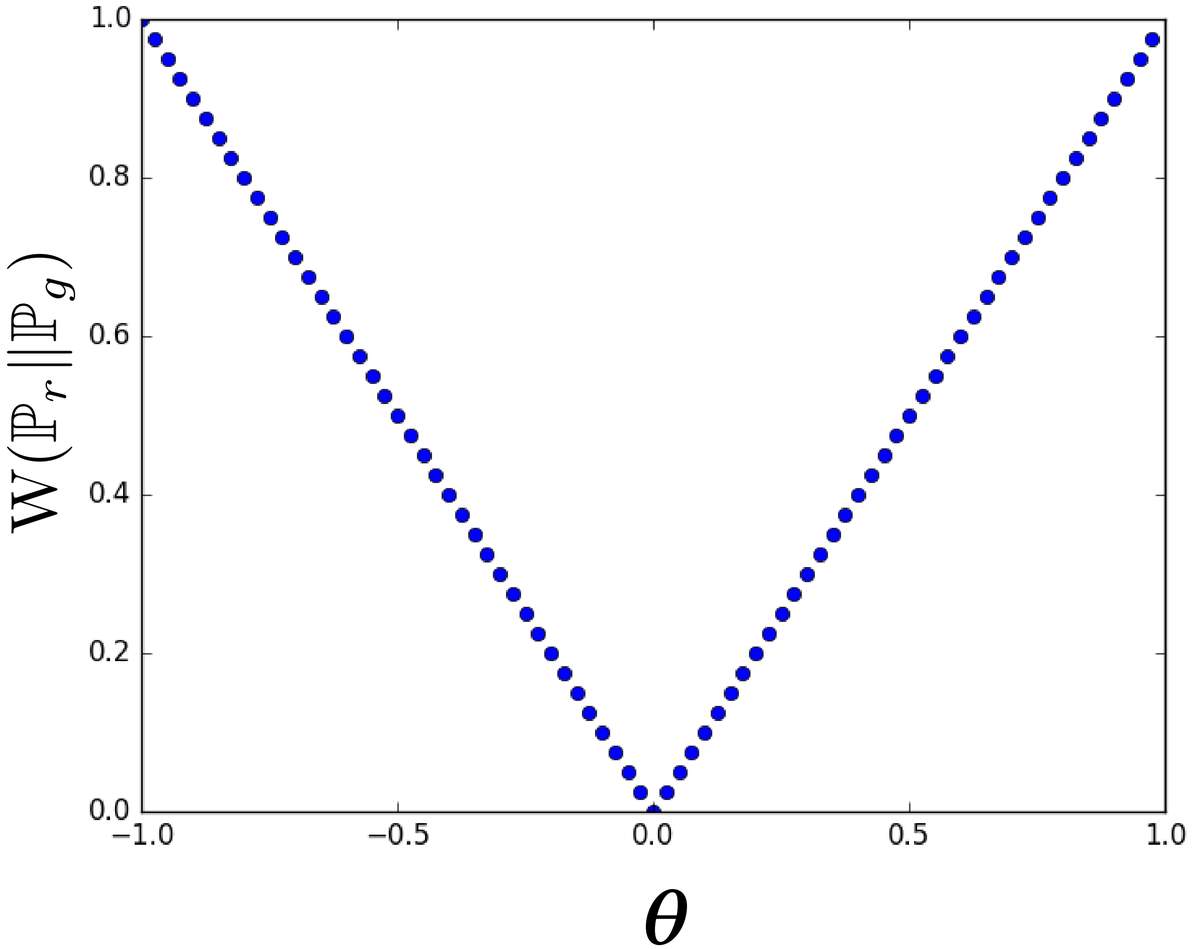


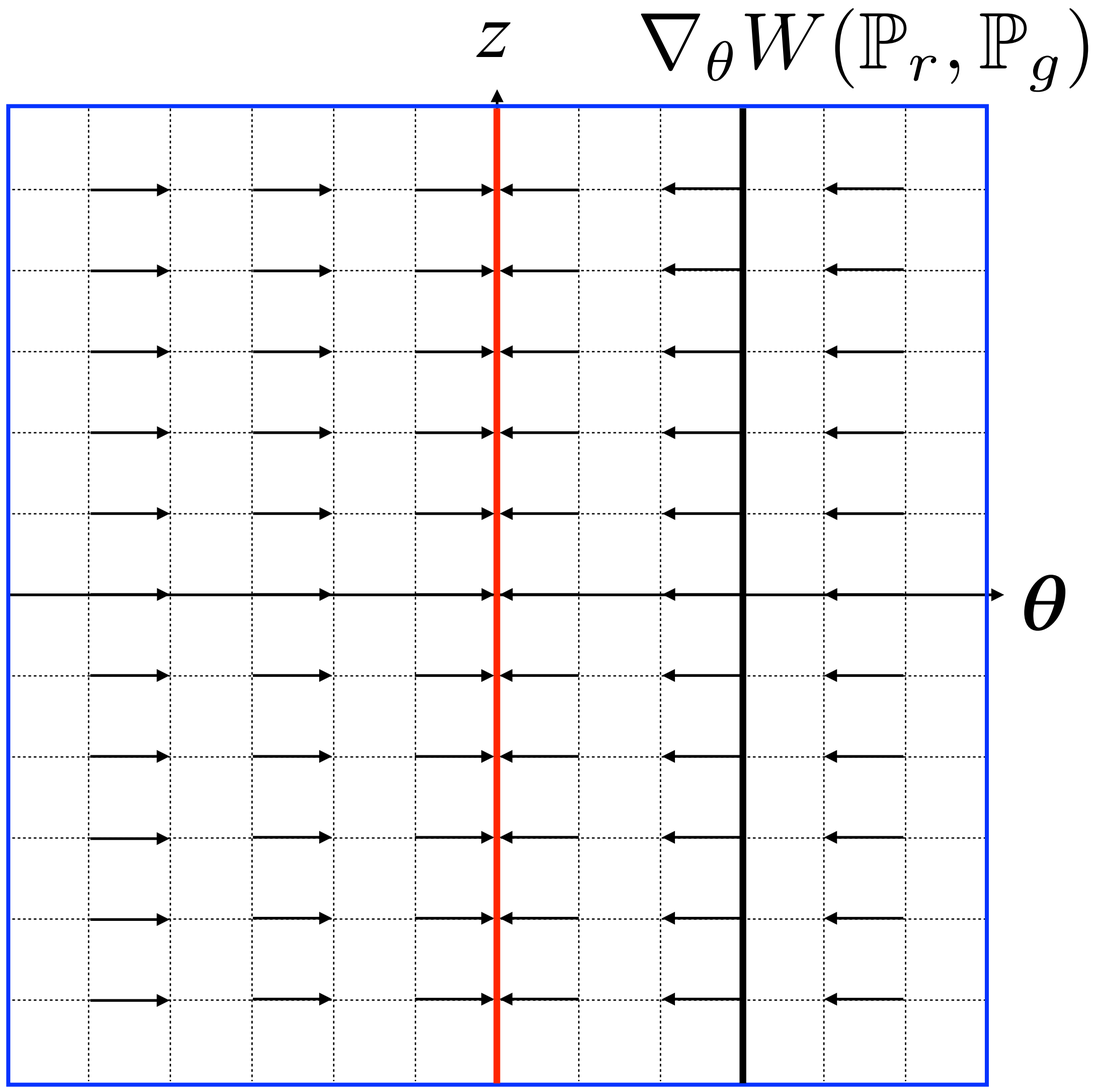


Example from Arjovsky et al. 2017

Wasserstein Distance

$$\mathrm{W}(\mathbb{P}_r \| \mathbb{P}_g) = |\theta|$$





Example from Arjovsky et al. 2017

Wasserstein GAN Arjovsky, Chintala, Bottou (2017)

- $W(\mathbb{P}_r, \mathbb{P}_g)$ might have nice properties compared to $\mathrm{JS}(\mathbb{P}_r, \mathbb{P}_g)$
- However, the infimum is intractable in:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma} \left[\|x - y\| \right]$$

- Can exploit Kantorovich-Rubinstein duality:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_g}[f(x)]$$

where the supremum is over all the 1-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$

Wasserstein GAN Arjovsky, Chintala, Bottou (2017)

- The WGAN Objective function:

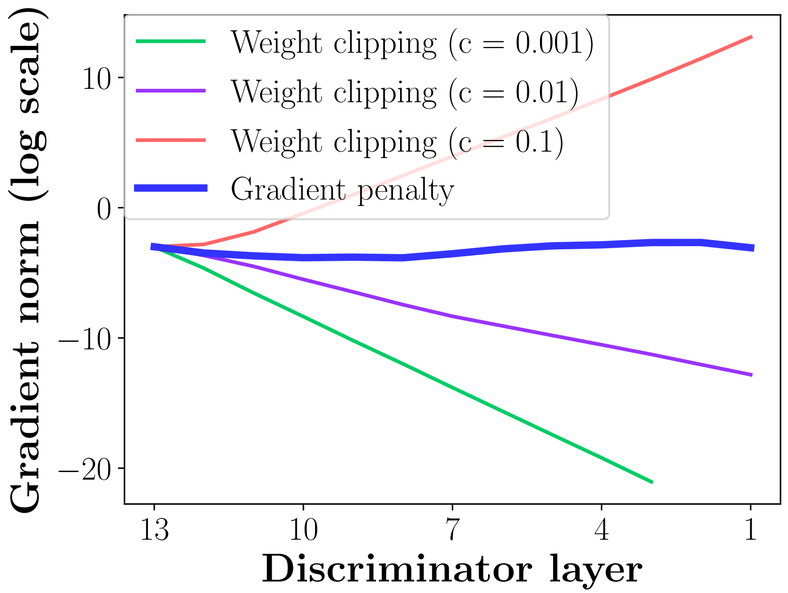
$$\min_{G} \max_{D \in \mathcal{D}} \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} \left[D(\boldsymbol{x}) \right] - \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} \left[D(\tilde{\boldsymbol{x}}) \right]$$

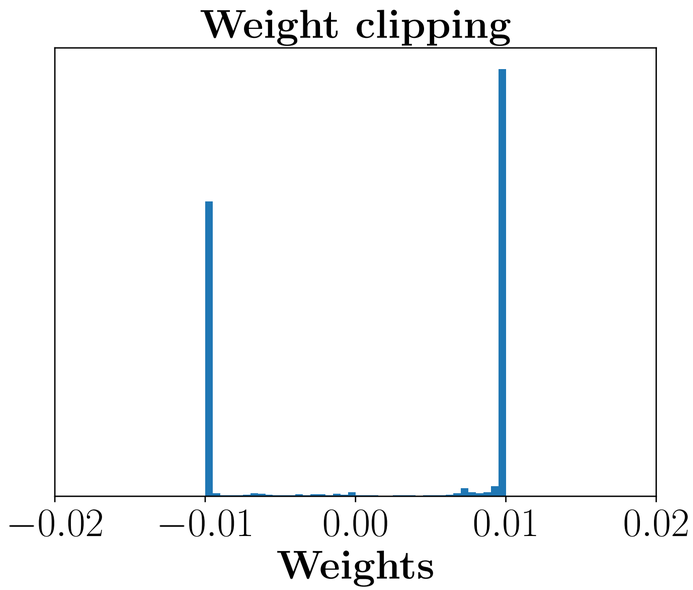
  where $\mathcal{D}$ is the set of 1-Lipschitz functions.

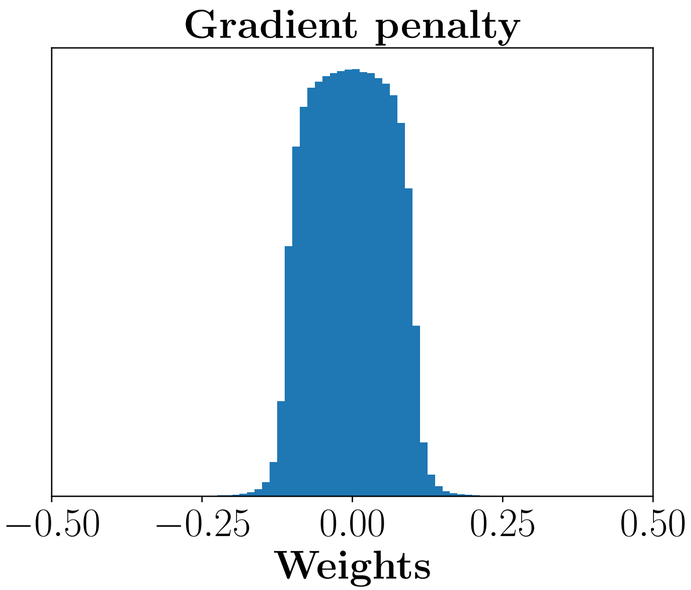
- Open question: how to effectively enforce the Lipschitz constraint on the critic D?
  - Arjovsky et al. (2017) propose to clip the weights of the critic to lie within a compact space [-c, c].
  - Results in a subset of the k-Lipschitz functions (k is a function of c).

Issues with Weight Clipping

1. Underuse capacity
2. Exploding and vanishing gradients







Gradient Penalty Approach

Gulrajani, Ahmed, Arjovsky, Dumoulin, Courville (2017)

- **A property of the optimal WGAN critic**: If $\tilde{\boldsymbol{x}} \sim \mathbb{P}_g$ then there is a point $\boldsymbol{x} \sim \mathbb{P}_r$, such that for all points $\boldsymbol{x}_t = t\boldsymbol{x} + (1-t)\tilde{\boldsymbol{x}}$ (on a straight line between $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$) then:

$$\nabla D^*(\boldsymbol{x}_t) = \frac{\boldsymbol{x} - \boldsymbol{x}_t}{\|\boldsymbol{x} - \boldsymbol{x}_t\|}$$

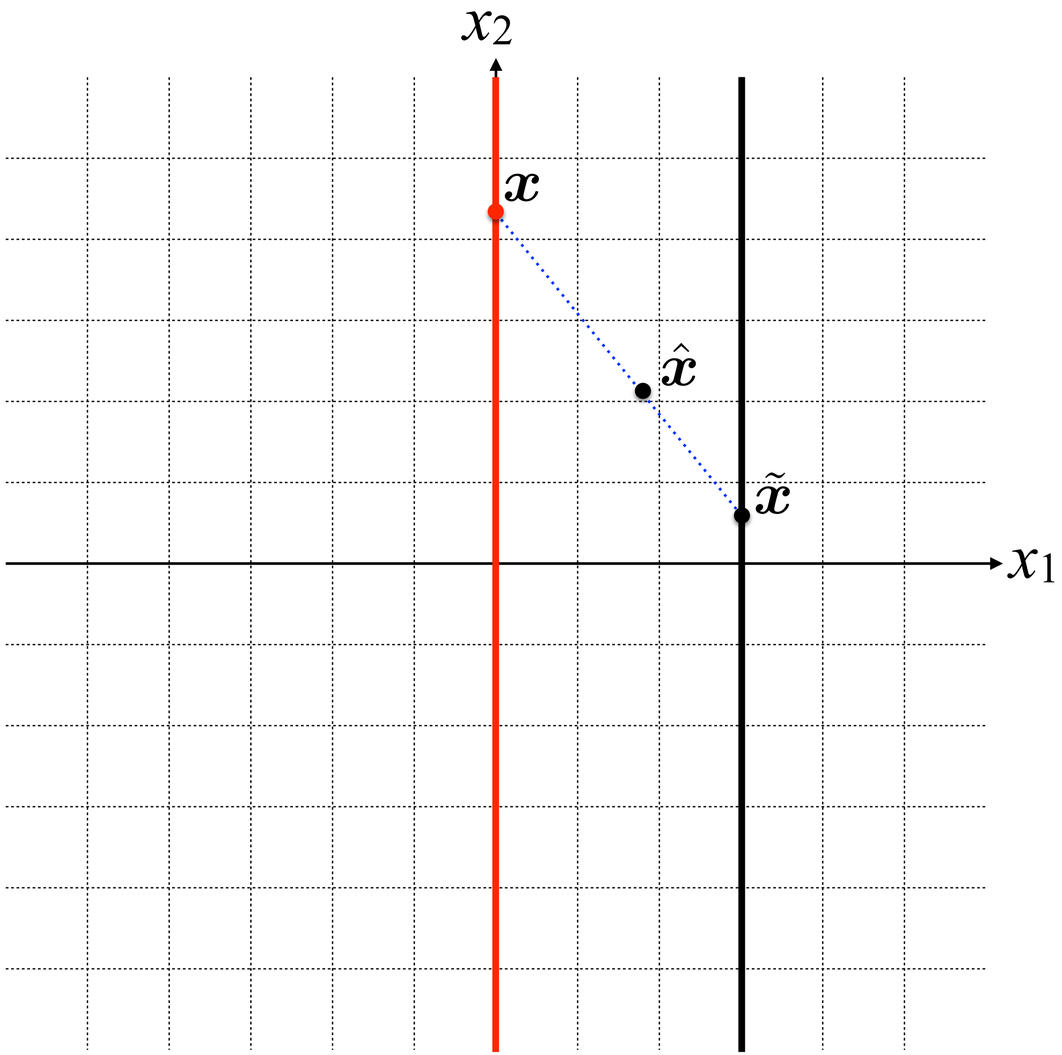
- This implies the optimal WGAN critic has gradient norm 1 at $\boldsymbol{x}_t$

- Gradient Penalty version of WGAN (i.e. the WGAN-GP) objective:

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g}[D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r}[D(\boldsymbol{x})]}_{\text{Original critic loss}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}}\left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2\right]}_{\text{Our gradient penalty}}$$

Gradient Penalty Approach

Gulrajani, Ahmed, Arjovsky, Dumoulin, Courville (2017)



Gradient penalty:

$$\mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}} \left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2 \right]$$

Sample along straight lines:

$$\epsilon \sim U[0,1], \boldsymbol{x} \sim \mathbb{P}_r, \tilde{\boldsymbol{x}} \sim \mathbb{P}_g$$

$$\hat{\boldsymbol{x}} = \epsilon \boldsymbol{x} + (1-\epsilon)\tilde{\boldsymbol{x}}$$

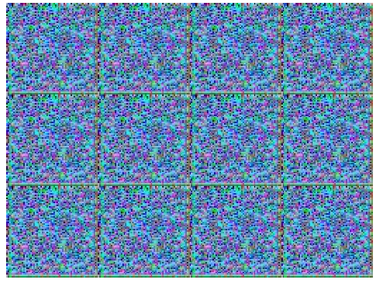
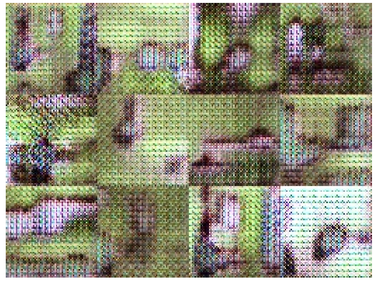
Comparison on difficult to train architectures

- Comparison based on recommended default parameter setting for each algorithm.
- WGAN-GP is more robust to variations in training setups.

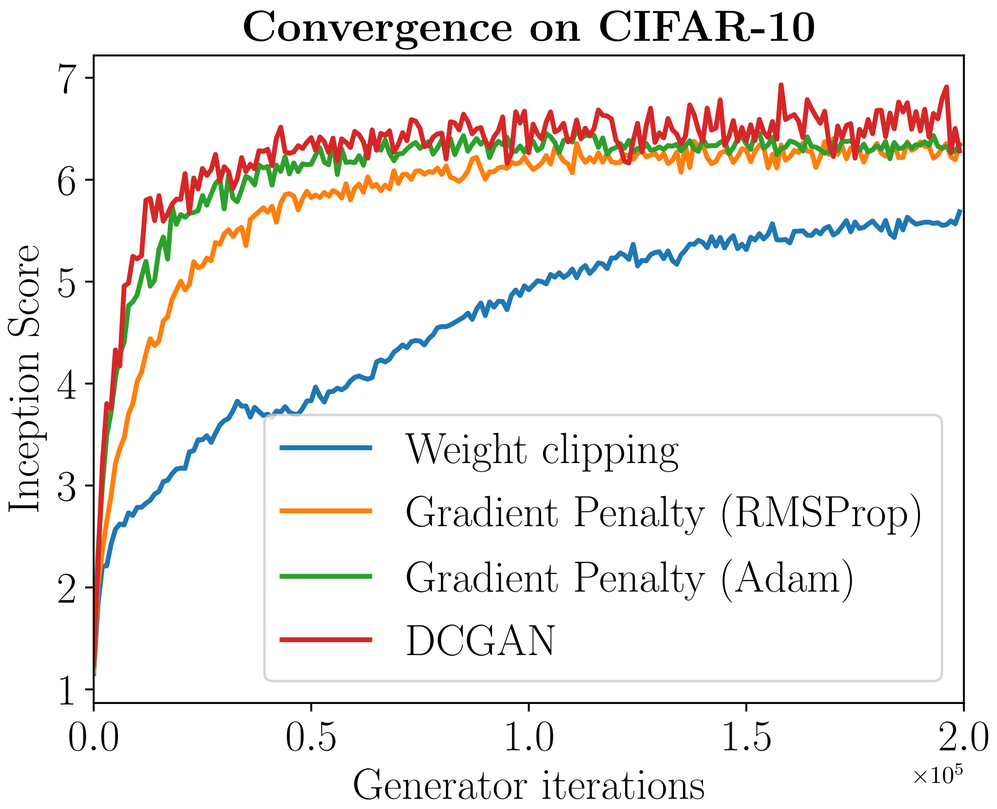
DCGAN	LSGAN	WGAN (clipping)	WGAN-GP (ours)
Baseline (G: DCGAN, D: DCGAN)			
G: No BN and a constant number of filters, D: DCGAN			
G: 4-layer 512-dim ReLU MLP, D: DCGAN			

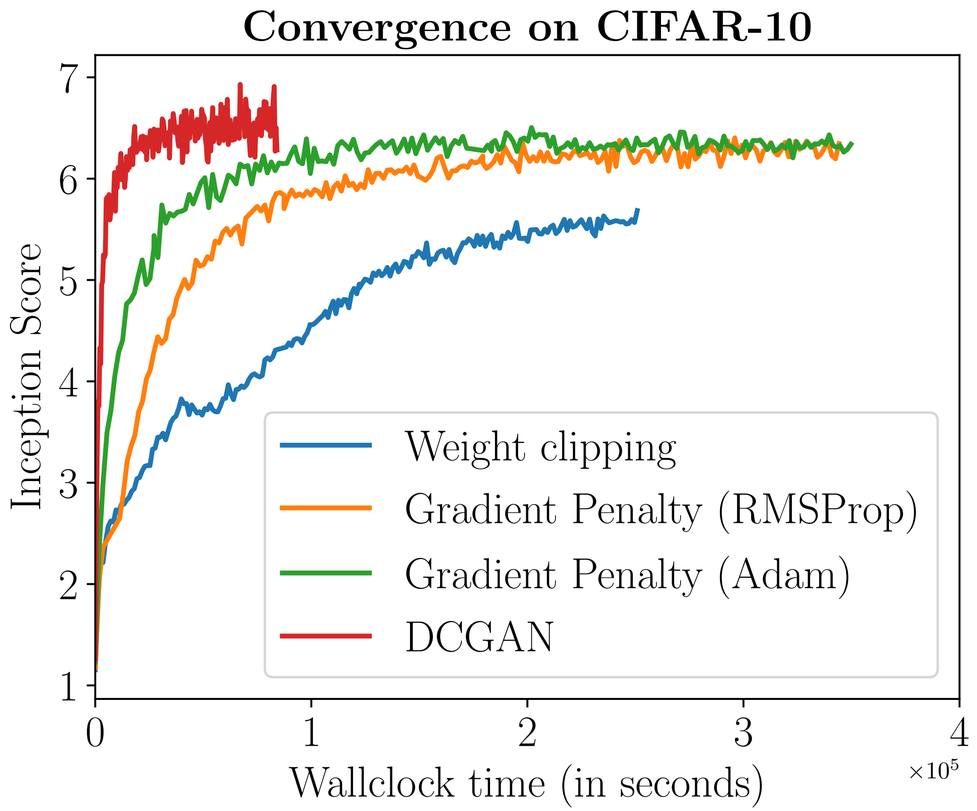
Comparison on difficult to train architectures

- Comparison based on recommended default parameter setting for each algorithm.
- WGAN-GP is more robust to variations in training setups.

DCGAN	LSGAN	WGAN (clipping)	WGAN-GP (ours)
Gated multiplicative nonlinearities everywhere in G and D			
			
$tanh$ nonlinearities everywhere in G and D			
			
101-layer ResNet G and D			
			

WGAN with Gradient Penalty





But what about inference…

- How can we use generative models?
    - GANs can generate content, but somethings you want to make inference about observed data.
- Can we incorporate an inference mechanism into GANs?
- Can we learn an inference mechanisms using an adversarial training paradigm?

Two papers, one model

- **ALI**: Vincent Dumoulin, Ishmael Belghazi, Olivier Mastropietro Ben Poole, Alex Lamb, Martin Arjovsky (2016) *ADVERSARIALLY LEARNED INFERENCE*, arXiv:1606.00704

- **BiGAN**: Donahue, Krähenbühl and Darrell (2016), *ADVERSARIAL FEATURE LEARNING*, arXiv:1605.09782

- But also showing results on Hierarchical ALI by Ishmael Belghazi, Sai Rajeshwar, Olivier Mastropietro and Negar Rostamzadeh

Adversarially learned inference: Main idea

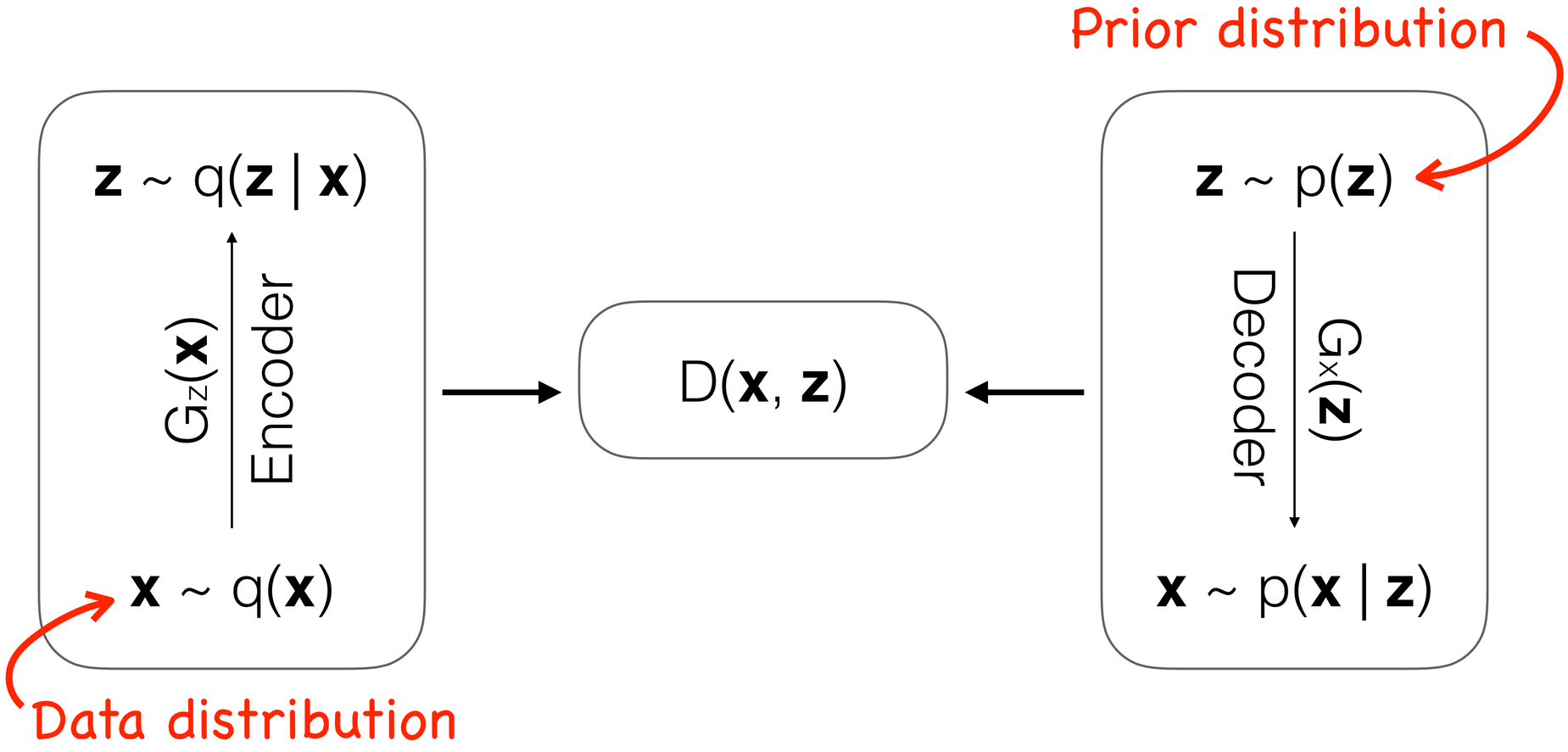
- Cast the learning of both an inference model (*encoder*) and a generative model (*decoder*) in a GAN-like adversarial framework.
- Discriminator is trained to discriminate between *joint* samples ($\mathbf{x}$, $\mathbf{z}$) from:
  - E*ncoder distribution* $q(\mathbf{x}, \mathbf{z}) = q(\mathbf{x})\, q(\mathbf{z} \mid \mathbf{x})$, or
  - D*ecoder distribution* $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})\, p(\mathbf{x} \mid \mathbf{z})$.

Data distribution

Prior distribution

- Generator learns conditionals $q(\mathbf{z} \mid \mathbf{x})$ and $p(\mathbf{x} \mid \mathbf{z})$ to fool the discriminator.

ALI: model diagram



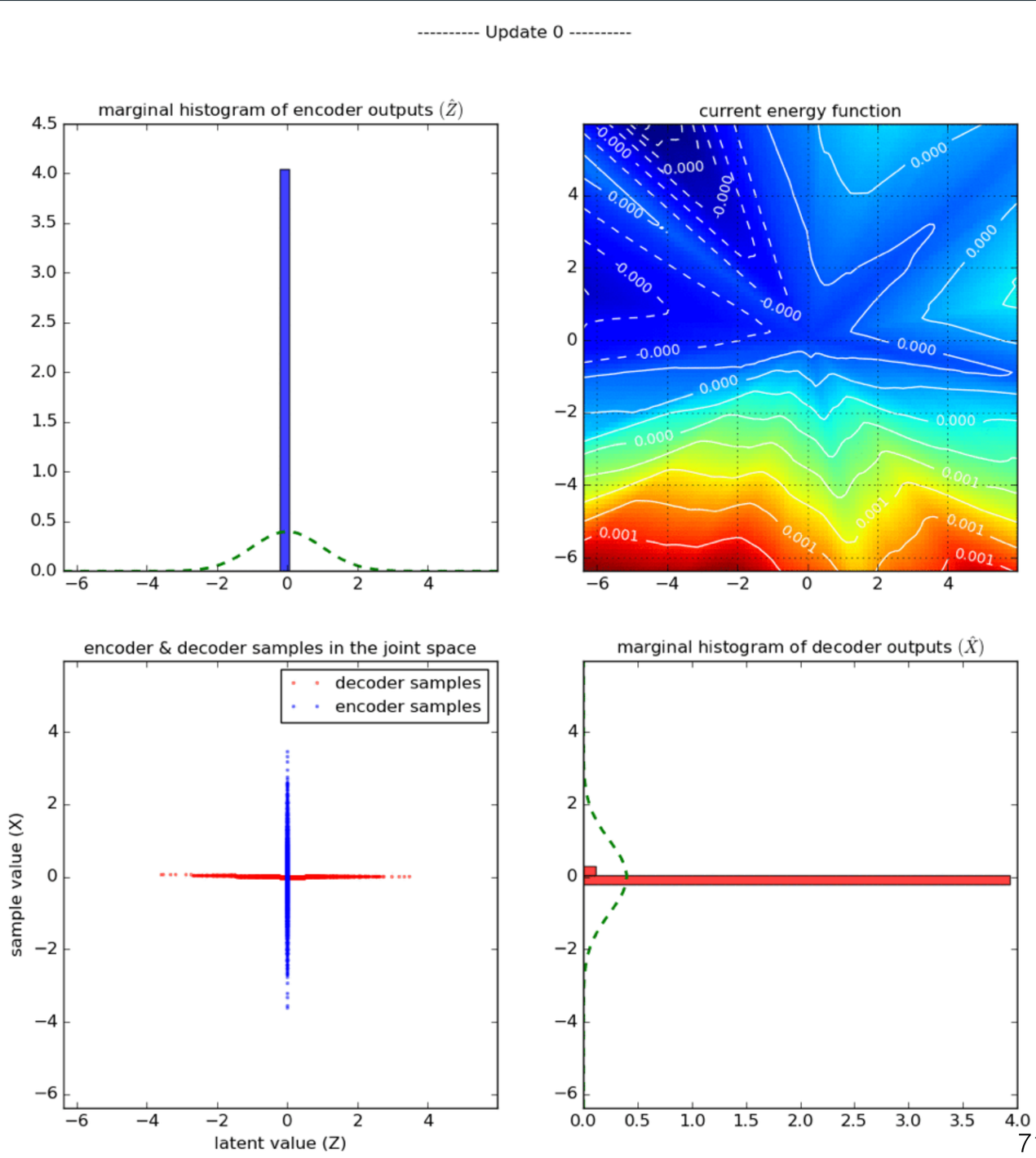
Toy Example

- Learning the Identity function:

  Encoder: $X \sim N(0,1)$

  Decoder: $Z \sim N(0,1)$

Zihang Dai



Theoretical properties

In analogy with GAN, under an ideal discriminator, the generator minimizes the Jensen-Shannon divergence between p(**x**, **z**) and q(**x**, **z**).

SVHN

Samples

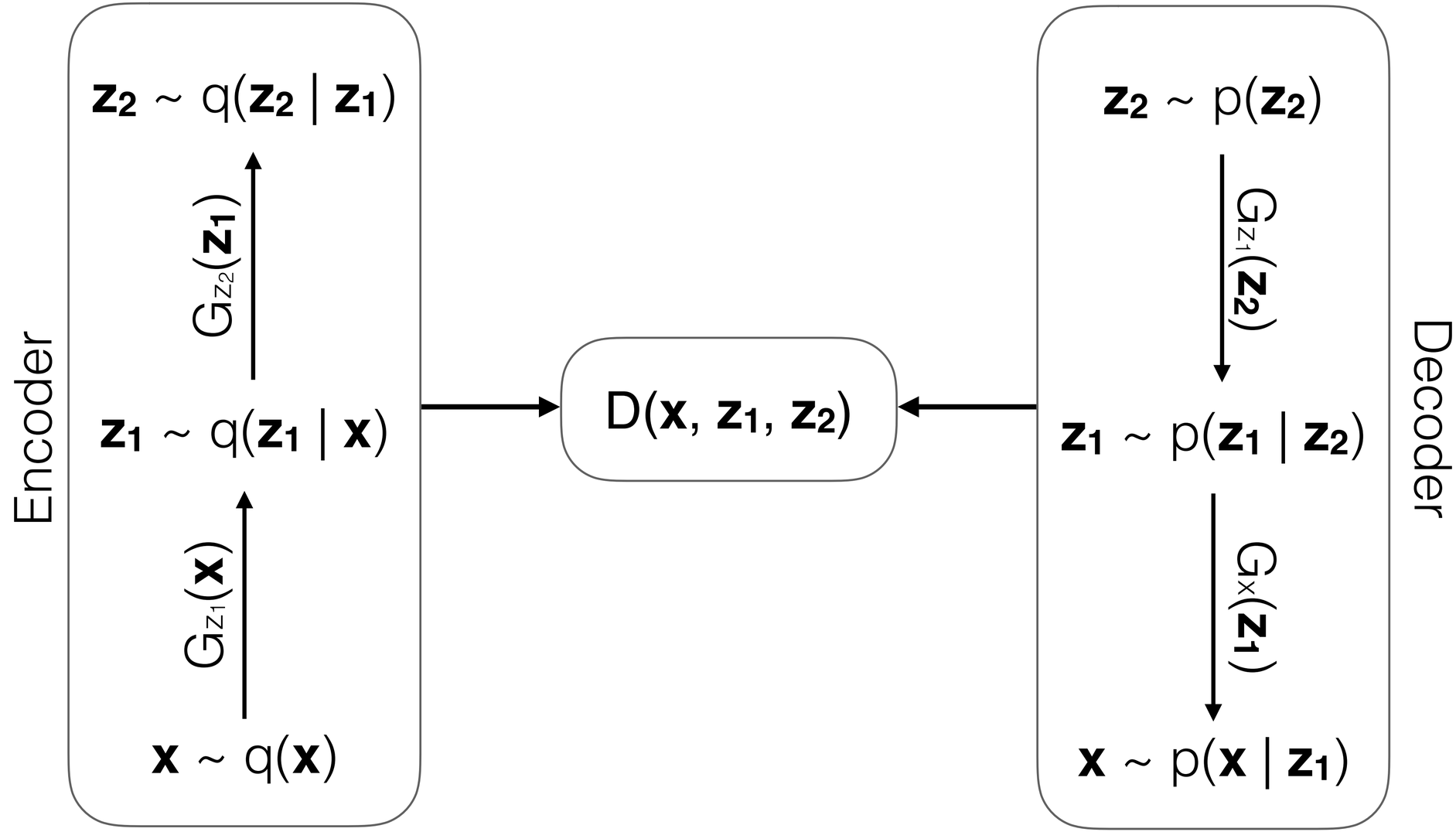
Reconstructions

CelebA face dataset

Samples

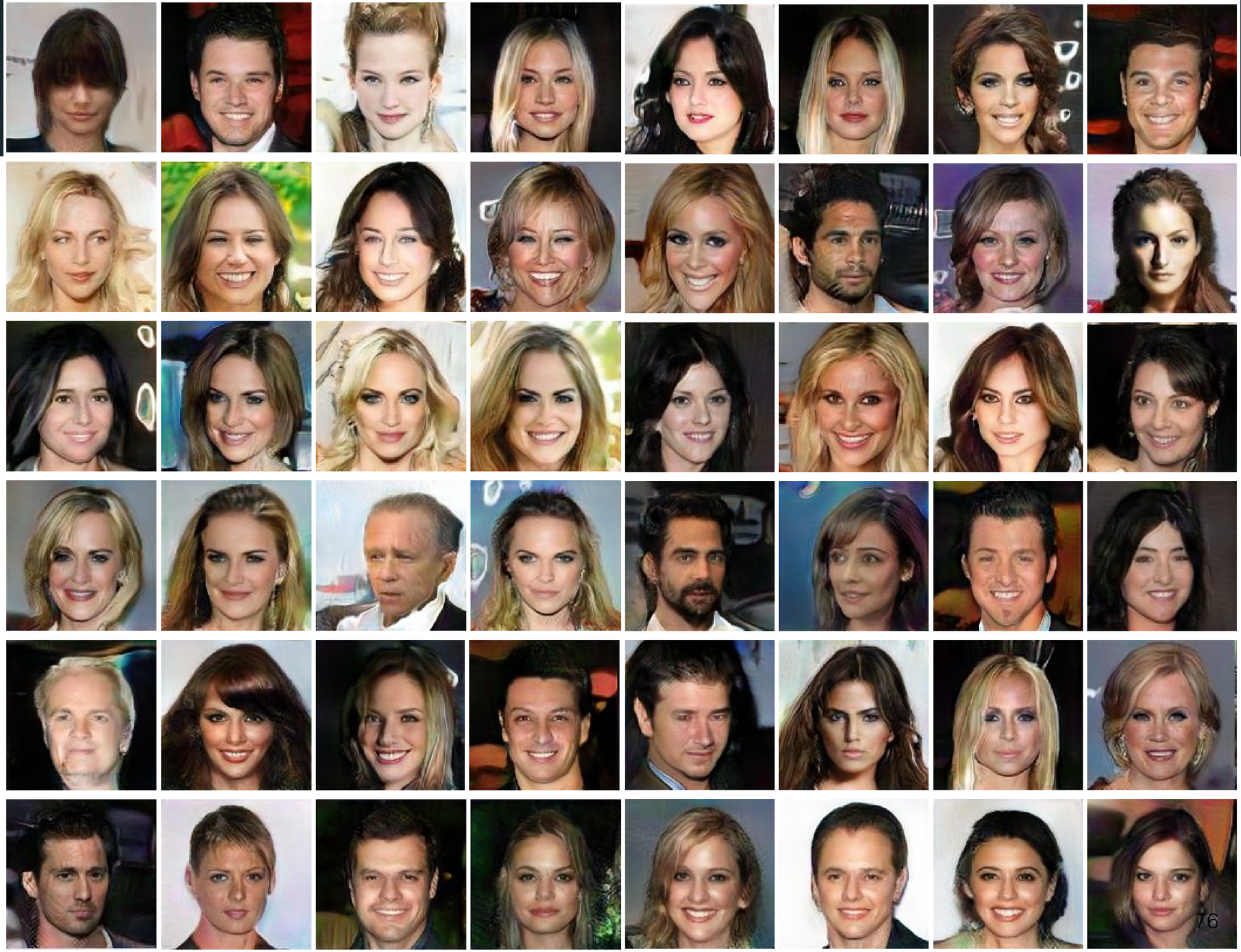
Reconstructions

Hierarchical ALI: model diagram



Hierarchical ALI

CelebA-128X128



Model samples

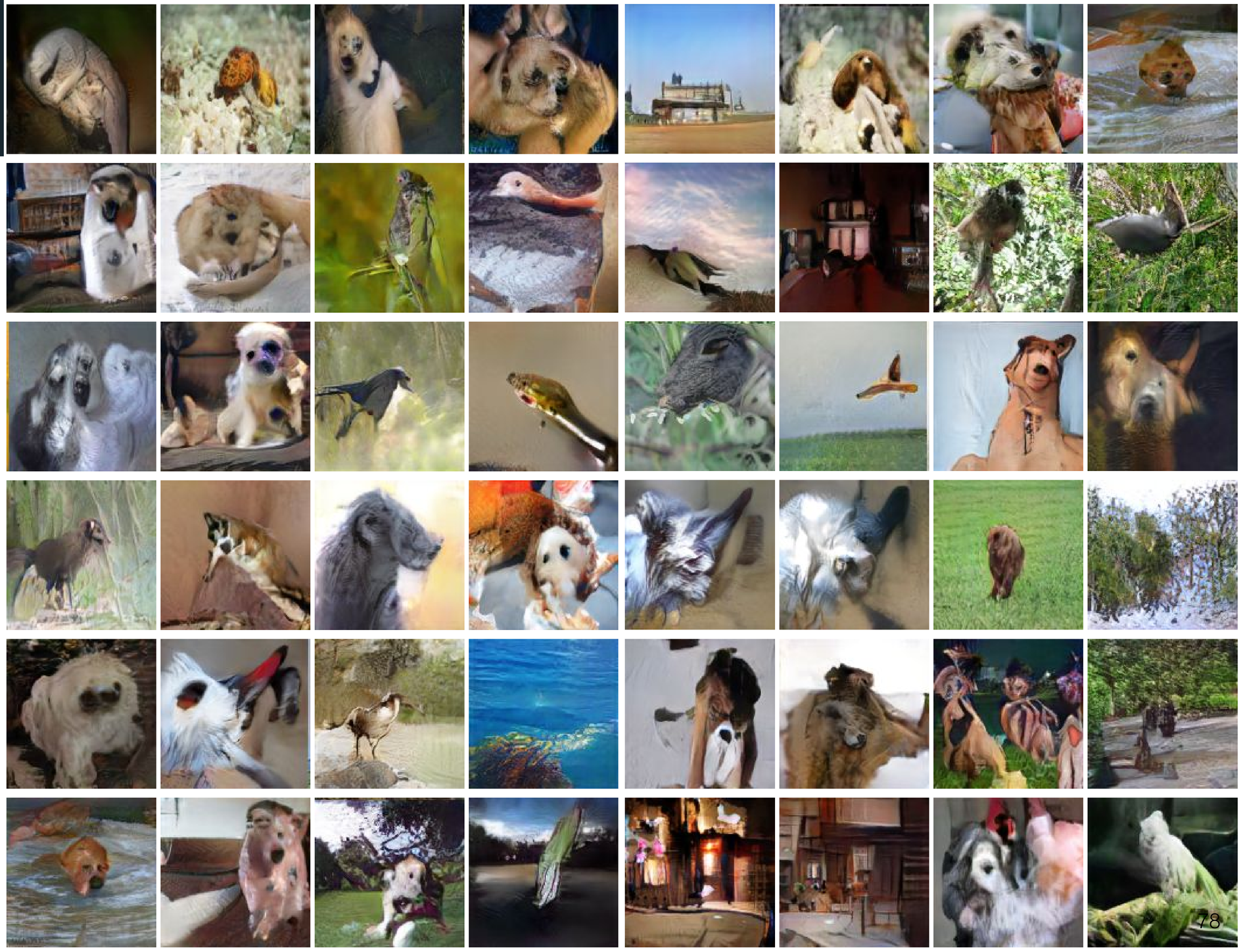
Hierarchical ALI: CelebA-128x128

Reconstructions given z_1, z_2

Reconstructions given z_2

Hierarchical ALI

Unconditional ImageNet-128x128



Model samples

Hierarchical ALI: ImageNet-128X128

Generative models II: Outline

- Autoregressive models
  - PixelCNN
- Latent variable models
  - Variational Autoencoders
    - VAEs
    - Inverse Autoregressive Flow: An improved strategy for inference
  - Generative Adversarial Networks
    - GAN
    - Wasserstein GAN
    - ALI