Lecture 10: Deep Generative Models

Last Week's Class Review Points

Negative Transfer

Blackbox vs Optimization based Meta-Learning

Negative Transfer

Challenge #1: Negative transfer

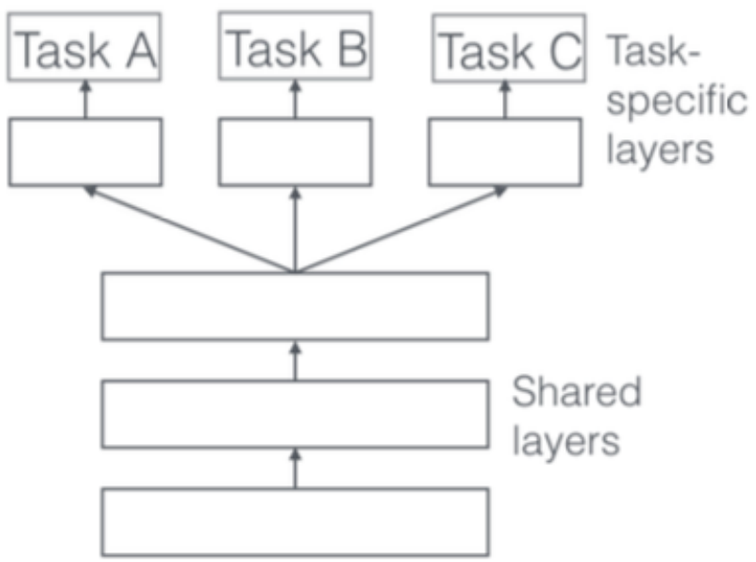
Negative transfer: Sometimes independent networks work the best.

Multi-Task CIFAR-100

recent approaches

	% accuracy	
task specific, 1-fc (Rosenbaum et al., 2018)	42	} multi-head architectures
task specific, all-fc (Rosenbaum et al., 2018)	49	
cross stitch, all-fc (Misra et al., 2016b)	53	} cross-stitch architecture
independent	67.7	} independent training

(Yu et al. Gradient Surgery for Multi-Task Learning. 2020)



Negative Transfer

- Common Causes
  - Small model capacity — not enough capacity to learn both tasks
    - Note: too large can avoid positive transfer
  - Very different tasks - model of Task1 forced to use useless features with respect to Task2
  - Optimization — some tasks learn at different rates

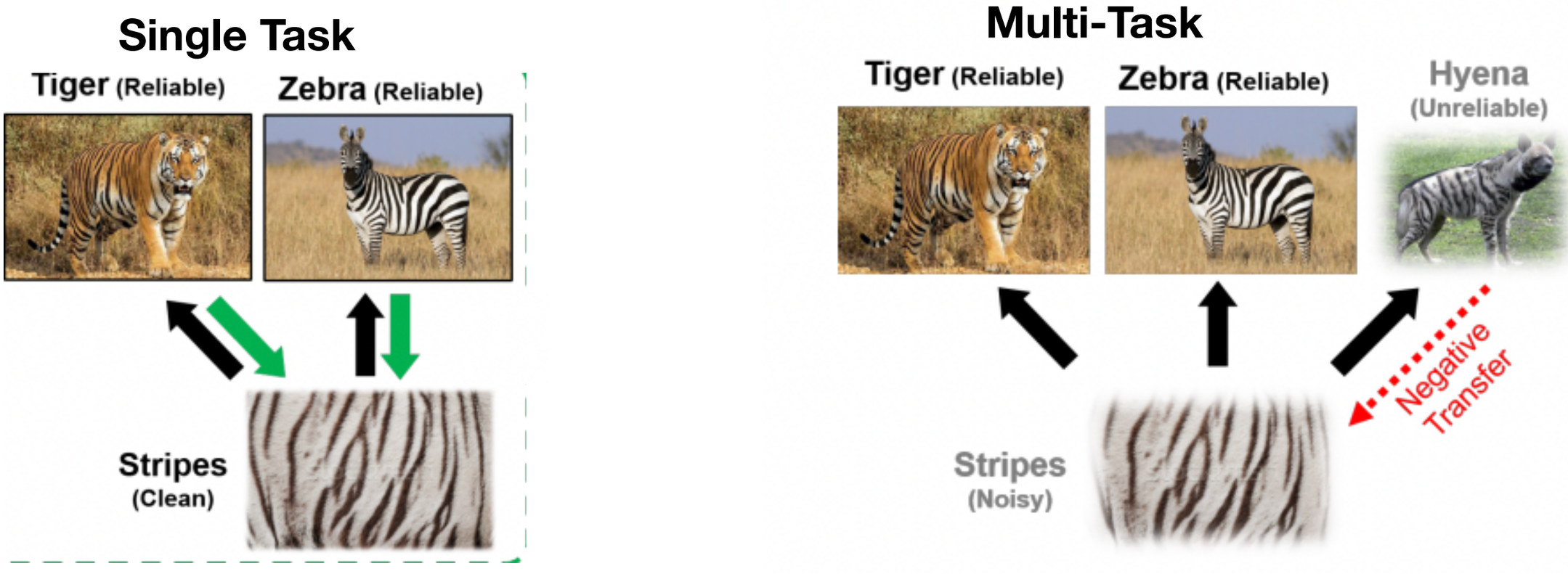


Image Credit: Deep Asymmetric Multi-Task Feature Learning

Generative Models

Generative Models

- Generative models are models that can generate the data
- Learning generative models is a subset of unsupervised learning
- There’s various things in machine/deep learning context we might want
  - Sampling
  - Density estimation
  - Unsupervised Representation learning
- We focus on faitful sampling and representation learning

The Promises of Unsupervised Learning

- The promise of unsupervised learning
  - Learn representations of data from unlabeled data
  - Use a bit of supervision to solve many tasks
- Generative models have for a longtime been considered a potential path to useful unsupervised representation learning

- If we can generate the data we have understood it's underlying behaviour

Sampling has lots of applications

- Sampling from generative models has lots of applications in its own right

- Image Synthesis is seful in computer graphics applications: e.g. movies

- Generating text/ text completion: e.g. chatbots

Preliminaries: AutoEncoders

Encoder Network

g_ϕ

z

Decoder Network

f_θ

$$Loss \rightarrow \min_{\theta,\phi} \|f_\theta \circ g_\phi(X) - X\|$$

Examples of Generative Models

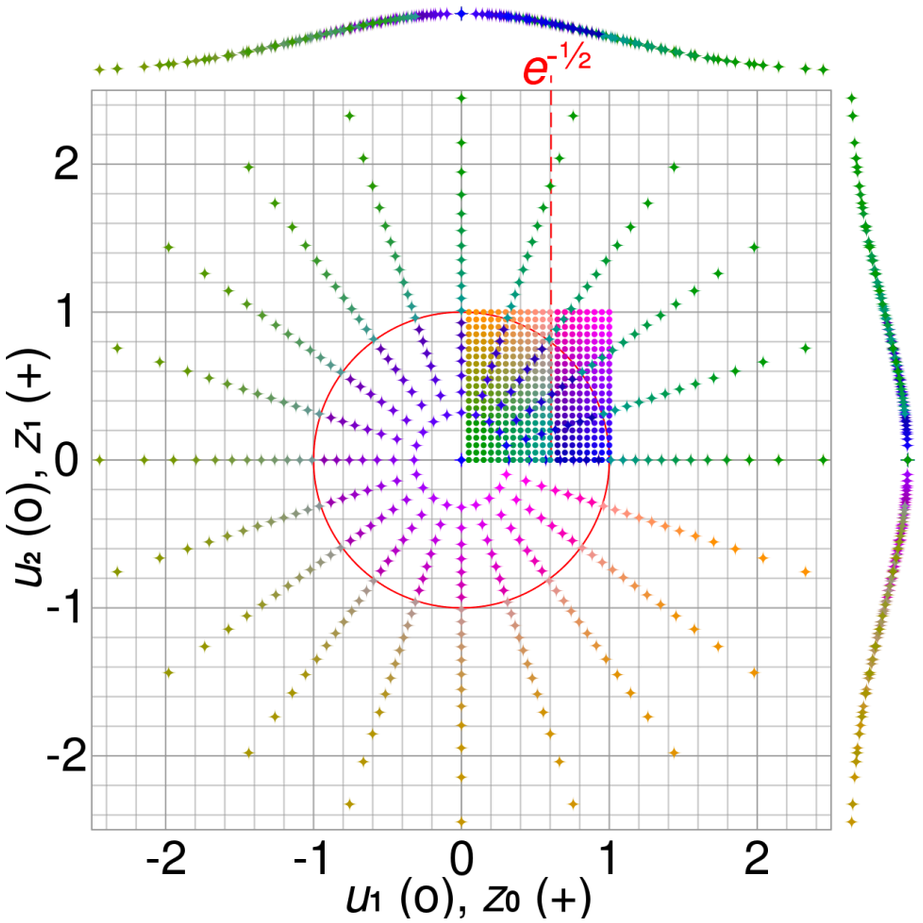
$$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- Gaussian distribution
- Given dataset X = x1,x2,x3… how do we fit a gaussian?
- How do we sample from this?

Box-Mueller

$$U_1, U_2 \sim Uniform(0,1)$$

We can sample U from say wall clock time



$$R = \sqrt{-2lnU_1}$$

$$\Theta = 2\pi U_2$$

$$X_1 = Rcos\Theta \qquad = \sqrt{-2lnU_1}cos(2\pi U_2)$$

$$X_2 = Rsin\Theta \qquad = \sqrt{-2lnU_1}sin(2\pi U_2)$$

Sampling from a unimodal gaussian distributions is tricky, what do we do for more complex ones?

Latent Variable Models

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

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

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

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

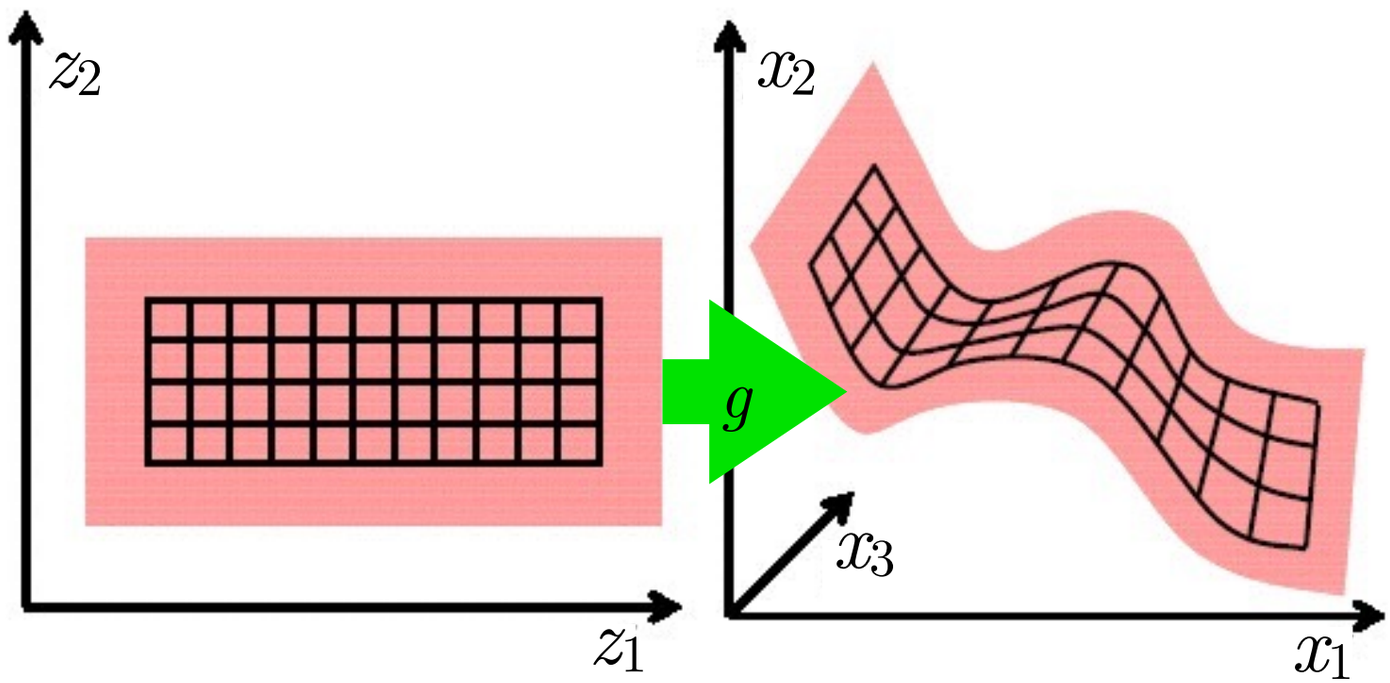
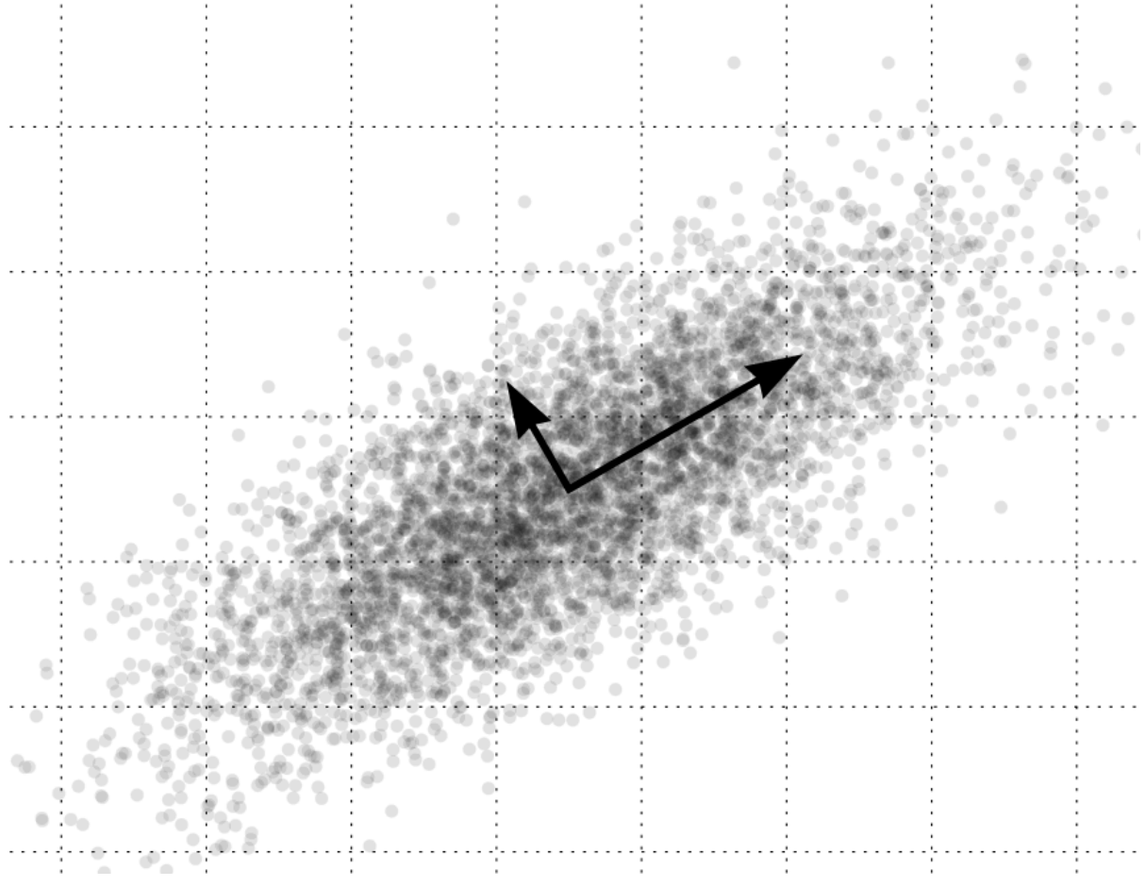


Image from: Ward, A. D., Hamarneh, G.: **3D Surface Parameterization Using Manifold Learning for Medial Shape Representation**, *Conference on Image Processing, Proc. of SPIE Medical Imaging*, 2007

Slide Credit: Aaron Courville

Latent Variable Generative Models

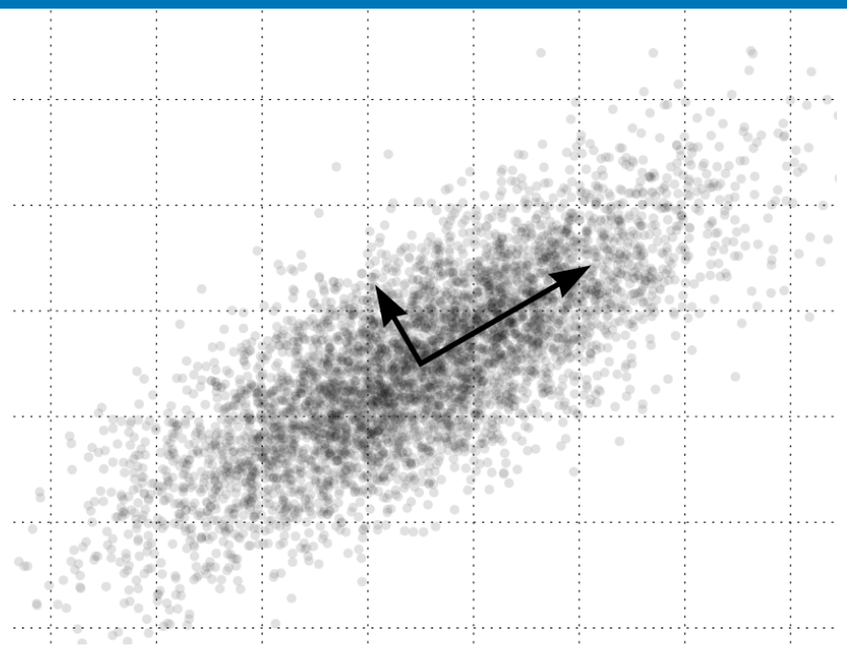
- Latent variable generative models arise in many areas of machine learning and statistics
  - Hidden Markov Models
  - PCA



Warmup: Probabilistic PCA

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

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



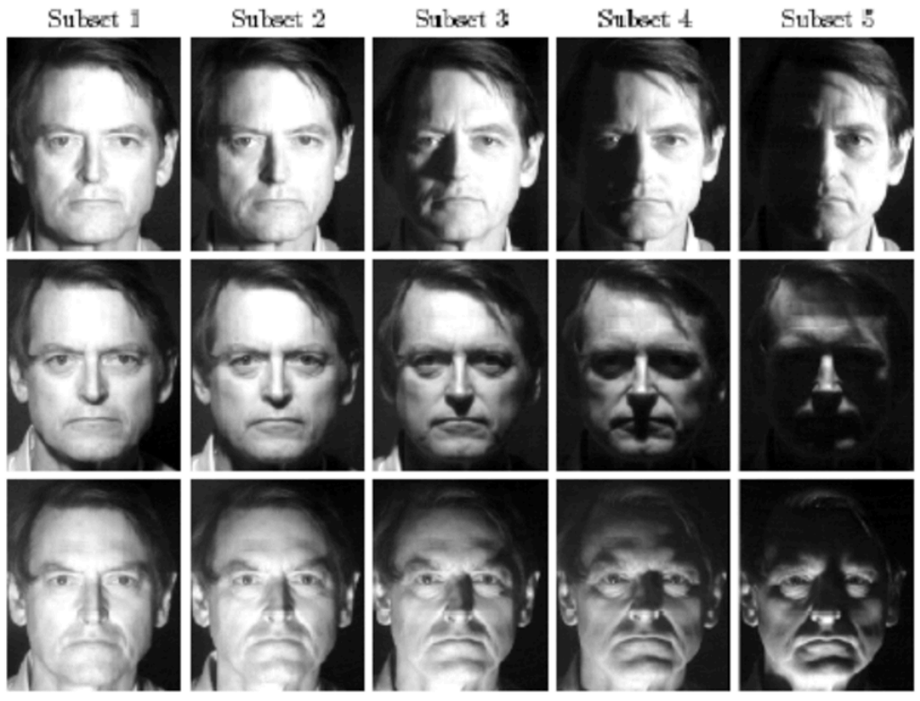
Top K principal components $\mathbf{W} \in \mathbb{R}^{D \times K}$

$$p(\mathbf{z}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$p(\mathbf{x} \mid \mathbf{z}) \sim \mathcal{N}(\mathbf{W}\mathbf{z}, \sigma^2\mathbf{I})$$ **Let's assume 0 mean**

$$p(\mathbf{x}) \sim \mathcal{N}(\mathbf{0}, \mathbf{W}\mathbf{W}^\mathbf{T} + \sigma^2\mathbf{I})$$ **Regular PCA** $\sigma^2 \to 0$

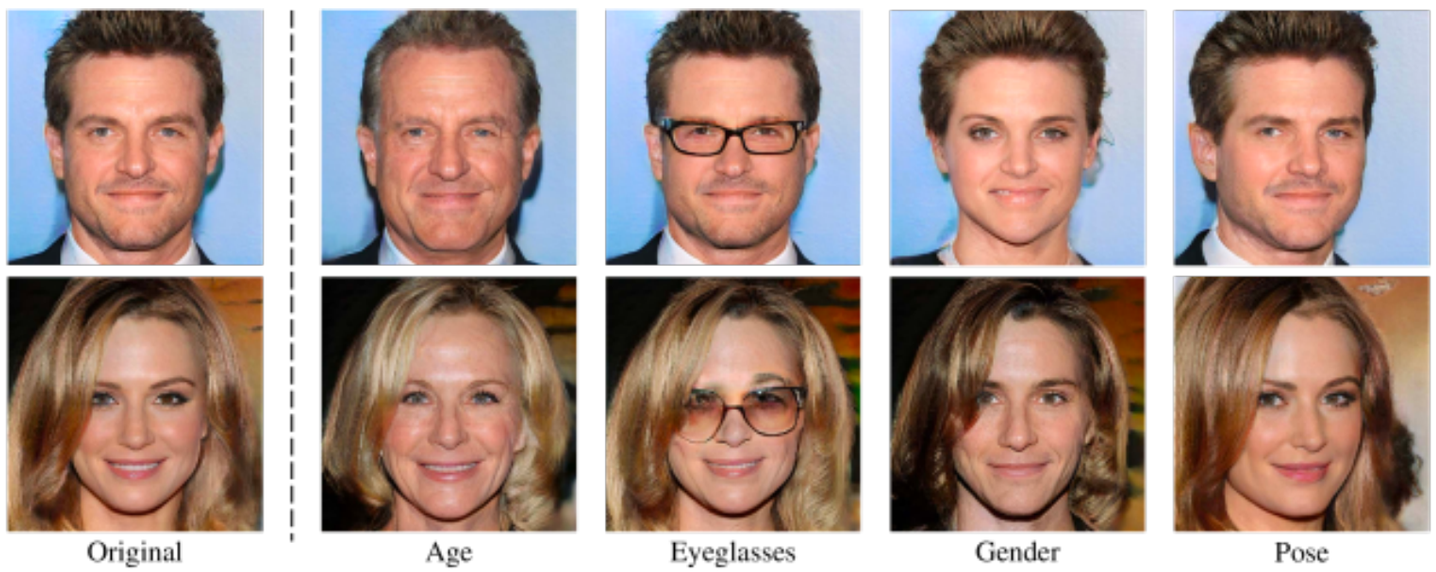
Exploring PCA latent space



PCA can capture some simple variability (e.g. lighting) for well aligned faces

$$p(\mathbf{x}\,|\,\mathbf{z}) \sim \mathcal{N}(\mathbf{W}\mathbf{z}, \sigma^2\mathbf{I})$$

however ultimately a linear model



We want to find latents which represent more high level (non-linear) factors of variation

~~$p(\mathbf{x}\,|\,\mathbf{z}) \sim \mathcal{N}(\mathbf{W}\mathbf{z}, \sigma^2\mathbf{I})$~~

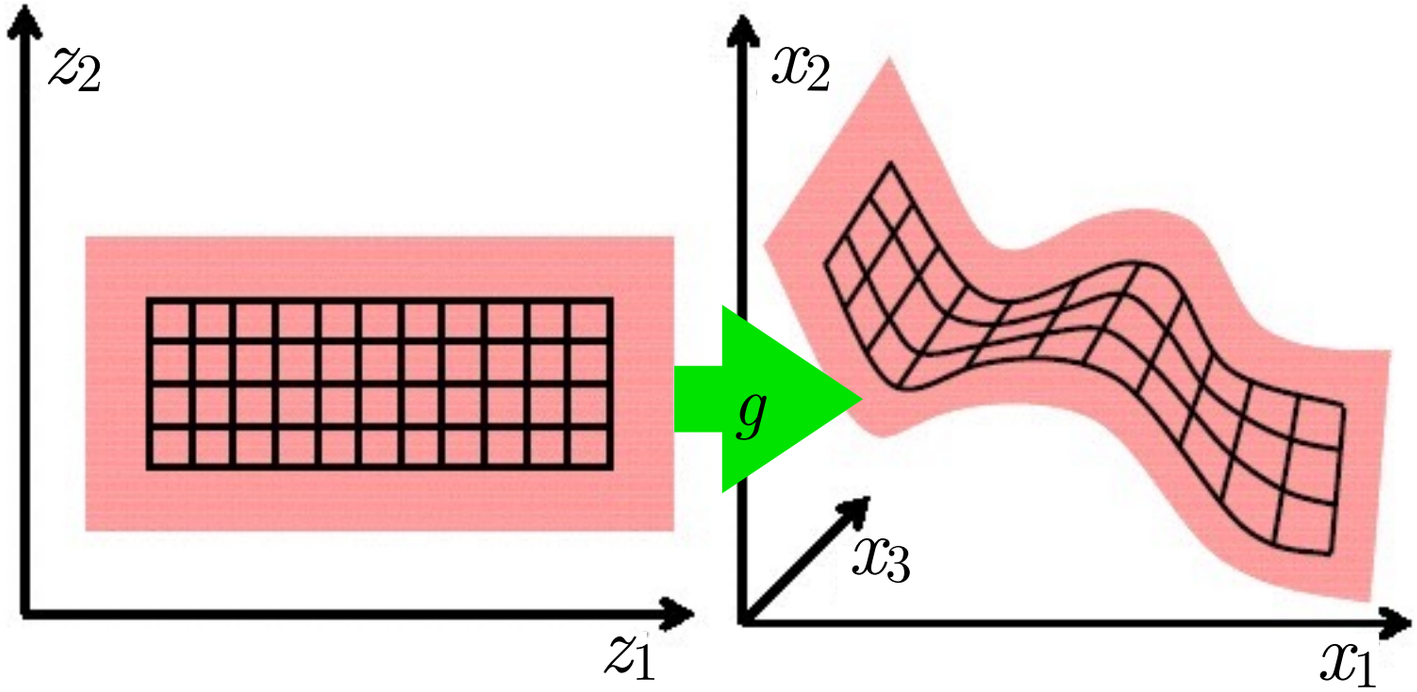
$$p(\mathbf{x}\,|\,\mathbf{z}) \sim \mathbf{NeuralNet}(\mathbf{z})$$

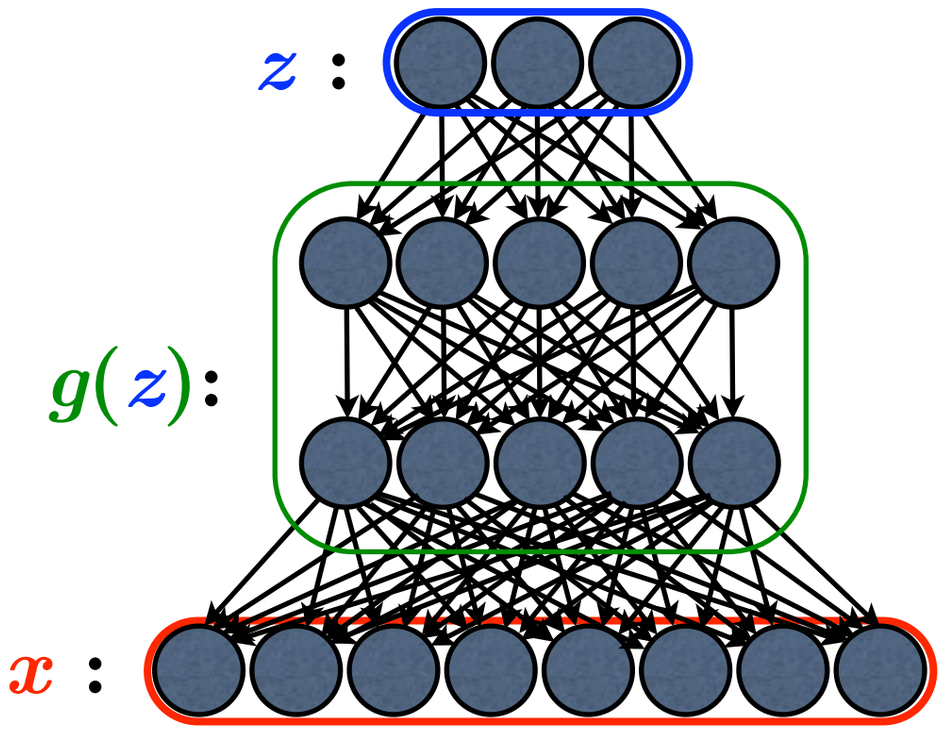
Variational Auto-Encoder

- Leverage ***neural networks*** to learn a latent variable model.

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

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

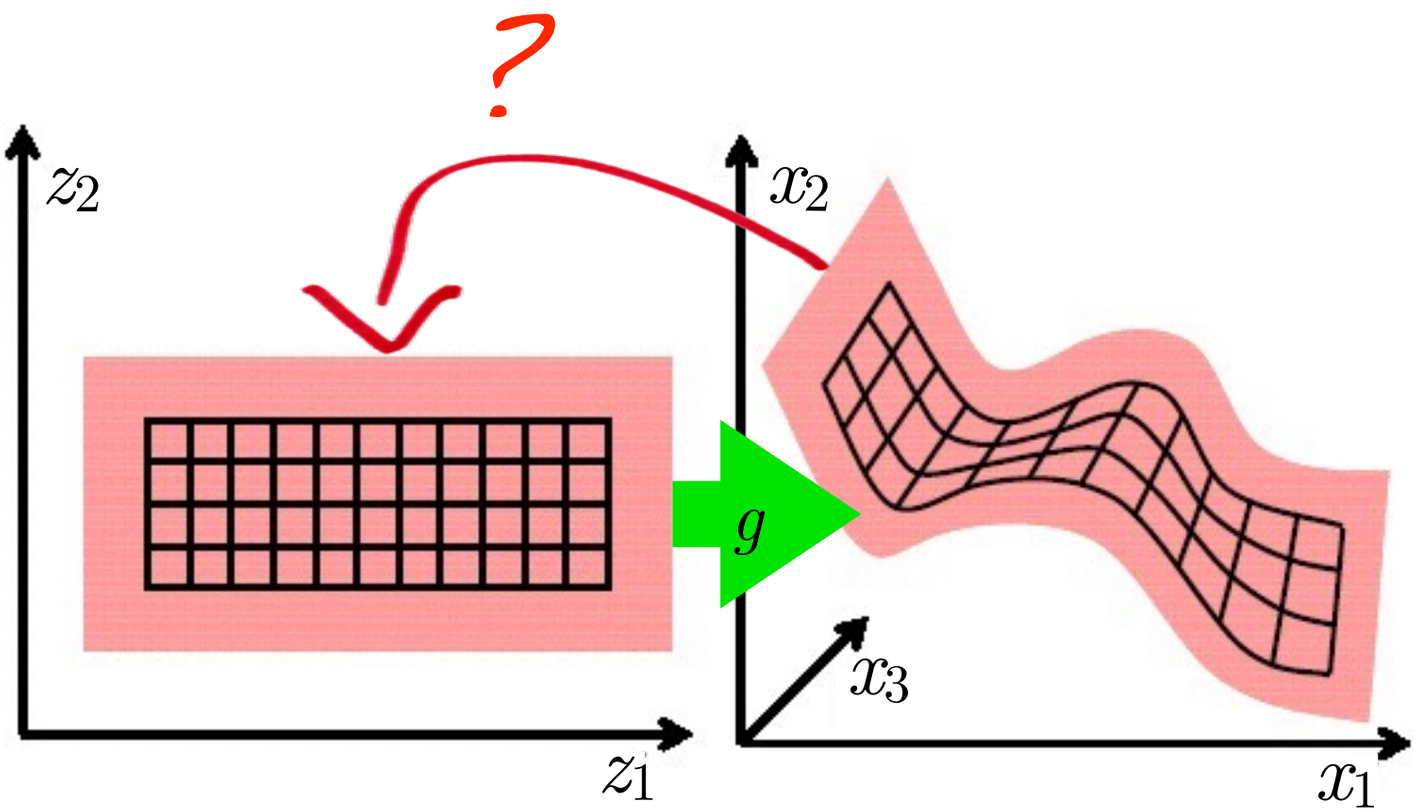


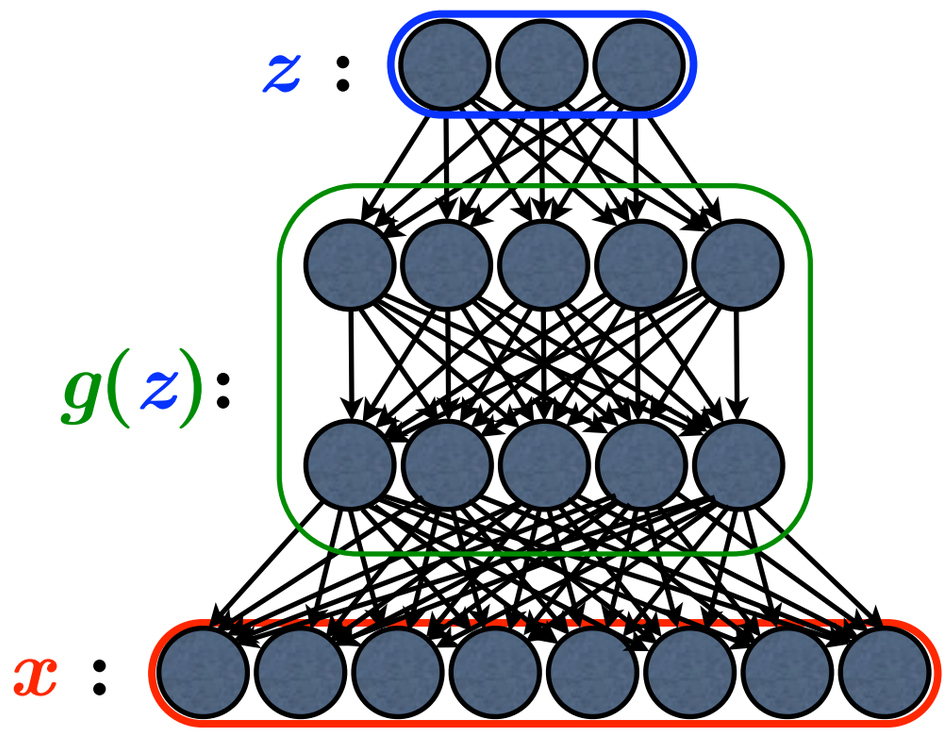


Slide Credit: Aaron Courville

Variational Auto-Encoder

- **Where does z come from?** — The classic directed model dilemma.
- Computing the posterior $p(z \mid x)$ is intractable.
- We need it to train the directed model.





The VAE approach: introduce an inference machine $q_\phi(z \mid x)$

Slide Credit: Aaron Courville

Variational Methods

Integral problem

$$\log p(x) = \log \int p(x|z)p(z)dz$$

Proposal

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

Slide Credit: Shakir Mohammed

Variational Lower Bound

Integral problem

$$\log p(x) = \log \int p(x|z)p(z)dz$$

Proposal

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

Importance Weight

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

Jensen's inequality

$$\log \int p(x)g(x)dx \geq \int p(x) \log g(x)dx$$

$$\log p(x) \geq \int q(x) \log \left(p(x|z)\frac{p(z)}{q(z)} \right) dz$$

$$= \int q(z) \log p(x|z) - \int q(z) \log \frac{q(z)}{p(z)}$$

Slide Credit: Shakir Mohammed

Variational Lower Bound

Integral problem

$$\log p(x) = \log \int p(x|z)p(z)dz$$

Proposal

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

Importance Weight

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

Jensen's inequality

$$\log \int p(x)g(x)dx \geq \int p(x) \log g(x)dx$$

$$\log p(x) \geq \int q(x) \log \left(p(x|z)\frac{p(z)}{q(z)} \right) dz$$

$$= \int q(z) \log p(x|z) - \int q(z) \log \frac{q(z)}{p(z)}$$

Variational lower bound

$$= \mathbb{E}_{q(z)}[\log p(x|z)] - KL[q(z)\|p(z)]$$

Slide Credit: Shakir Mohammed

Variational Auto-Encoder

- 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: $p_\theta(x) \geq \mathcal{L}(\theta, \phi, x)$

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

Slide Credit: Aaron Courville

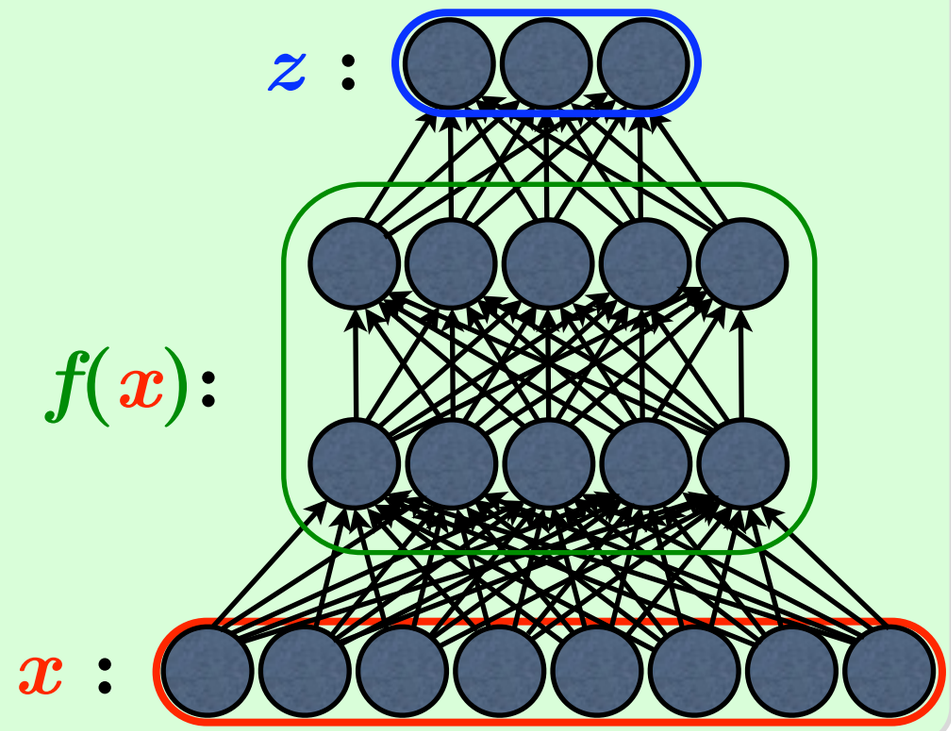
Variational Auto-Encoder

- 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:

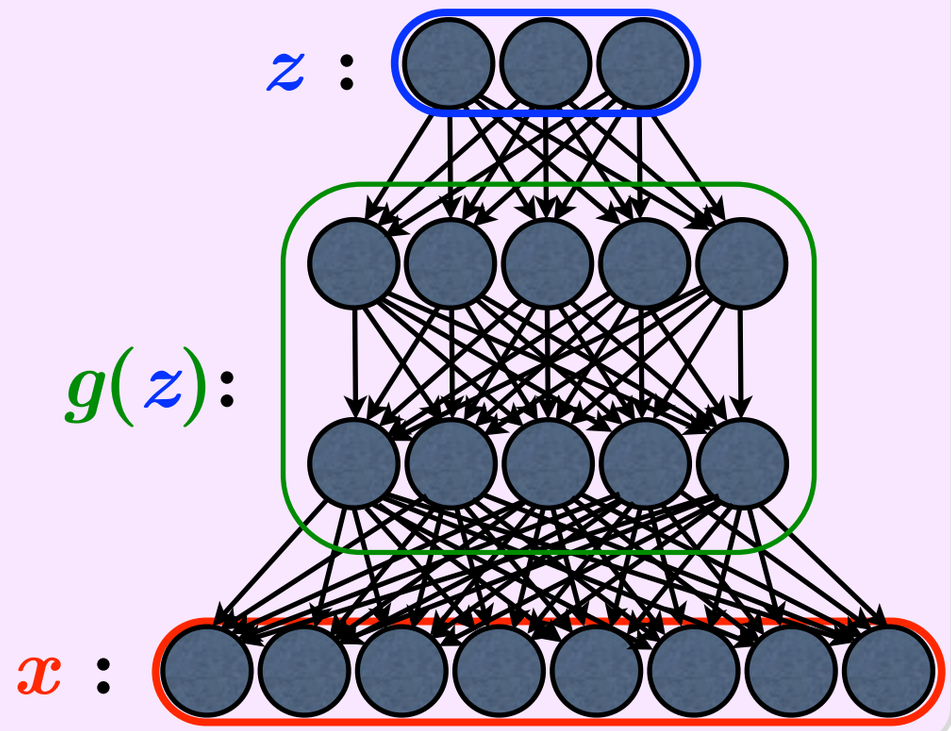
$$\mathcal{L}(\theta, \phi, x) = -D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| \, p_\theta(z)\right) + \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right]$$

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

$$q_\phi(z \mid x) = q(z; f(x, \phi))$$



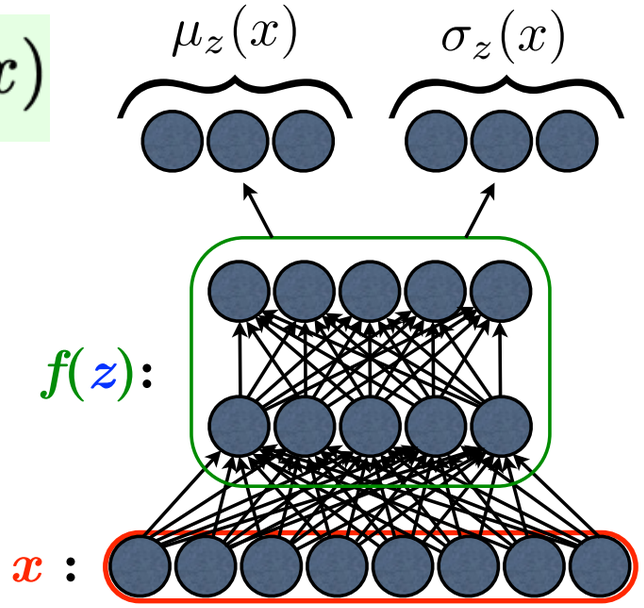
$$p_\theta(x \mid z) = p(x; g(z, \theta))$$

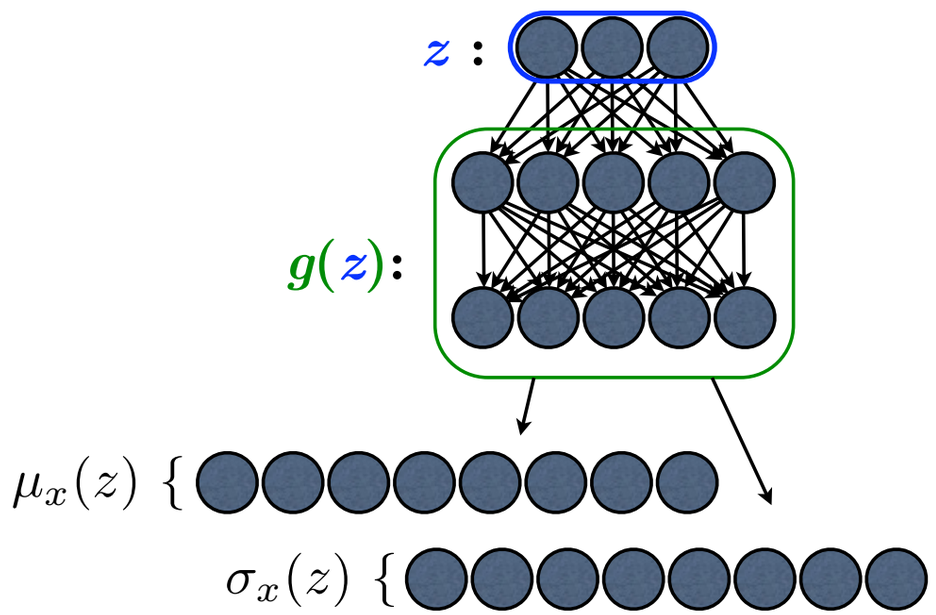


Slide Credit: Aaron Courville

Parametrization

$q_\phi(z \mid x)$





$p_\theta(x \mid z)$

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

Let's say we parametrize $q_\phi(z \mid x)$ **as isotropic Gaussian**

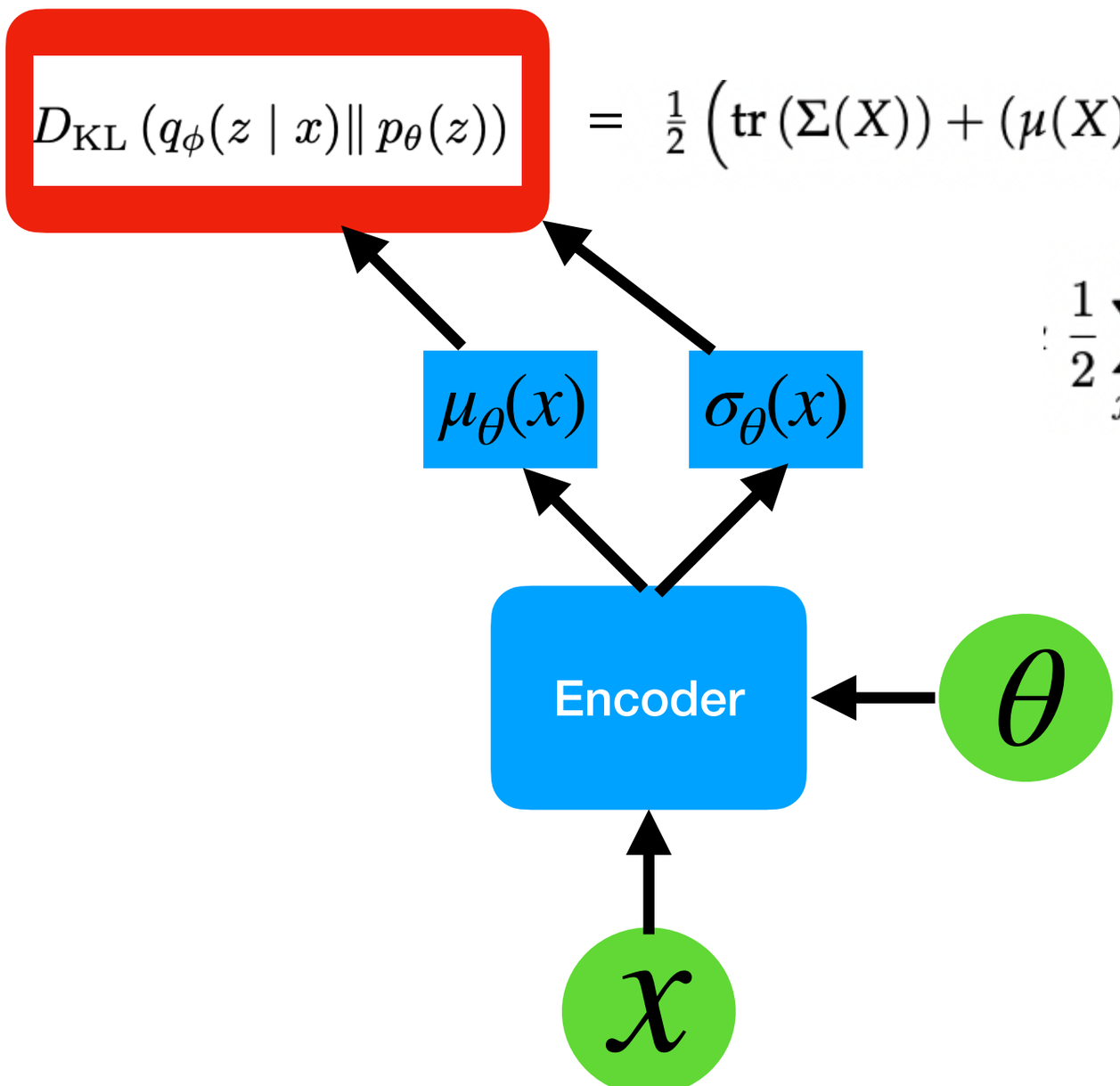
=

How do we take gradients here?

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

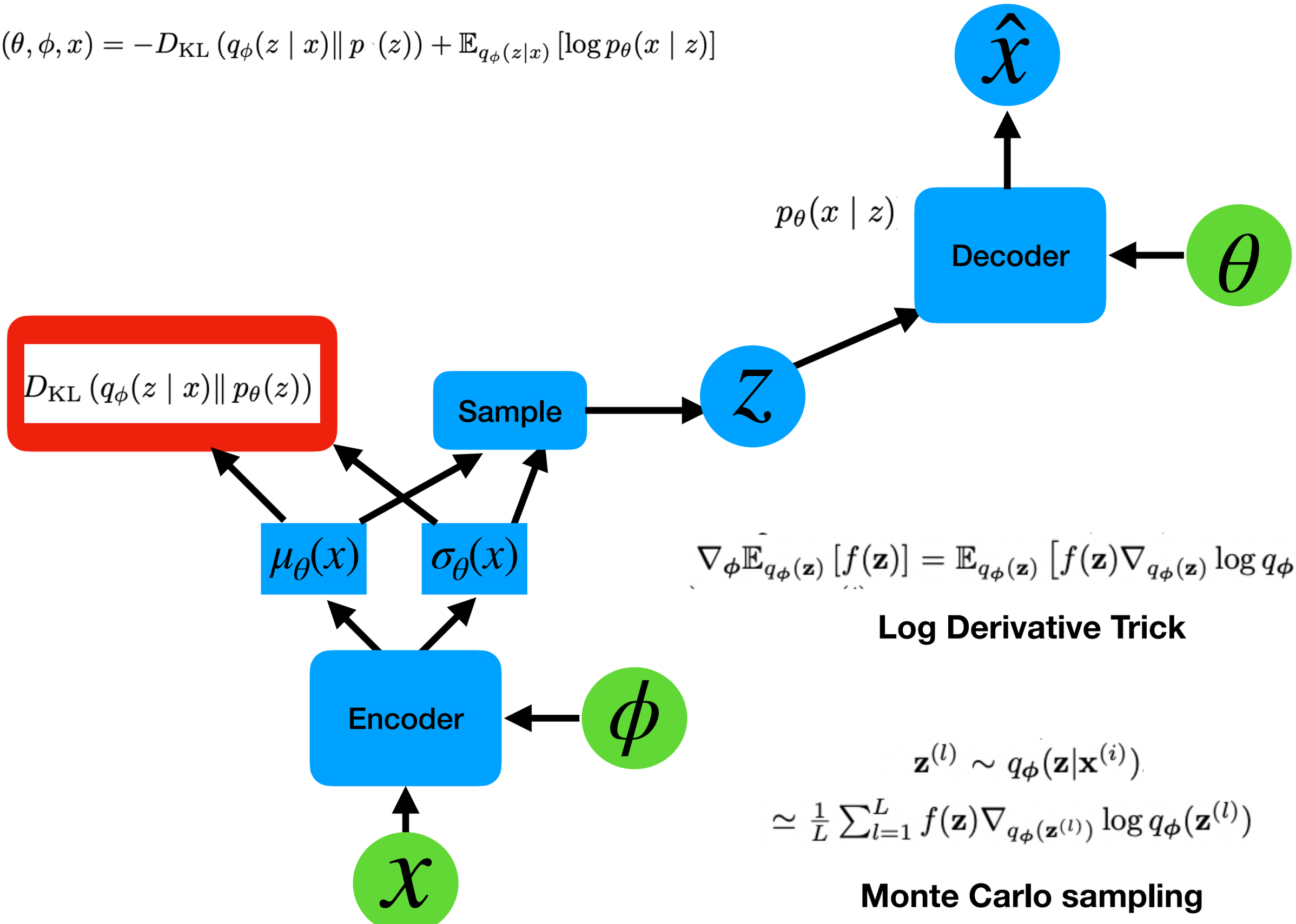
$$D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| p_\theta(z)\right) = \frac{1}{2}\left(\mathrm{tr}\left(\Sigma(X)\right) + \left(\mu(X)\right)^\top \left(\mu(X)\right) - k - \log\det\left(\Sigma(X)\right)\right)$$

$$: \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right) .$$



How do we take gradients through the sampling

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



$$\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}\left[f(\mathbf{z})\right] = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}\left[f(\mathbf{z}) \nabla_{q_{\boldsymbol{\phi}}(\mathbf{z})} \log q_{\boldsymbol{\phi}}(\mathbf{z})\right]$$

Log Derivative Trick

$$\mathbf{z}^{(l)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})$$

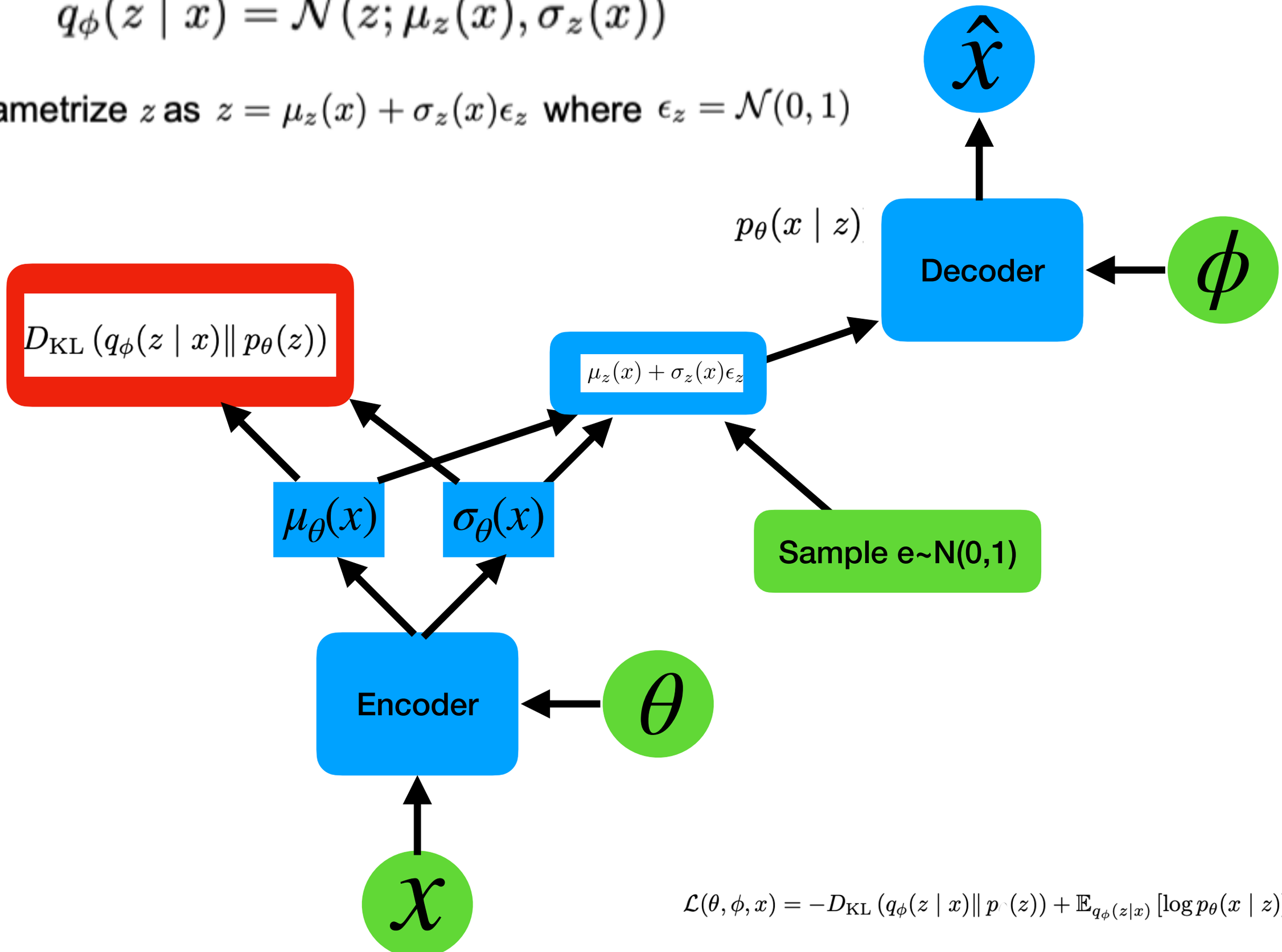
$$\simeq \frac{1}{L} \sum_{l=1}^{L} f(\mathbf{z}) \nabla_{q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)})} \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)})$$

Monte Carlo sampling

Reparametrization Trick

$$q_\phi(z \mid x) = \mathcal{N}(z; \mu_z(x), \sigma_z(x))$$

Parametrize z as $z = \mu_z(x) + \sigma_z(x)\epsilon_z$ where $\epsilon_z = \mathcal{N}(0,1)$



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

Side Note Log Derivative Trick

$$\nabla_\theta \mathbb{E}_{p(z;\theta)}[f(z)] = \int \nabla_\theta p(z;\theta) f(z) dz$$

$$= \int \frac{p(z;\theta)}{p(z;\theta)} \nabla_\theta p(z;\theta) f(z) dz$$

$$= \int p(z;\theta) \nabla_\theta \log p(z;\theta) f(z) dz = \mathbb{E}_{p(z;\theta)}[f(z) \nabla_\theta \log p(z;\theta)]$$

$$\approx \frac{1}{S} \sum_{s=1}^{S} f(z^{(s)}) \nabla_\theta \log p(z^{(s)};\theta) \quad z^{(s)} \sim p(z)$$ **Monte Carlo**

Slide Credit: Shakir Mohammed

Reparametrization “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 $\epsilon_z = \mathcal{N}(0, 1)$

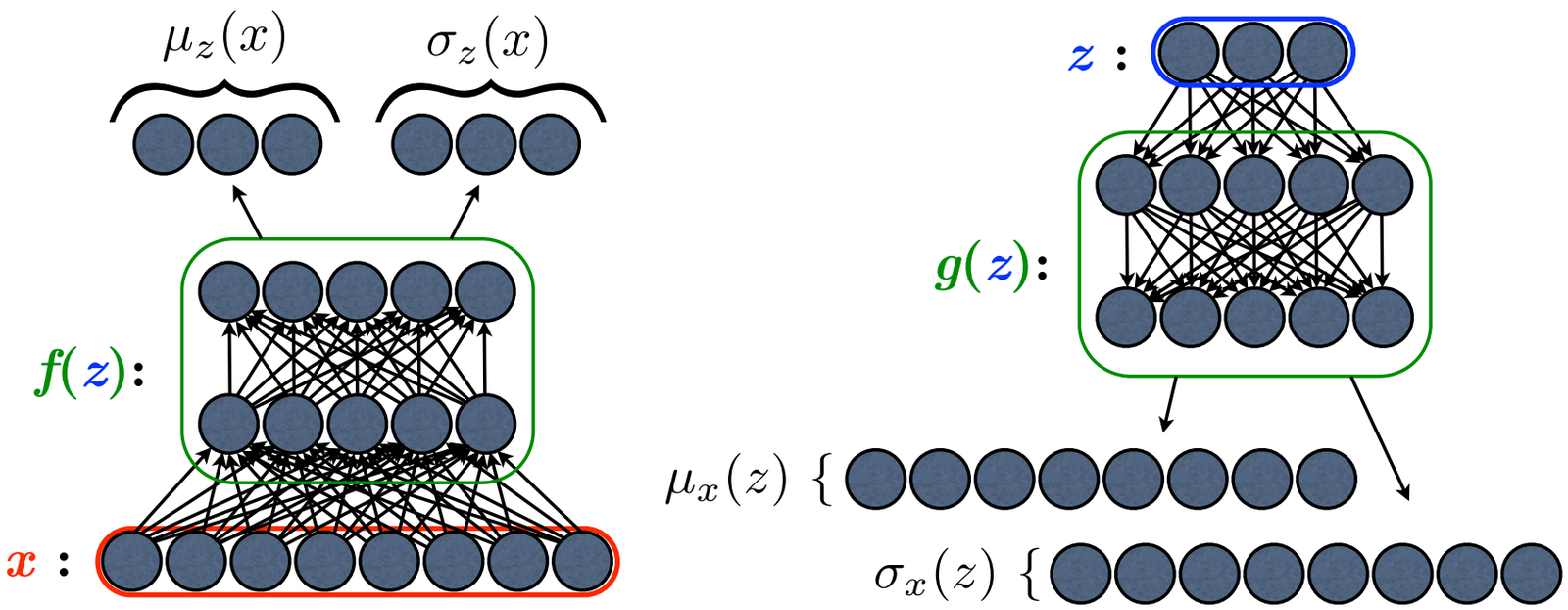


Image Credit: Aaron Courville

Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

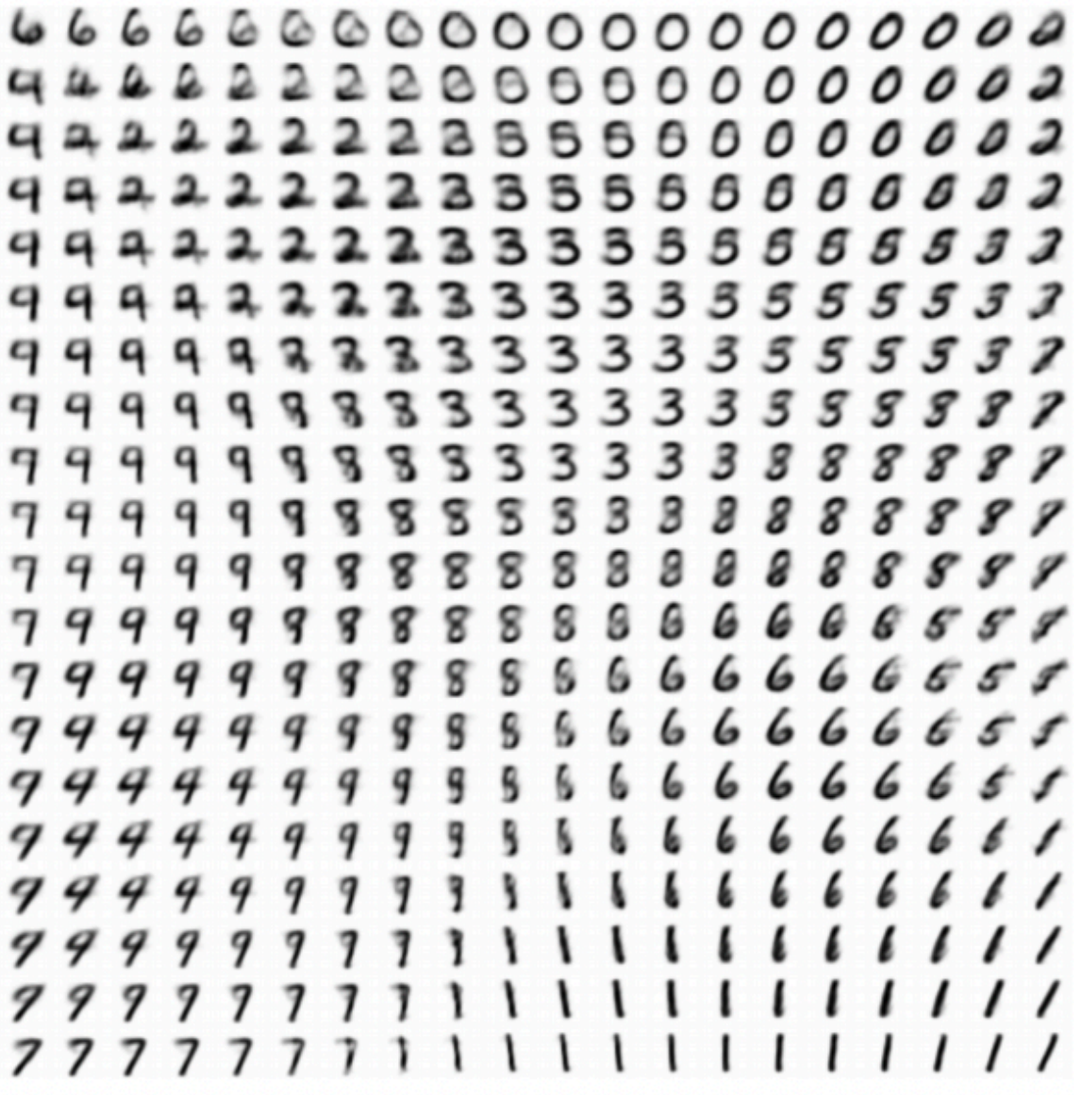
Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Implementing VAEs

https://colab.research.google.com/github/smartgeometry-ucl/dl4g/blob/master/variational_autoencoder.ipynb

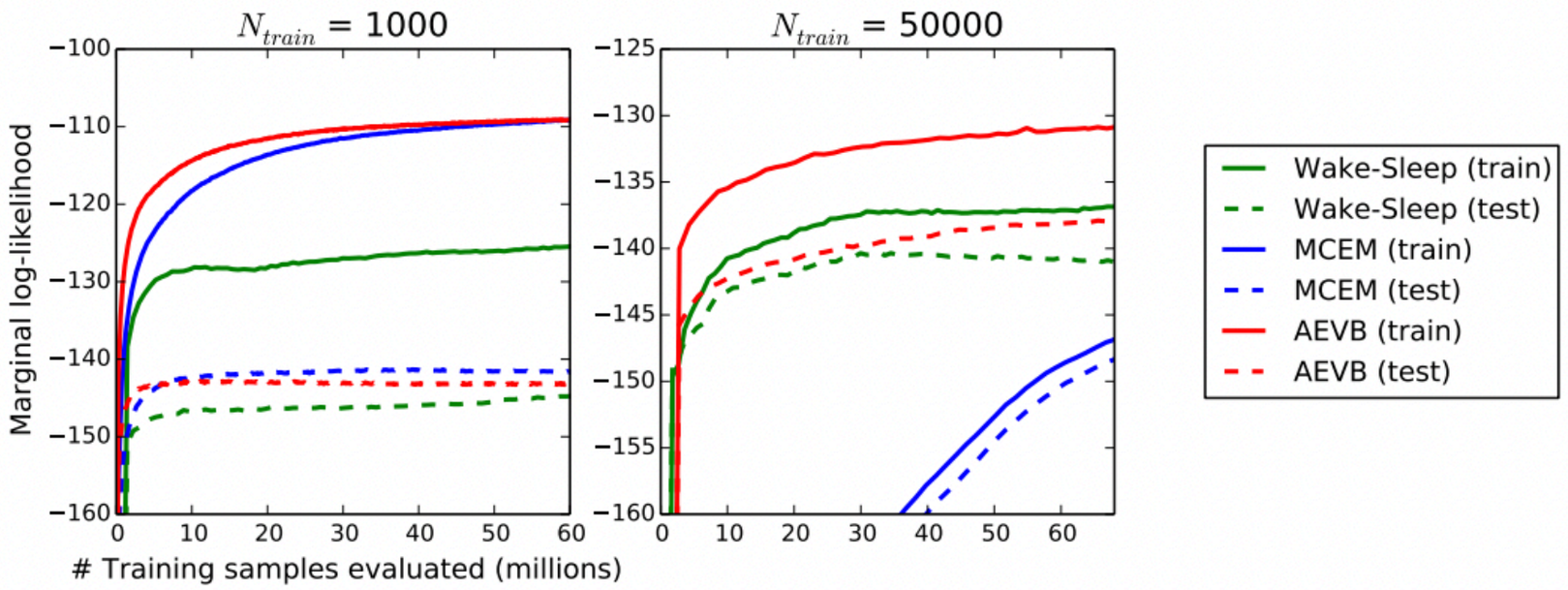
Variational Autoencoders Results

(a) Learned Frey Face manifold



(b) Learned MNIST manifold

Variational Autoencoders Results



- Able to train more stable and obtain better likelihood estimates on withheld data than other methods at the time

- Improved likelihood not always leading to better image quality

What are some issues?

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

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

Reconstruction objective is poor for e.g. images

Better likelihood not always related to visual quality

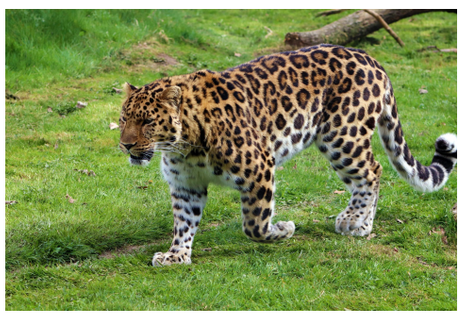
What if we can try to learn a good loss

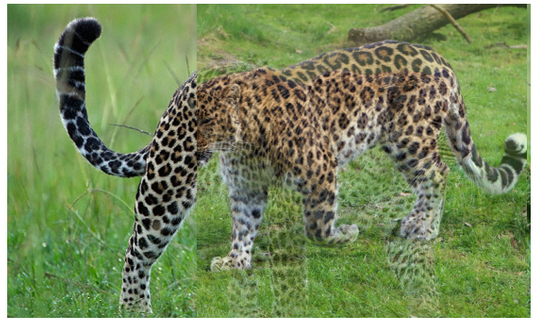
$$= -D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| p_\theta(z)\right) + \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right]$$

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

Reconstruction objective is poor for e.g. images

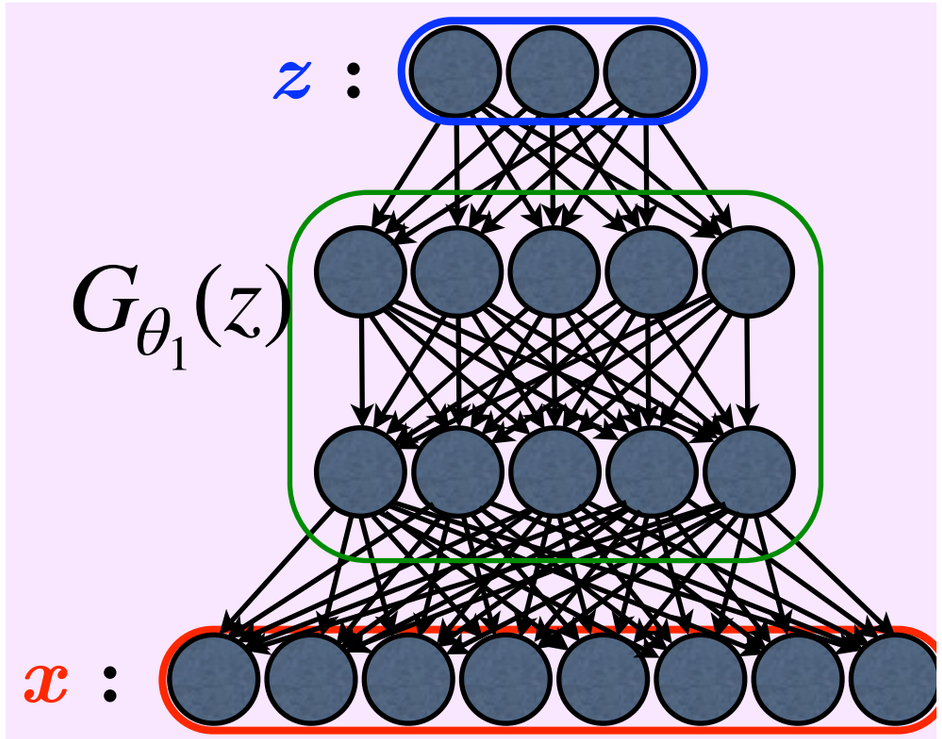
Better likelihood not always related to visual quality





Generative Adversarial Network

Generator



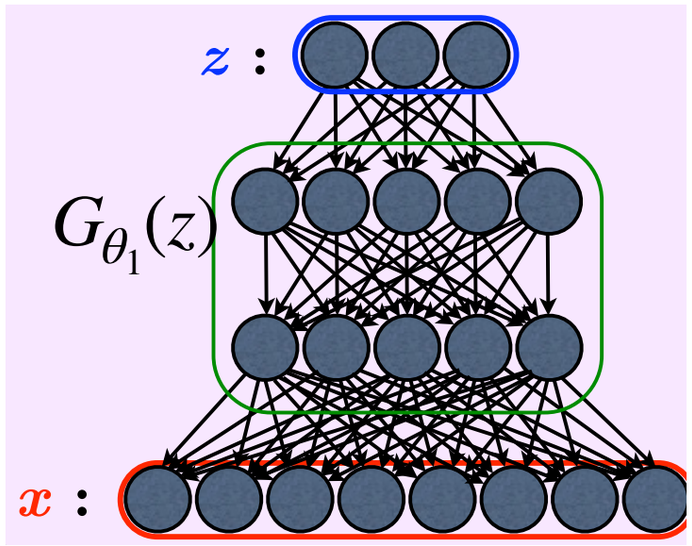
Generative Adversarial Nets

Ian J. Goodfellow[*] **Jean Pouget-Abadie**[†] **Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair**[‡] **Aaron Courville, Yoshua Bengio**[§]
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

Z is simple and transforms to a complicated distribution

Generative Adversarial Network

Generator



Real Data distribution

$D(z)$

Discriminator

Generative Adversarial Nets

Ian J. Goodfellow[*], **Jean Pouget-Abadie**[†], **Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair**[‡], **Aaron Courville, Yoshua Bengio**[§]
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

Discriminator Learns

$G_{\theta_1}(z)$

Generator

Real Data distribution

$D_{\theta_2}(z)$

Discriminator

$$= \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[-\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}}[-\log(1 - D(G(\mathbf{z})))]$$

Discriminator loss is the flip of the generator

Generator Loss

$G_{\theta_1}(z)$

Generator

Real Data distribution

$D_{\theta_2}(z)$

Discriminator

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[-\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}}[-\log(1 - D(G(\mathbf{z})))]$$

Generator Loss is the flip of the discriminator
Independent of the first term

Min Max Formulation

$G_{\theta_1}(z)$

Generator

Real Data distribution

$D_{\theta_2}(z)$

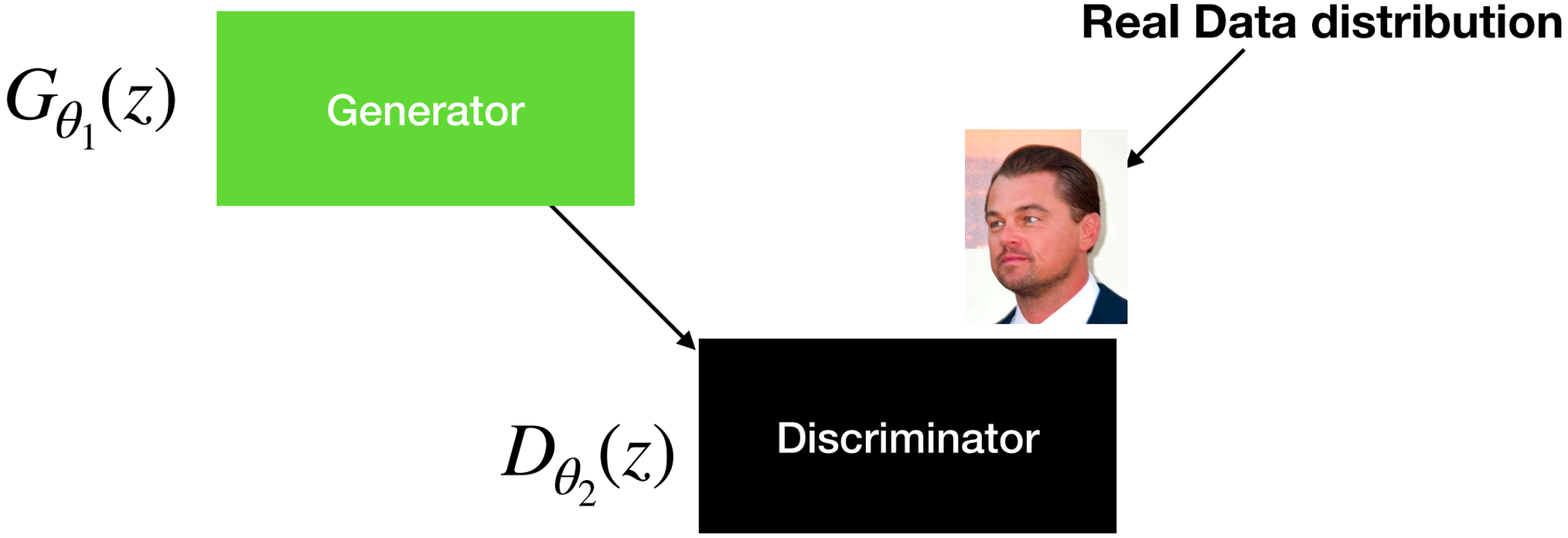
Discriminator

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Generative Adversarial Nets

Ian J. Goodfellow[*], **Jean Pouget-Abadie**[†], **Mehdi Mirza**, **Bing Xu**, **David Warde-Farley**, **Sherjil Ozair**[‡], **Aaron Courville**, **Yoshua Bengio**[§]
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

Training



$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

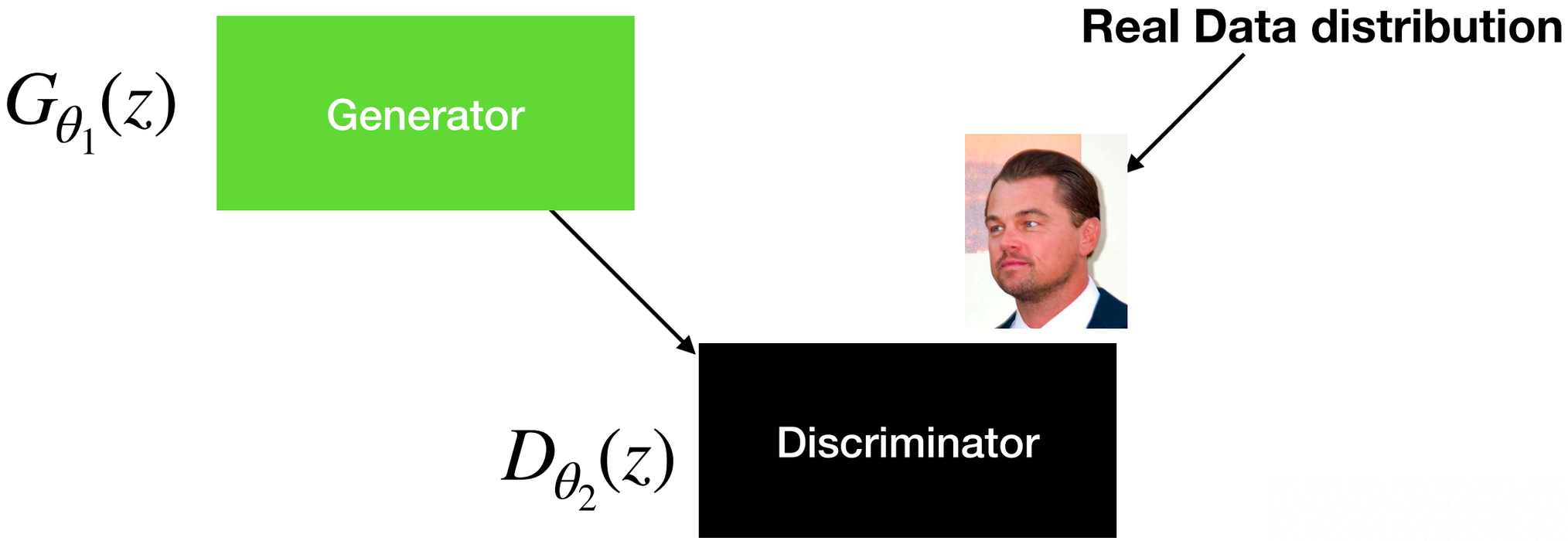
end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

Tweaking the generator loss



$$\mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]$$

- If generator is bad the D(G(z)) -> 0 and we saturate … difficult to get gradients (similar to how sigmoid activations have issues)

- Modification $\mathbb{E}_{\mathbf{z}}[-\log D(G(\mathbf{z}))]$

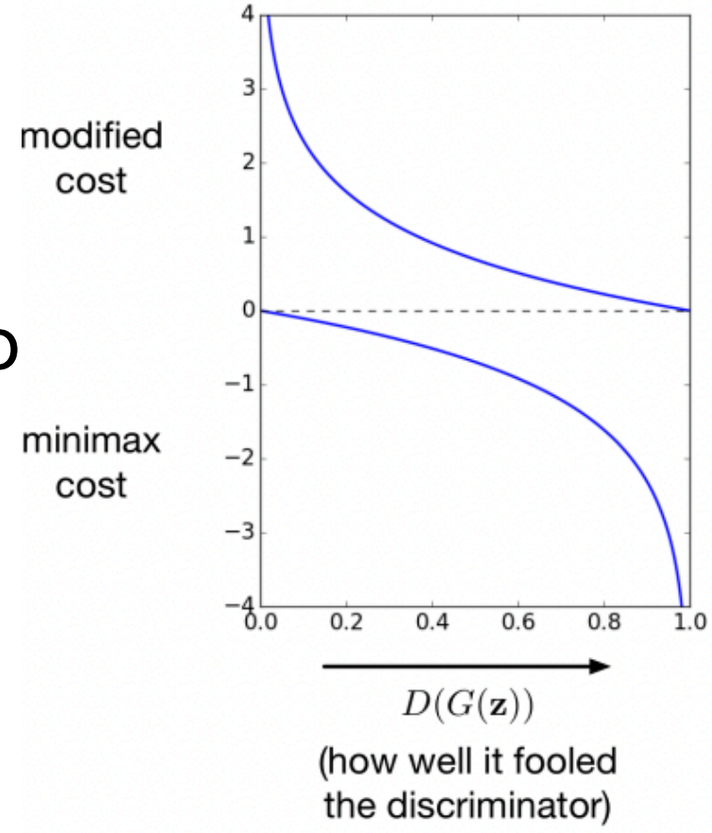


Image Credit: Roger Grosse

Implementation

https://colab.research.google.com/github/smartgeometry-ucl/dl4g/blob/master/gan.ipynb#scrollTo=zspHCtRzBkBq

Image Credit: Roger Grosse

Stability of Learning

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

- Many issues arise in GAN training due to the dynamics between discriminator and generator

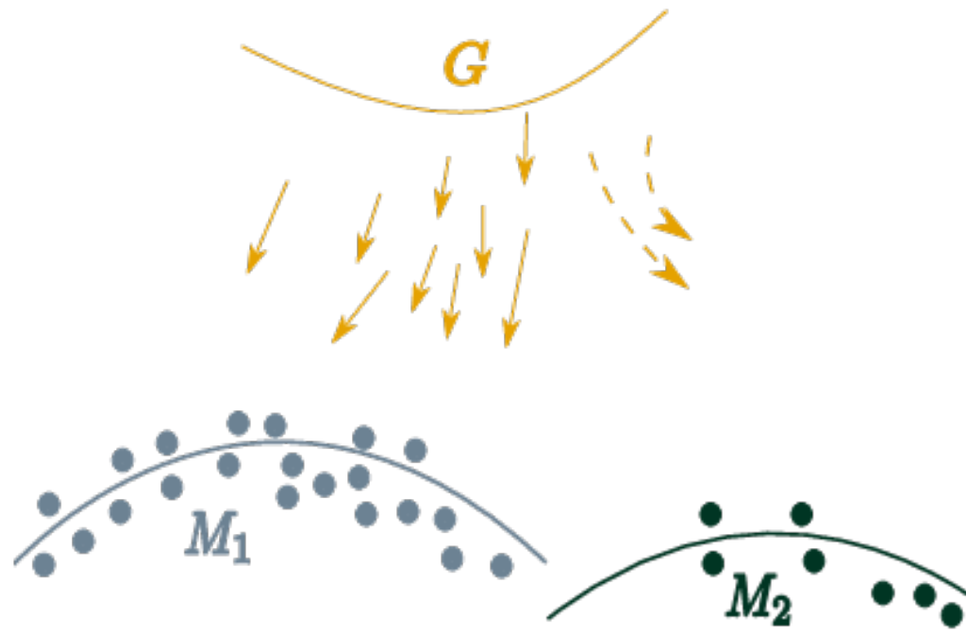
- If D is weak or slow

  - Easy to construct “adversarial examples for D” .. small perturbations that fool D

  - Mode collapse .. G starts spitting out the same example which it finds is good for fooling D

- If D is too strong

  - G might not be able to “catch up”



GAN Zoo

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

- Since GAN appeared in 2014 many works have studied ways to stabilize the training procedures
- Most effective are gradient penalties and spectral normalization on D
- Many many heuristics for training developed

Modern GAN

LARGE SCALE GAN TRAINING FOR HIGH FIDELITY NATURAL IMAGE SYNTHESIS

Andrew Brock[*†]
Heriot-Watt University
ajb5@hw.ac.uk

Jeff Donahue[†]
DeepMind
jeffdonahue@google.com

Karen Simonyan[†]
DeepMind
simonyan@google.com

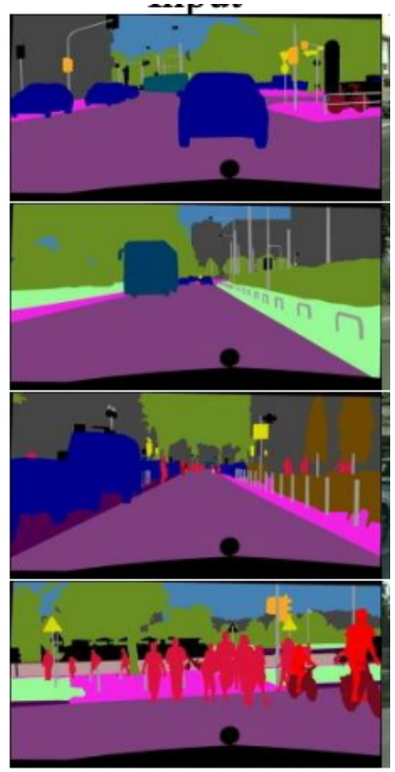
Modern GAN

GAN

- GAN’s produce faithful and realistic looking images
- They can operate well in high resolution images and are less blurry than MSE based objectives
- They maintain a latent variable and for CNN’s can be efficient in decoding

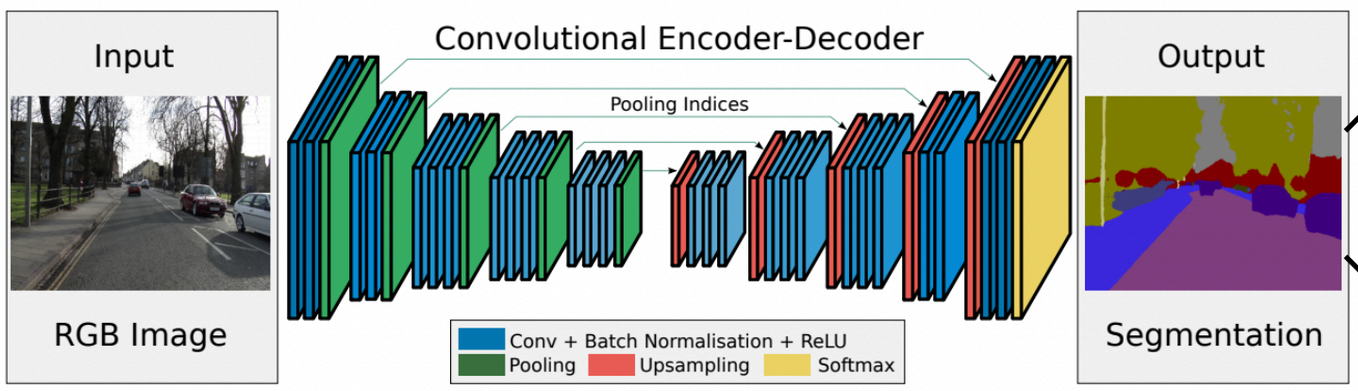
Adversarial Training

Adversarial Loss can constrain output distribution to match a known set of real targets



Real Segmentation

Discriminator

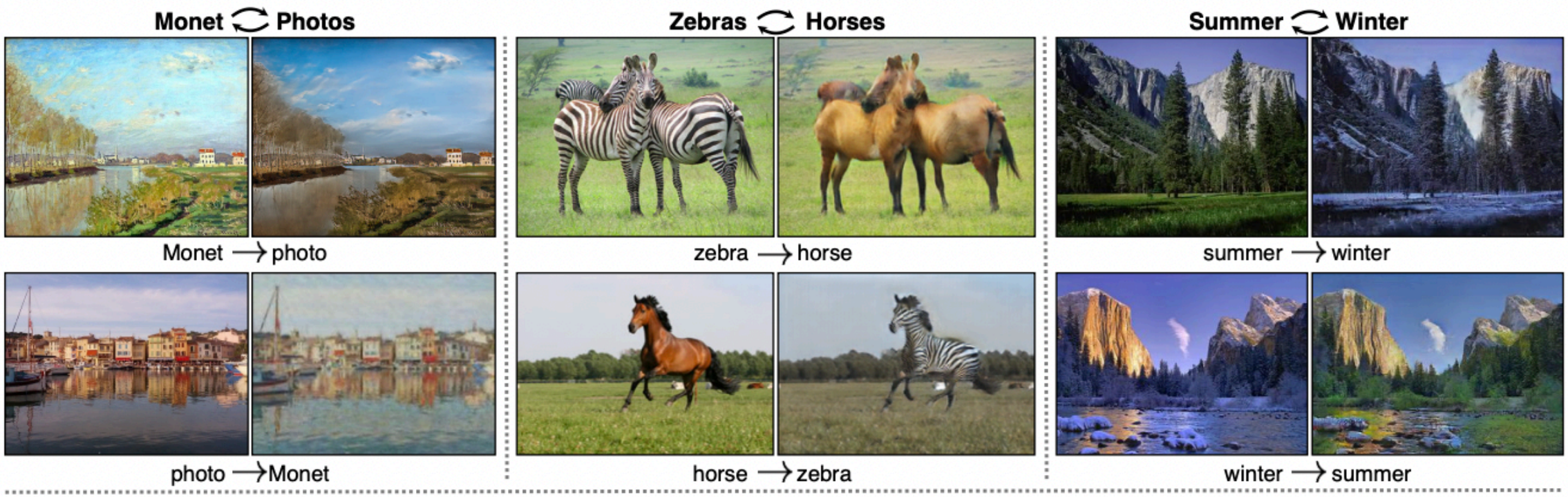


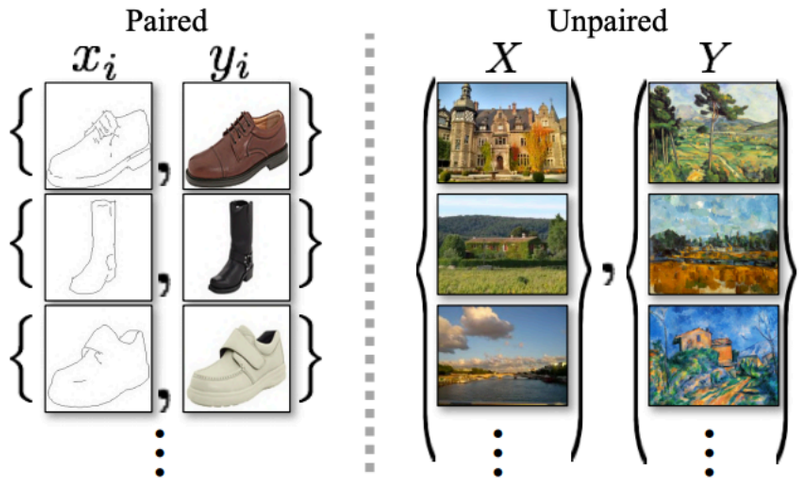
+ noise

Segmentation Loss

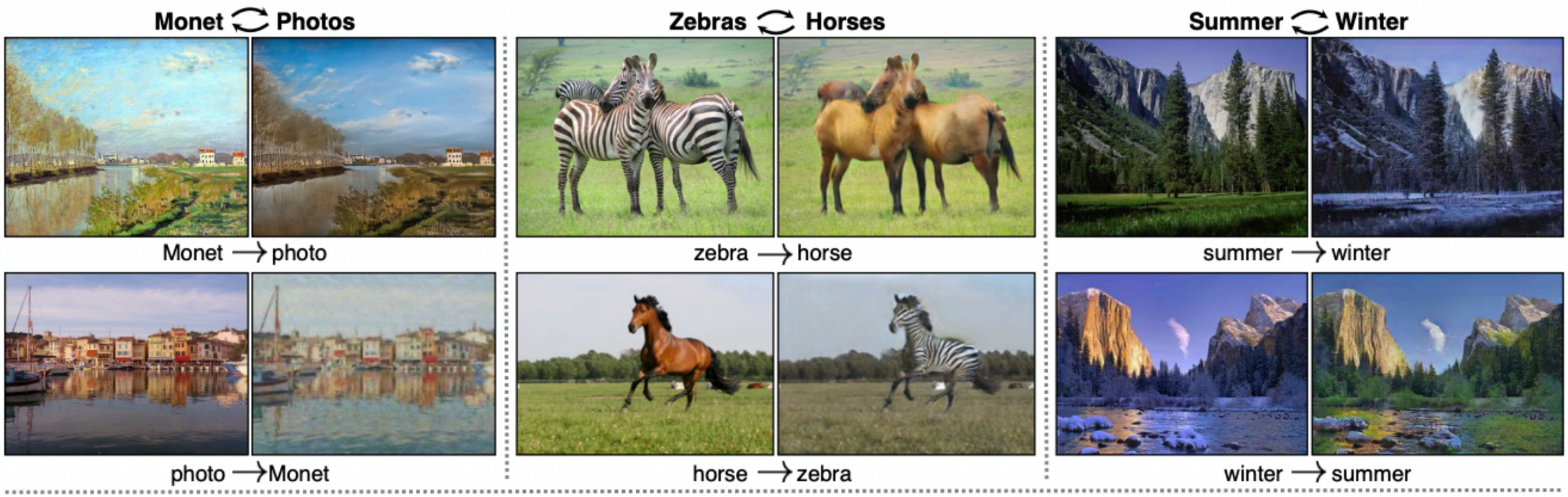
For multi-variable outputs we can combine the ground truth targets with an adversarial loss

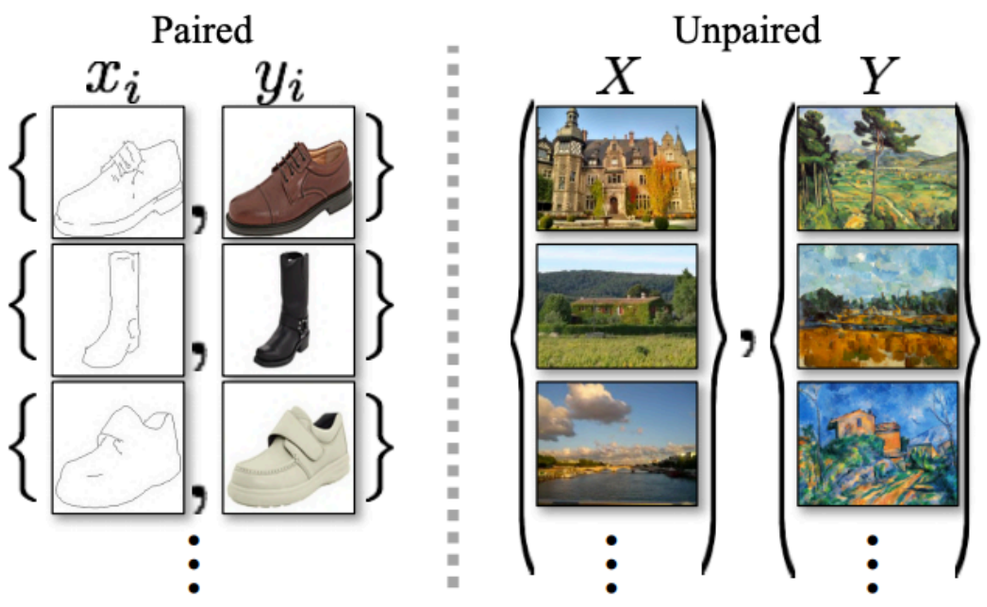
Unpaired Translation



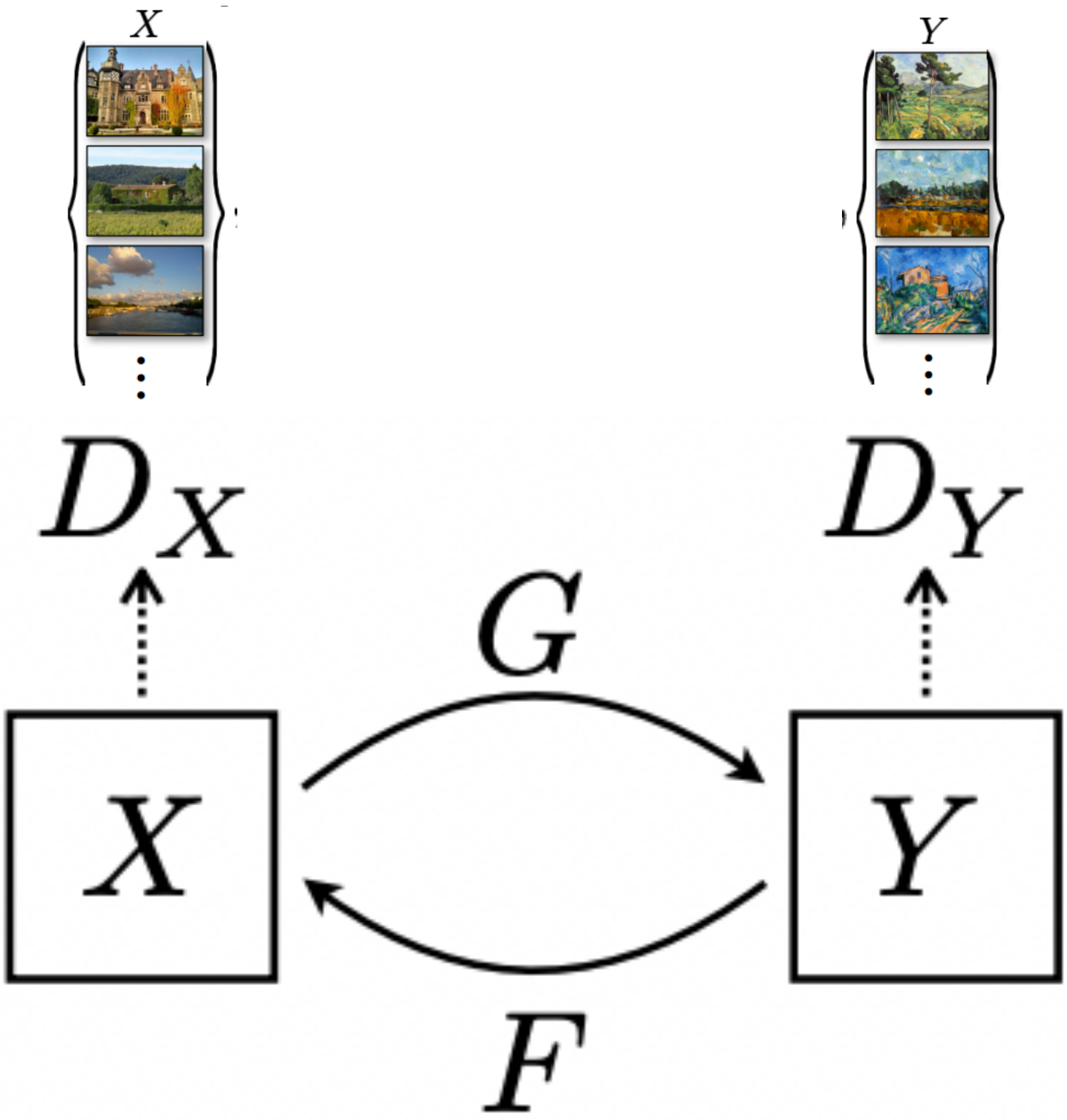


Unpaired Translation





Unpaired Translation



$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))],$$

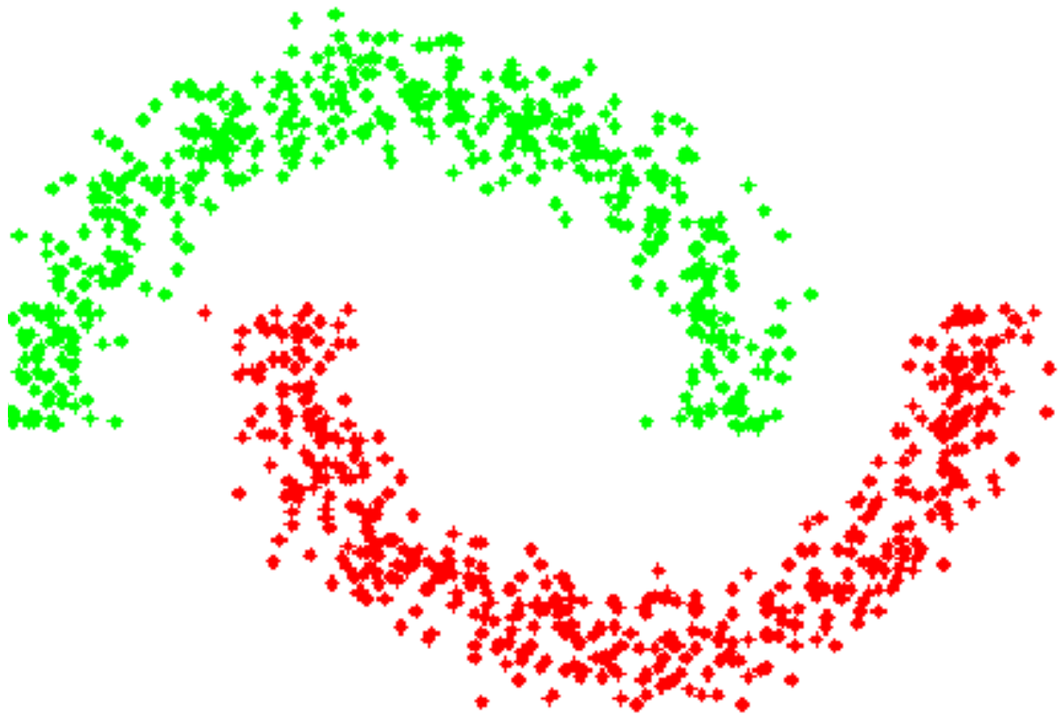
$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F),$$

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

Other Constraints on mapping possible

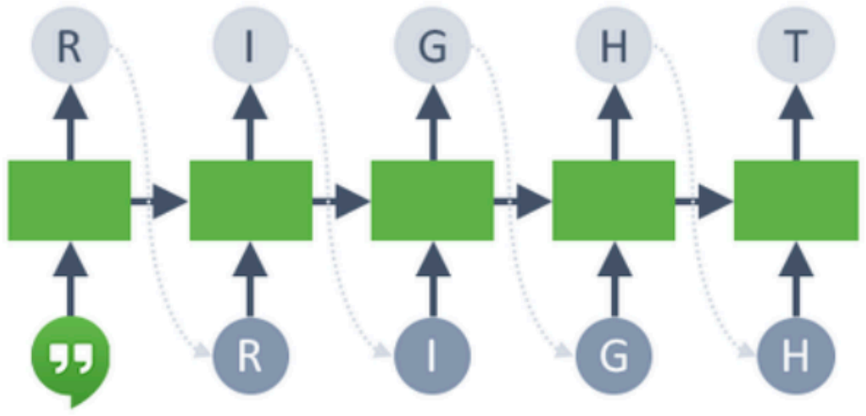
Semi-Supervised Learning

- Generative models are often used for semi-supervised learning
- At a high level this is typically done by adding the generative objective to the supervised objective and optimizing jointly

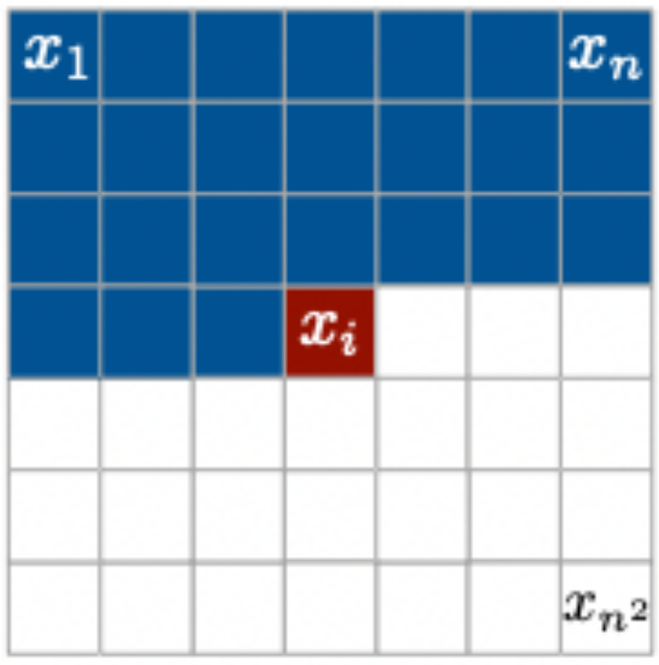


Autoregressive Models

- Autoregressive models - predict next sequence step from all previous steps $p(\mathbf{x}) = \prod_{i=1}^{D} p(x_i \mid \mathbf{x}_{<i})$
  - Various classical methods with assumptions (e.g. next state depends only on previous)
  - We have seen already 2 autoregressive models
    - RNNs for text generation
    - Transformers for text generation
      - Use full history to make next prediction
  - Learn by maximizing the likelihood - can apply softmax + cross entropy
- Can we apply this idea to images?



Auto-Regressive Models For Images



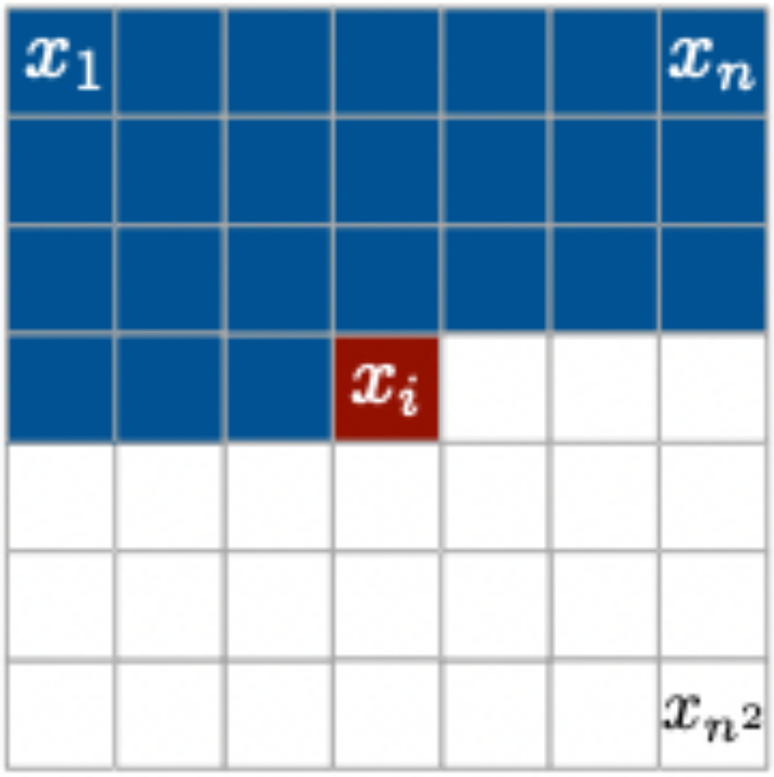
Context

Auto-Regressive Models Images

Pixel Recurrent Neural Networks

Aäron van den Oord AVDNOORD@GOOGLE.COM
Nal Kalchbrenner NALK@GOOGLE.COM
Koray Kavukcuoglu KORAYK@GOOGLE.COM

Google DeepMind



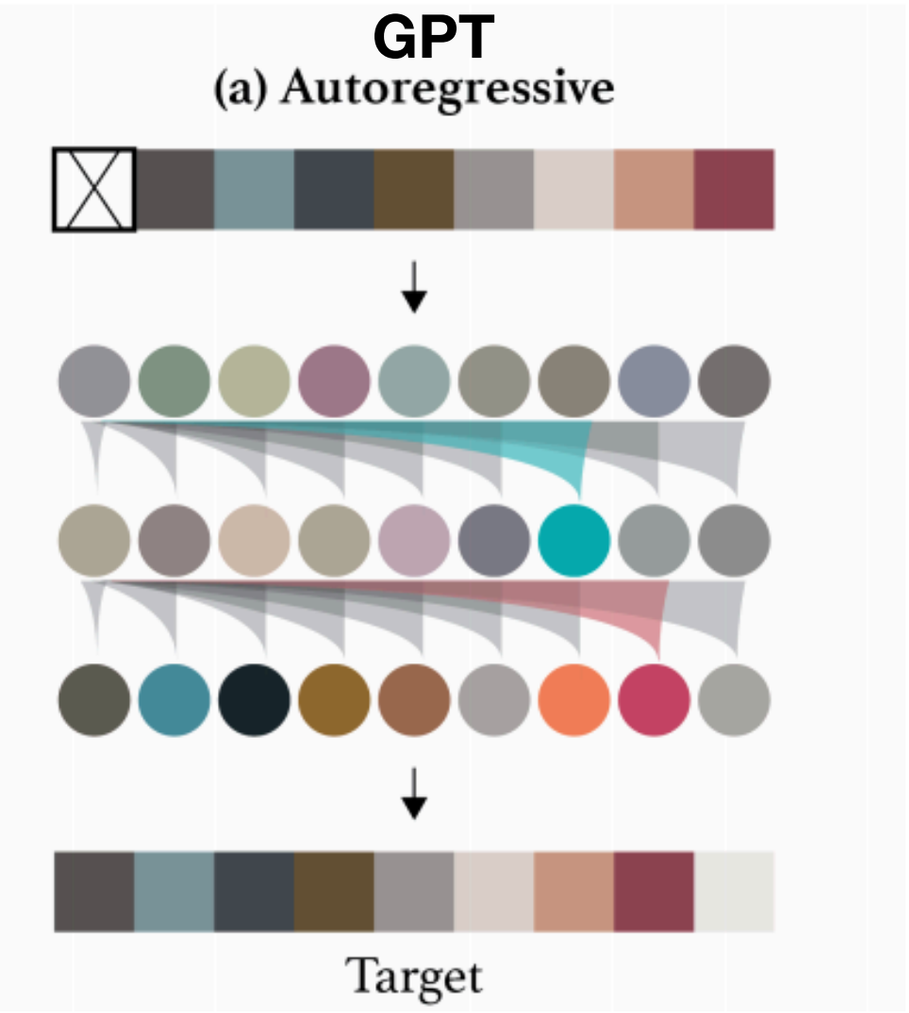
Context

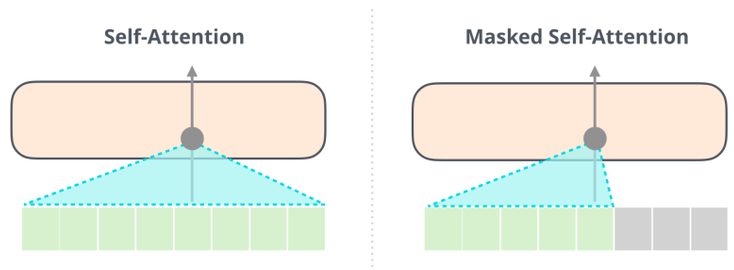
Figure 1. Image completions sampled from a PixelRNN.

Advantages of Auto-Regressive Models

- Advantages
  - Simple objective function
  - No need for approximate inference like in VAE
  - Stable convergence properties
- Disadvantages
  - “Causal” — can be slow and hard to parallelize

GPT





Queries

robot	must	obey	orders

X

Keys

robot	must	obey	orders
robot	must	obey	orders
robot	must	obey	orders
robot	must	obey	orders

=

Scores (before softmax)

0.11	0.00	0.81	0.79
0.19	0.50	0.30	0.48
0.53	0.98	0.95	0.14
0.81	0.86	0.38	0.90

Mask+ softmax

Scores

1	0	0	0
0.48	0.52	0	0
0.31	0.35	0.34	0
0.25	0.26	0.23	0.26

RNN are hard to parallelize

Pixel Recurrent Neural Networks

Aäron van den Oord AVDNOORD@GOOGLE.COM
Nal Kalchbrenner NALK@GOOGLE.COM
Koray Kavukcuoglu KORAYK@GOOGLE.COM

Google DeepMind

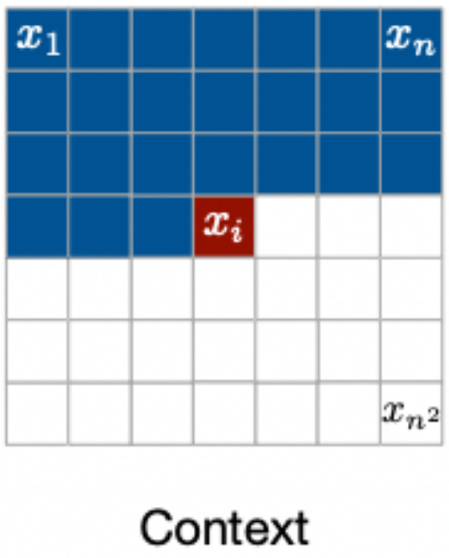
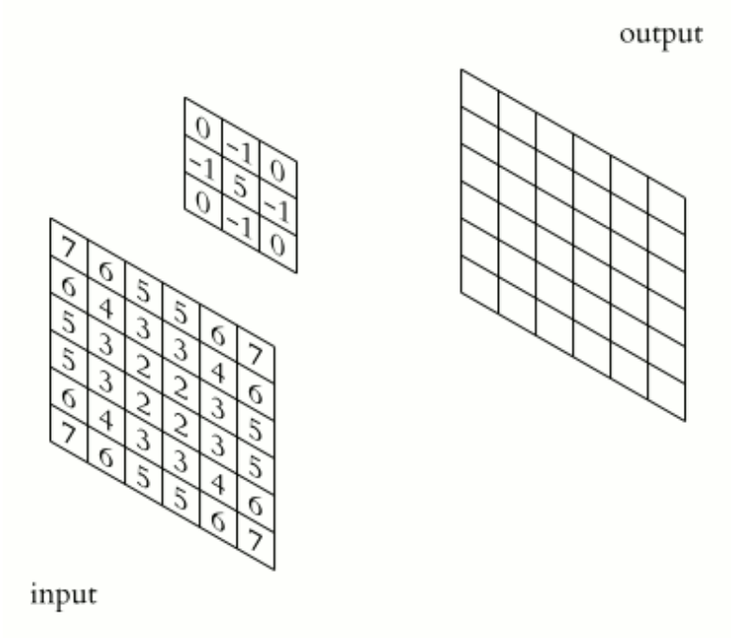
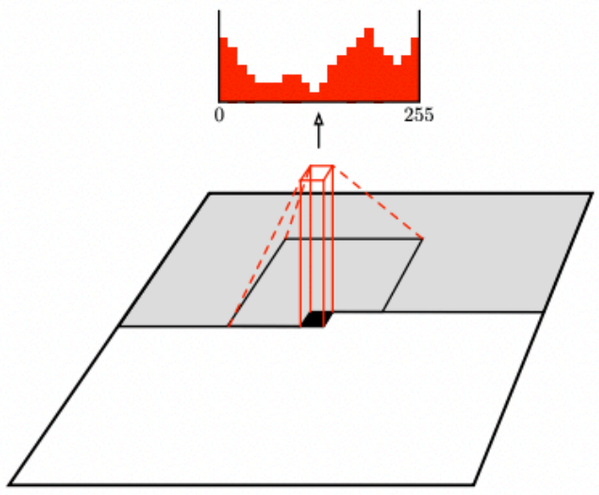
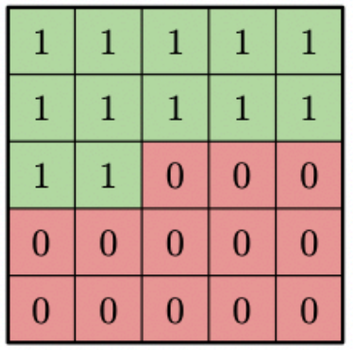


Figure 1. Image completions sampled from a PixelRNN.

PixelCNN







- Apply a CNN with masked con filters to the image
- Output of CNN is same size as input
- Related idea to what we saw in the NMT Transformer decoder and GPT
- Much faster to train but still slow at inference

Conditional Image Generation with PixelCNN Decoders

Aäron van den Oord
Google DeepMind
avdnoord@google.com

Nal Kalchbrenner
Google DeepMind
nalk@google.com

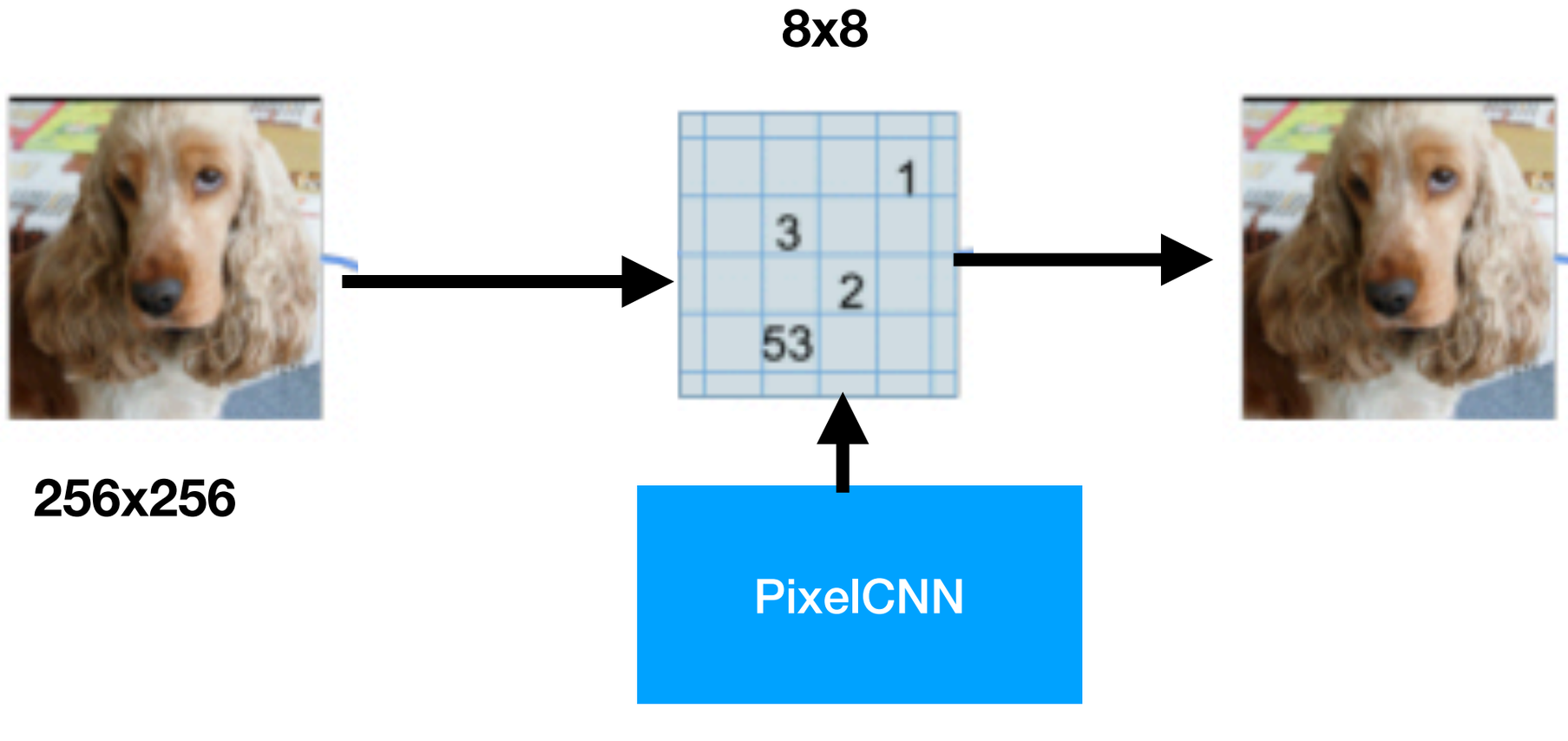
Oriol Vinyals
Google DeepMind
vinyals@google.com

Lasse Espeholt
Google DeepMind
espeholt@google.com

Alex Graves
Google DeepMind
gravesa@google.com

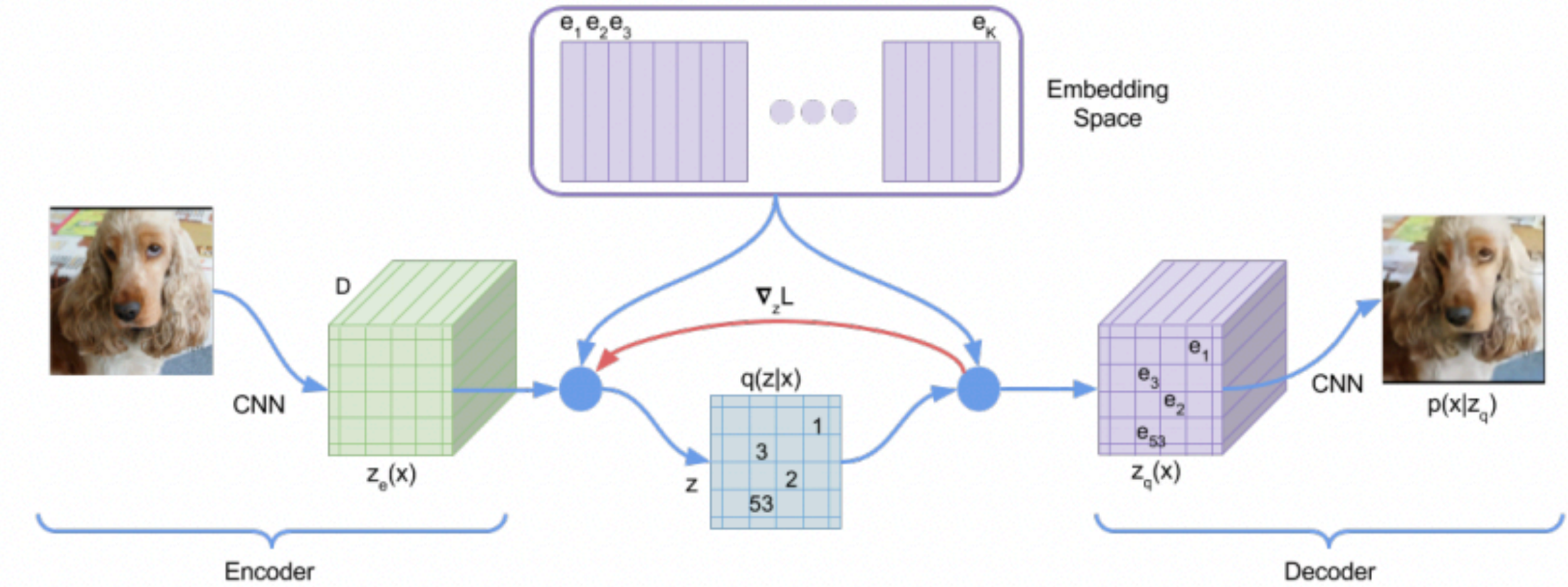
Koray Kavukcuoglu
Google DeepMind
korayk@google.com

Scaling Up AutoRegressive Models



- Idea for scaling up autoregressive model
  - Learn a discrete auto encoder (lower dimensional) that still maintains spatial structure
  - Learn PixelCNN over the discrete latents
  - How do we learn discrete auto encoder?!

VQ-VAE



$$L = \log p(x|z_q(x)) + \|\text{sg}[z_e(x)] - e\|_2^2 + \beta\|z_e(x) - \text{sg}[e]\|_2^2,$$

Neural Discrete Representation Learning

Aaron van den Oord
DeepMind
avdnoord@google.com

Oriol Vinyals
DeepMind
vinyals@google.com

Koray Kavukcuoglu
DeepMind
korayk@google.com

VQ-VAE2

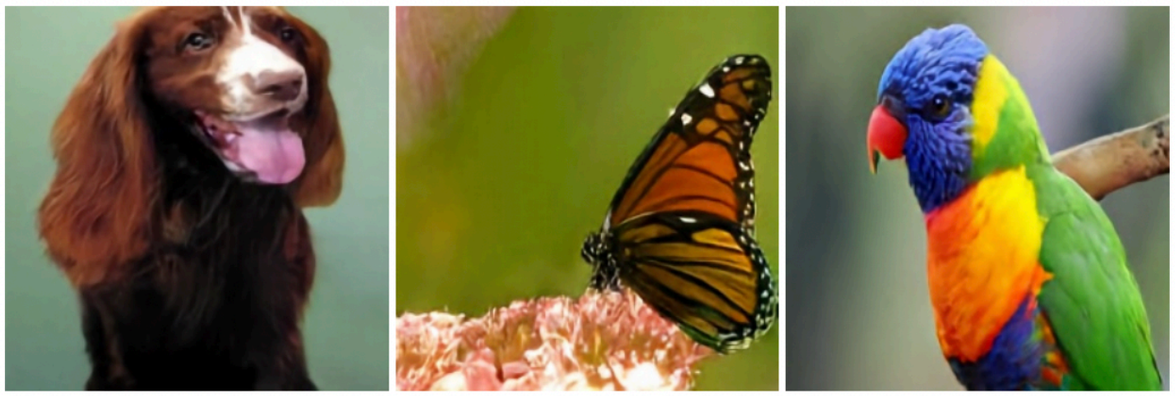
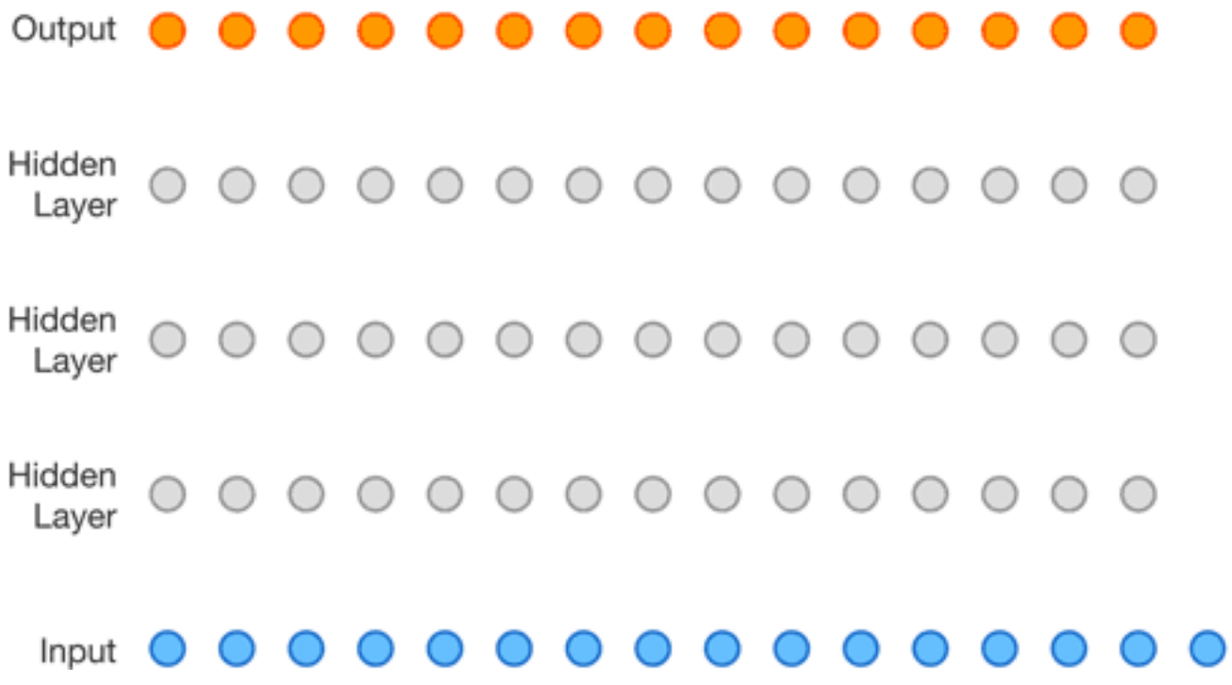


Figure 1: Class-conditional 256x256 image samples from a two-level model trained on ImageNet.

- VQ-VAE2 produces generated images rivalling state of the art generative models

- Same idea as VQ-VAE (pixelcnn + discrete latents) + several levels of discrete latents + engineering magic

WaveNet



- Text -> Speech
- Trained on raw waveform (long sequence)
- Autoregressive
- Causal convolutions -> similar to masked cones
- Dilated convolutions increase receptive field rapidly

WAVENET: A GENERATIVE MODEL FOR RAW AUDIO

Aäron van den Oord **Sander Dieleman** **Heiga Zen**[†]

Karen Simonyan **Oriol Vinyals** **Alex Graves**

Nal Kalchbrenner **Andrew Senior** **Koray Kavukcuoglu**

{avdnoord, sedielem, heigazen, simonyan, vinyals, gravesa, nalk, andrewsenior, korayk}@google.com
Google DeepMind, London, UK
[†] Google, London, UK