

STAT 453: Introduction to Deep Learning and Generative Models

Ben Lengerich

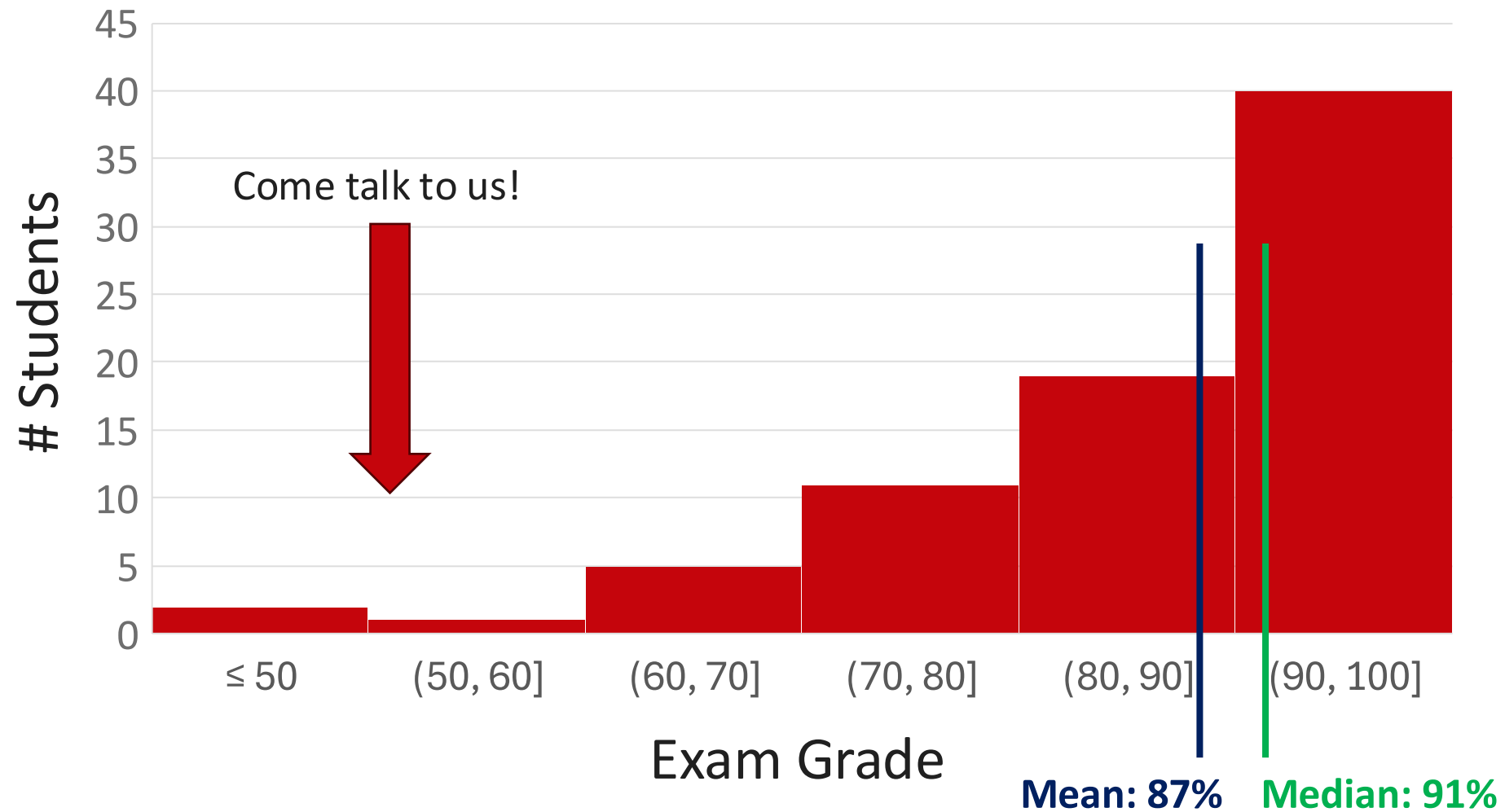
Lecture 14: VAEs and GANs

October 27, 2026

Reading: See course homepage

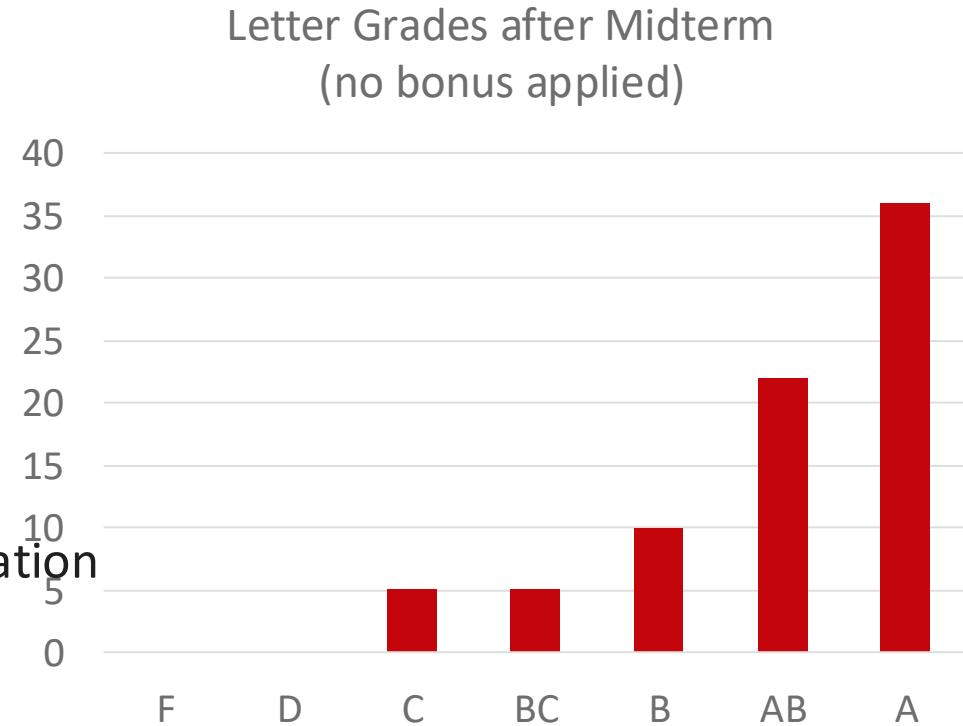


Midterm exam



Grades in context

- In the bank:
 - Project Proposal (5%)
 - Midterm Exam (20%)
 - HWs (8%)
- Looking ahead:
 - Project (25%)
 - Midway Report, Final Report, Presentation
 - HWs (12%)
 - HW3, HW4, HW5
 - Final Exam (30%)
 - Lecture Notes (up to +5%)



Your Feedback

- Thanks for filling out our anonymous [feedback form](#)
- Results:
 - Most rate as an average course with solid alignment of lectures, homeworks, and exams
 - Request for more (visual) examples as we work out theory
 - Homework load seems light (good 😊)
 - Project: Some people don't have compute environment figure out yet → please reach out!

Generative Models



Last time: Two paths to $P(Y|X)$

- **Discriminative:**

Observe X, Y



Learn $P(Y|X)$

- **Generative:**

- Learn $P(X|Y), P(Y)$
- Calculate $P(X) = \int_Y P(X, Y) dY$

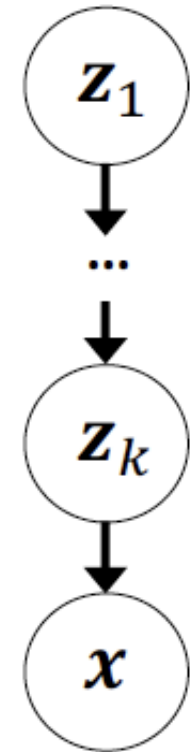
Observe X, Y



$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

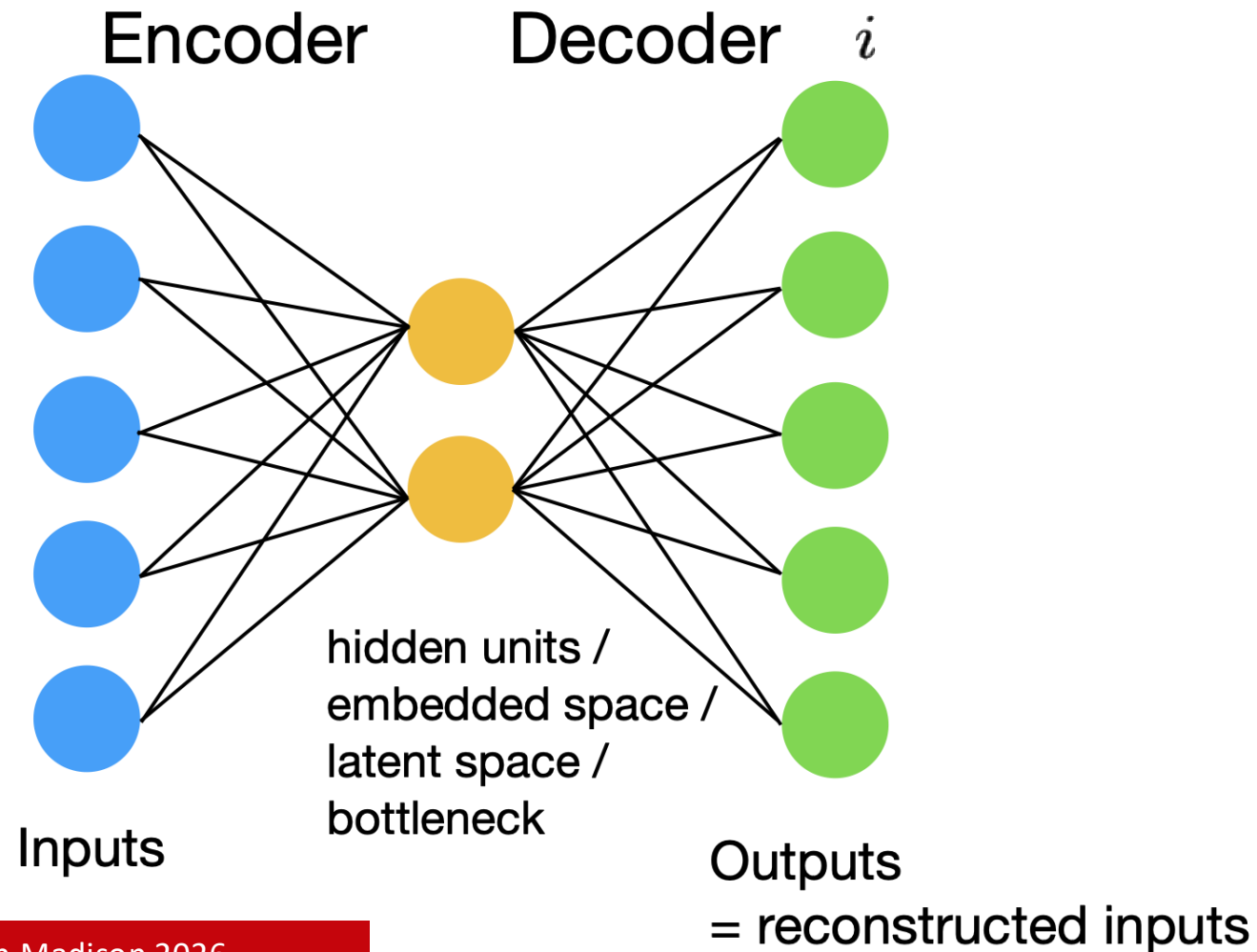
Last Time: Deep Generative Models

- Goal: Generative models of the form $P(X, Y, \theta)$ without strong simplifying assumptions.
- Define probabilistic distributions over a set of variables
- “Deep” means multiple layers of hidden variables!
- **Hidden structure z that explains high-dim. x**
- Fundamental challenge: We never observe z
- This makes two core computations difficult:
 - Marginal likelihood:** $p_{\theta}(x) = \int p_{\theta}(x, z) dz$
 - Posterior inference:** $p_{\theta}(z | x) \propto p_{\theta}(x | z)p(z)$
- Each type of DGM makes a tradeoff



Last Time: A Basic Fully-Connected Autoencoder

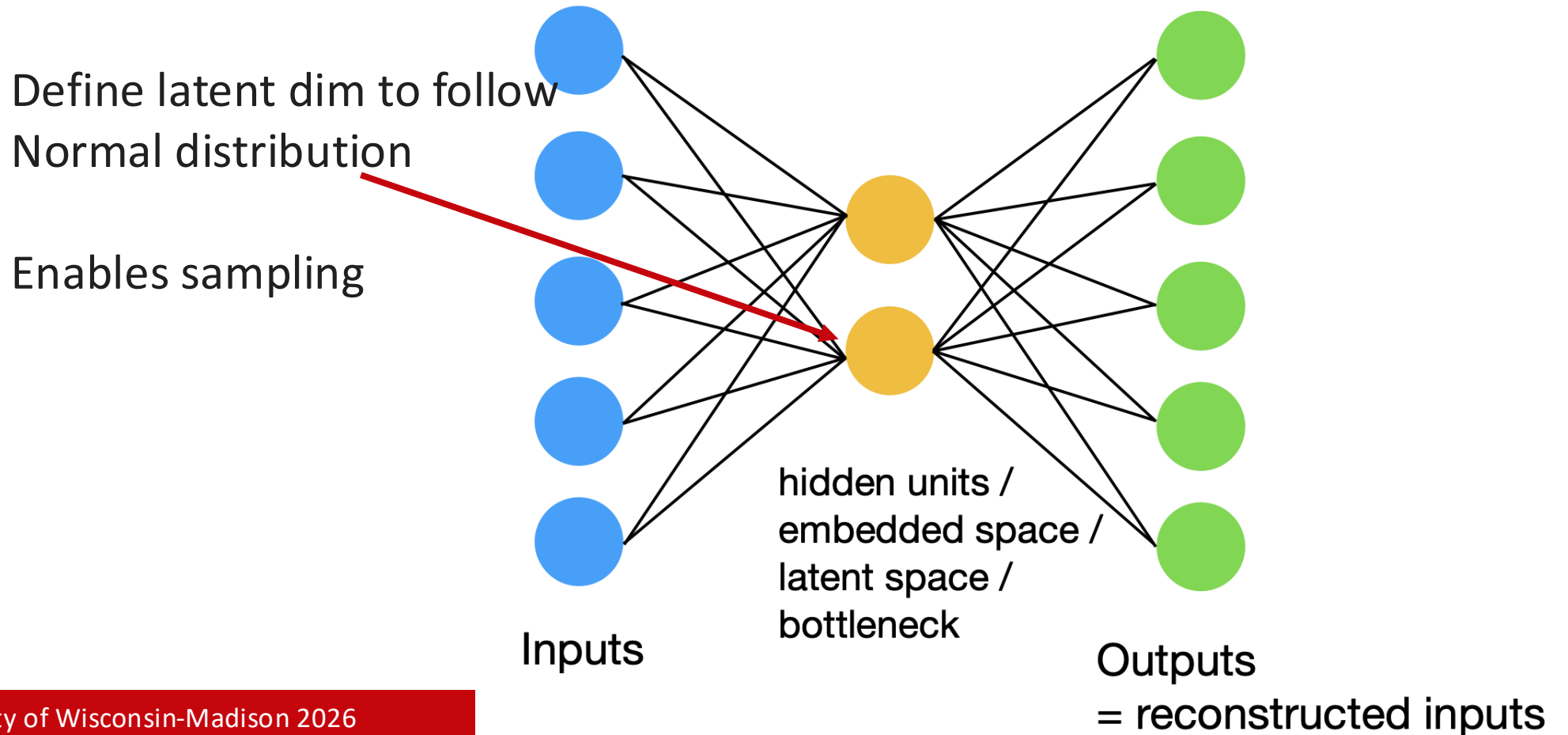
$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2^2 = \sum_i (x_i - x'_i)^2$$



Today: Variational Autoencoders

Kullback-Leibler divergence term
where $p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1)$

$$L^{[i]} = -\mathbb{E}_{z \sim q_w(z|x^{[i]})} [\log p_w(x^{[i]}|z)] + \text{KL}(q_w(z|x^{[i]}) || p(z))$$



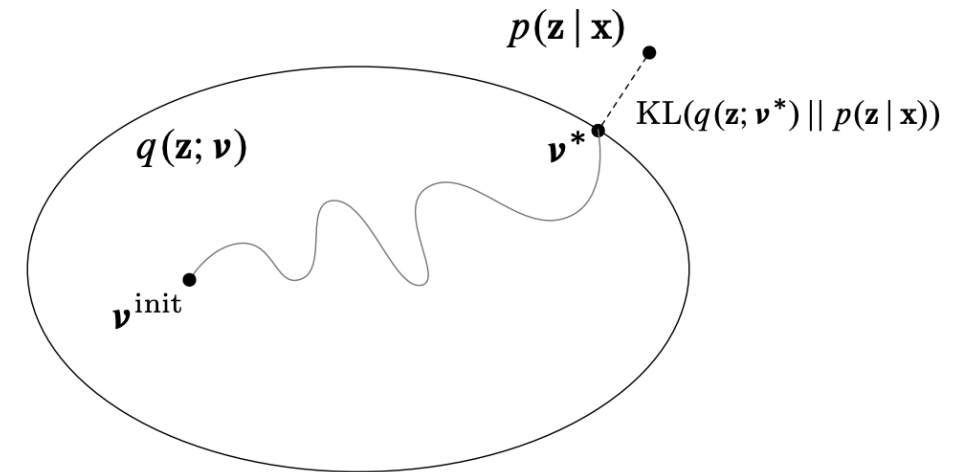
Variational Autoencoders (VAEs)

Variational Inference

- The main idea behind variational methods is to pick a family of distributions with its own **variational parameters**,

$$q(z_{1:m} \mid \nu)$$

- Then, find ν that makes q close to the posterior of interest.
- Use q with the fitted parameters as a proxy for the posterior
 - e.g., to form predictions about future data or to investigate the posterior distribution of the hidden variables



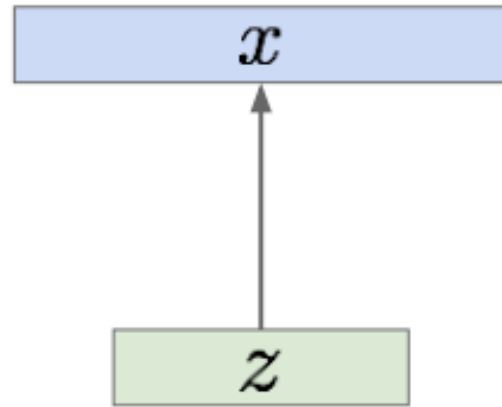
Variational Autoencoders

Sample from
true conditional

$$p_{\theta^*}(x \mid z^{(i)})$$

Sample from
true prior

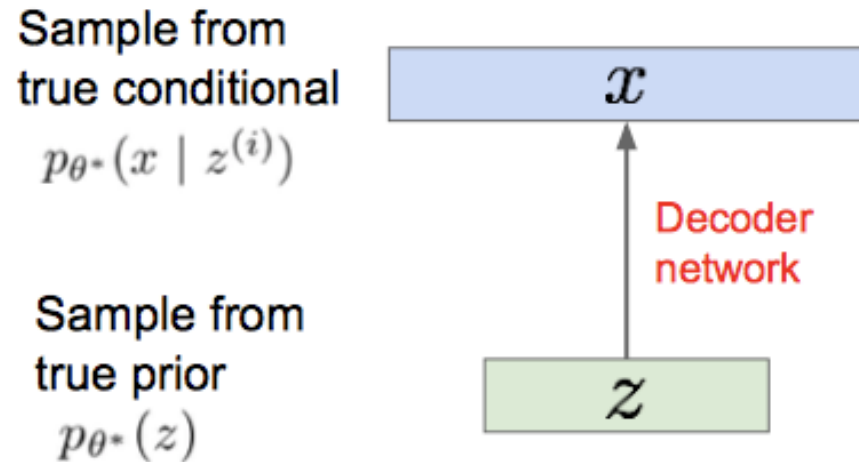
$$p_{\theta^*}(z)$$



We want to estimate the true parameters θ^* of this generative model.

How should we represent this model?

Variational Autoencoders



We want to estimate the true parameters θ^* of this generative model.

How should we represent this model?

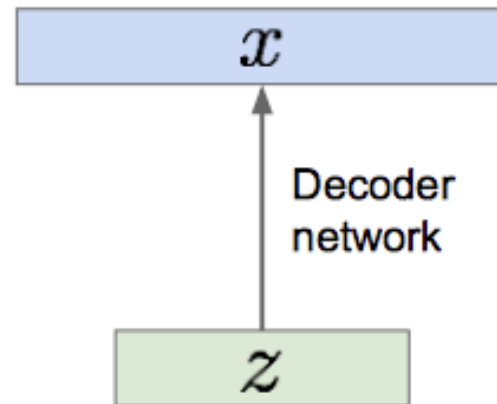
Choose prior $p(z)$ to be simple, e.g. Gaussian.

Conditional $p(x|z)$ is complex (generates image) => represent with neural network

Variational Autoencoders

Sample from
true conditional
 $p_{\theta^*}(x \mid z^{(i)})$

Sample from
true prior
 $p_{\theta^*}(z)$



We want to estimate the true parameters θ^* of this generative model.

How to train the model?

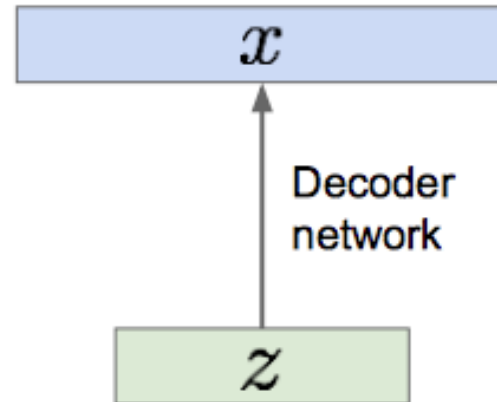
Variational Autoencoders

Sample from
true conditional

$$p_{\theta^*}(x | z^{(i)})$$

Sample from
true prior

$$p_{\theta^*}(z)$$



We want to estimate the true parameters θ^* of this generative model.

How to train the model?

maximize likelihood of training data

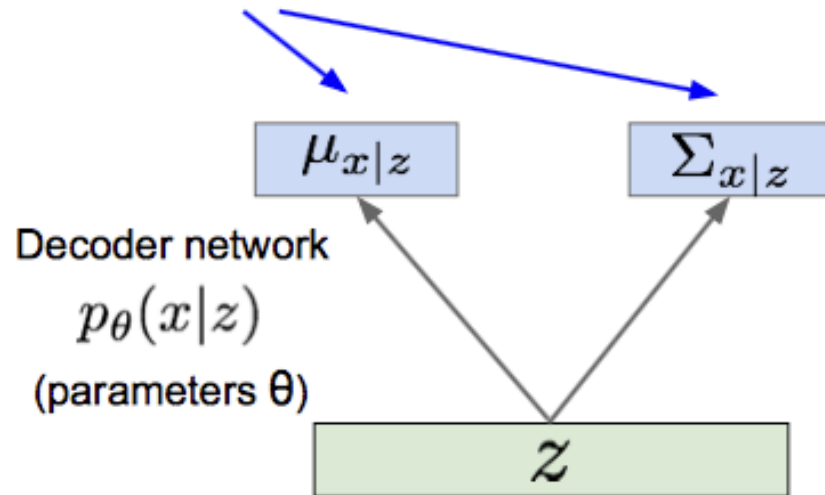
$$p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz$$

Now with latent z

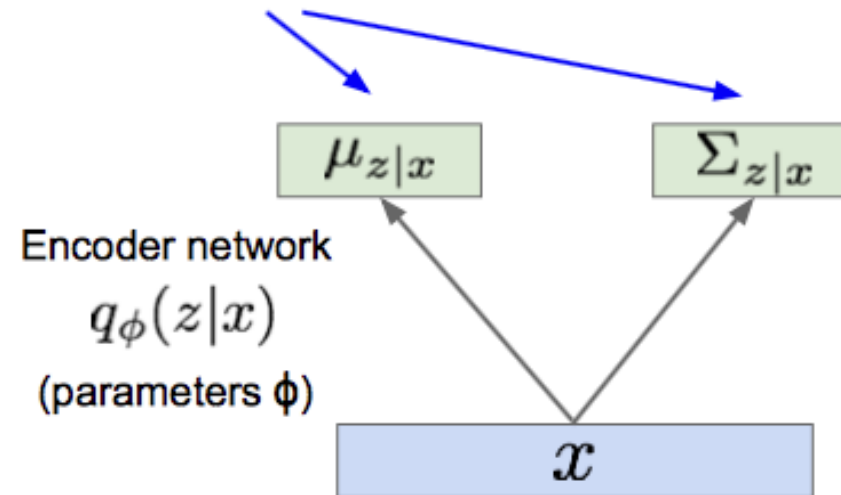
Variational Autoencoders

- Probabilistic **decoder** and **encoder** networks

Mean and (diagonal) covariance of $\mathbf{x} | \mathbf{z}$



Mean and (diagonal) covariance of $\mathbf{z} | \mathbf{x}$



Variational Autoencoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\log p_{\theta}(x^{(i)}) = \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} [\log p_{\theta}(x^{(i)})] \quad (p_{\theta}(x^{(i)}) \text{ Does not depend on } z)$$

↑
encoder network

Variational Autoencoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\begin{aligned}
 \log p_{\theta}(x^{(i)}) &= \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} [\log p_{\theta}(x^{(i)})] && (p_{\theta}(x^{(i)}) \text{ Does not depend on } z) \\
 &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] && (\text{Bayes' Rule}) \\
 &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \right] && (\text{Multiply by constant}) \\
 &= \mathbf{E}_z [\log p_{\theta}(x^{(i)} | z)] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] && (\text{Logarithms}) \\
 &= \mathbf{E}_z [\log p_{\theta}(x^{(i)} | z)] - D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z)) + D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z | x^{(i)}))
 \end{aligned}$$

“encoder”

decoder

our choice of
variational
family

our choice
of prior

our choice of
variational
family

not up to us ☹️

Can approximate this
term with sampling.

Can calculate this term exactly
(if we choose nicely).

Hmm...

Variational Autoencoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\begin{aligned}
 \log p_{\theta}(x^{(i)}) &= \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] \quad (p_{\theta}(x^{(i)}) \text{ Does not depend on } z) \\
 &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Bayes' Rule}) \\
 &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \right] \quad (\text{Multiply by constant}) \\
 &= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Logarithms}) \\
 &= \underbrace{\mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z | x^{(i)}))}_{\geq 0}
 \end{aligned}$$

Tractable lower bound which we can take
gradient of and optimize! ($p_{\theta}(x|z)$ differentiable,
KL term differentiable)

Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z \left[\log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z \left[\log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

Let's look at computing the bound (forward pass) for a given minibatch of input data

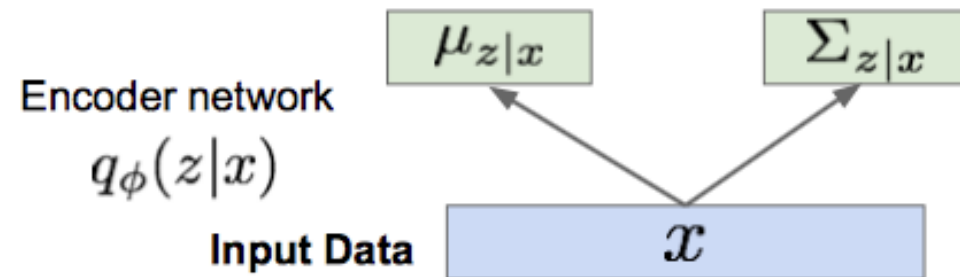
Input Data

x

Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

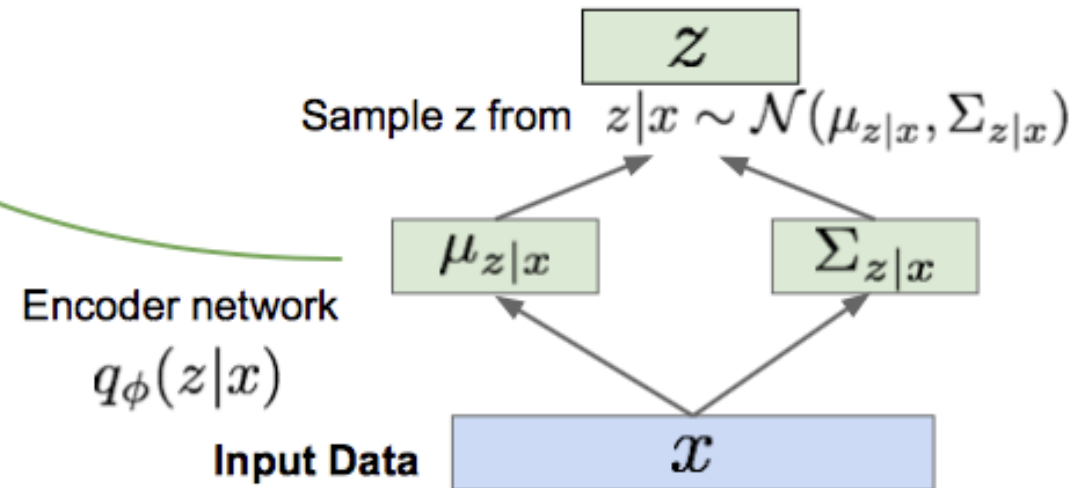


Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z \left[\log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

Make approximate posterior distribution close to prior

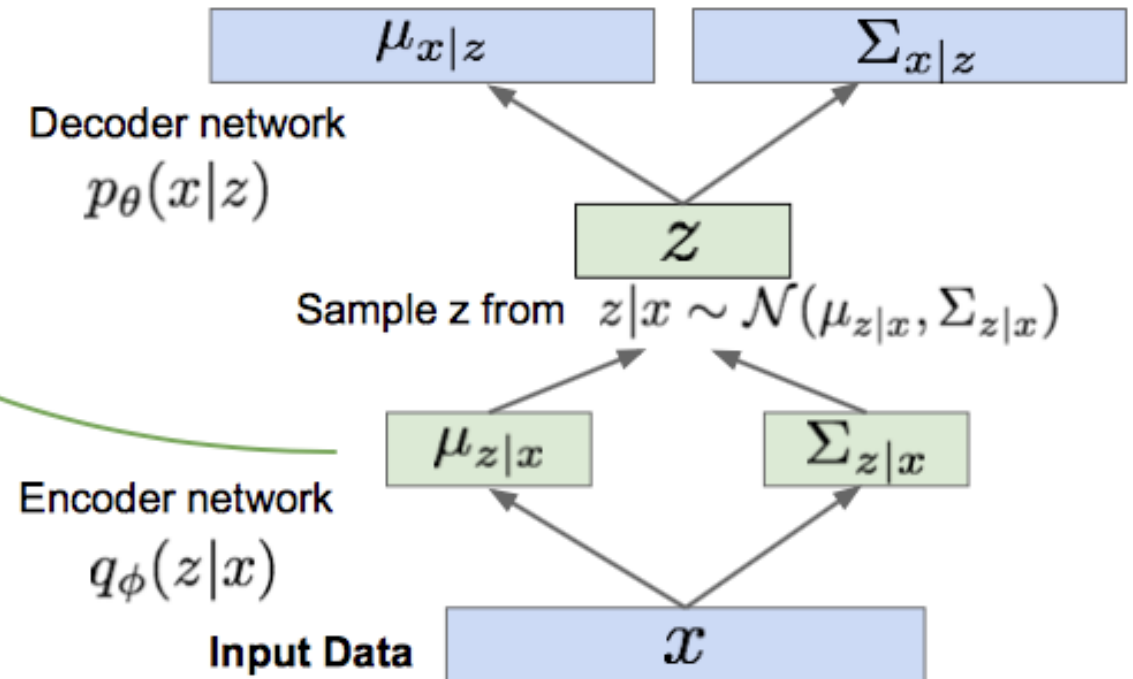


Variational Autoencoders

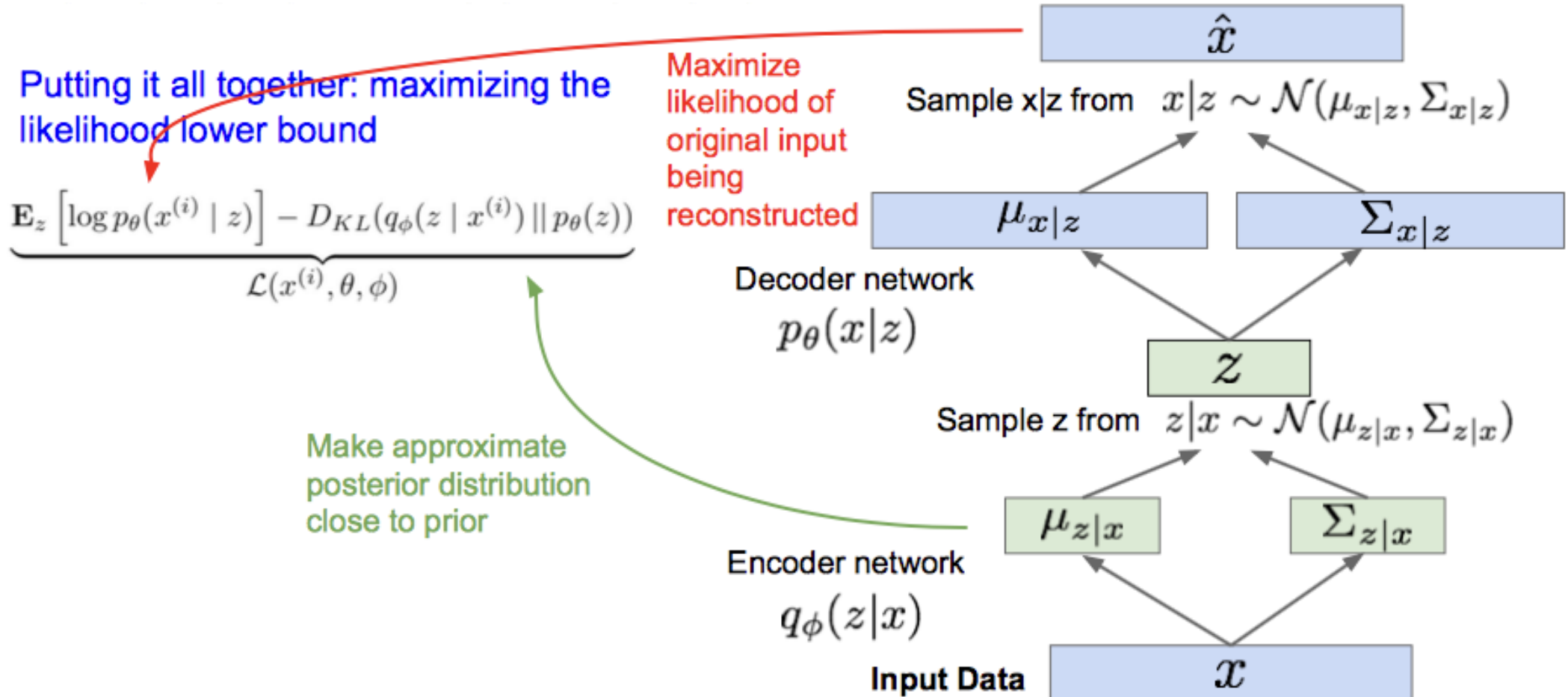
Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbb{E}_z \left[\log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

Make approximate posterior distribution close to prior

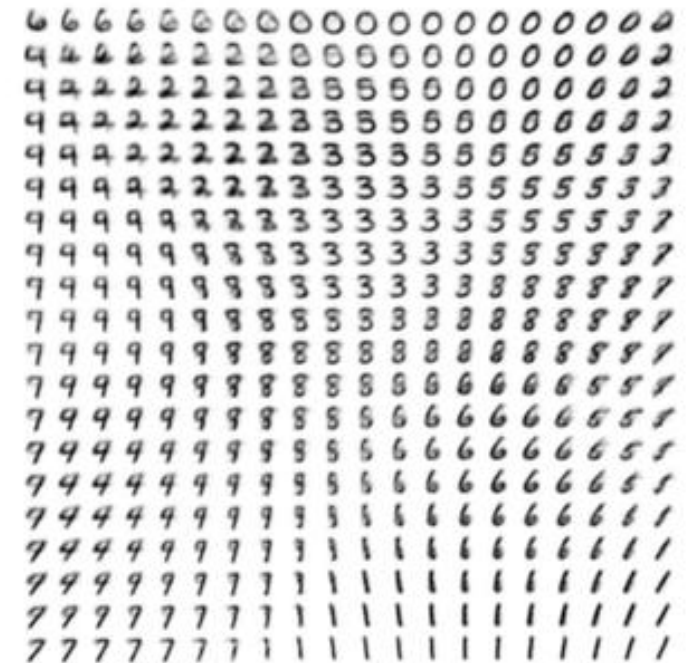
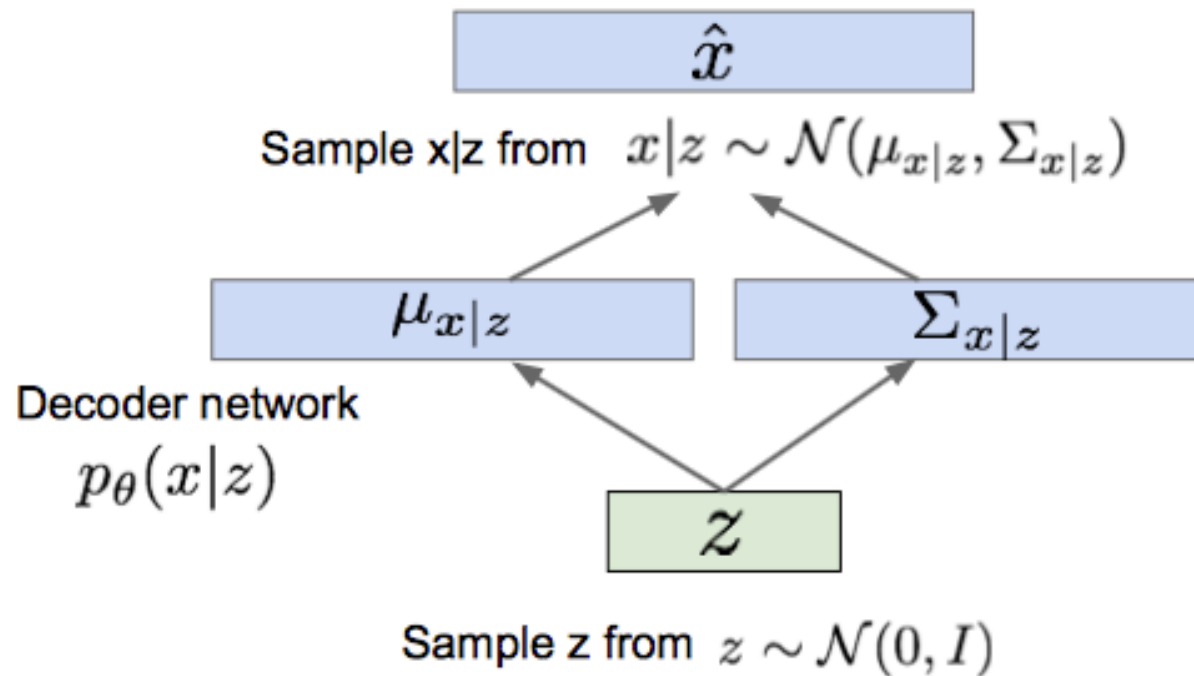


Variational Autoencoders



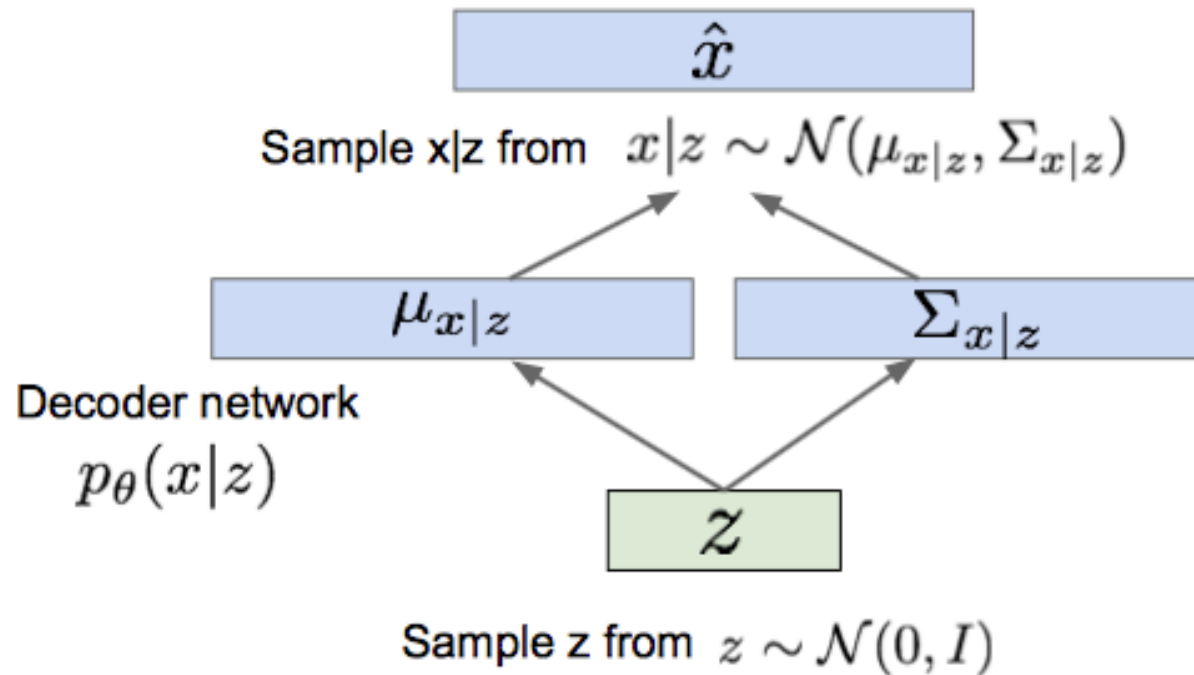
Variational Autoencoders: Generating

Use decoder network. Now sample z from prior!



Variational Autoencoders: Generating

Use decoder network. Now sample z from prior!



- VAEs tend to generate **blurred** images due to the mode covering behavior (more later)



Celebrity faces [Radford 2015]

Generative Adversarial Networks (GANs)





Generative Adversarial Nets (GANs)

arXiv.org > stat > arXiv:1406.2661

Search...

Help | Advanced S

Statistics > Machine Learning

[Submitted on 10 Jun 2014]

Generative Adversarial Networks

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio

We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G . The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D , a unique solution exists, with G recovering the training data distribution and D equal to $1/2$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

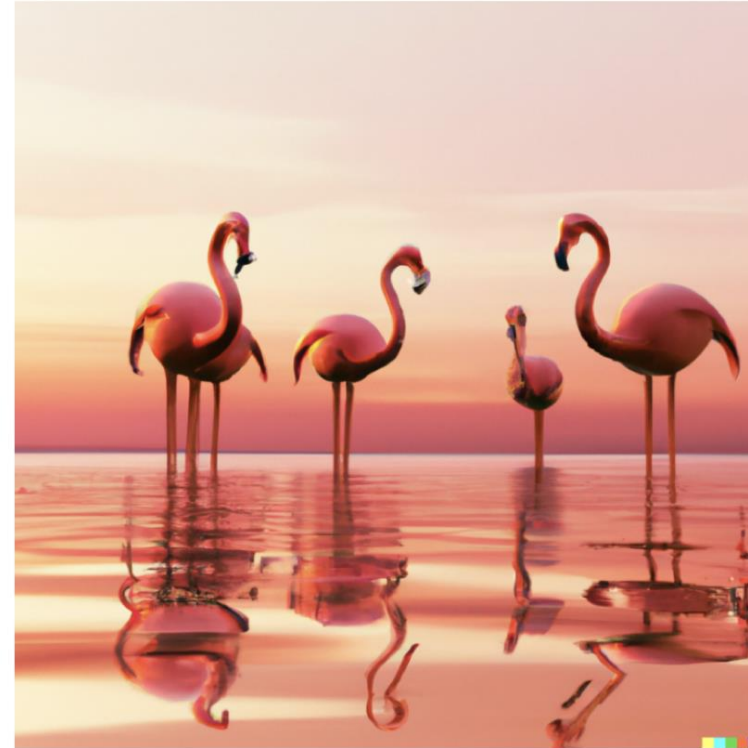
<https://arxiv.org/abs/1406.2661>

Lots of GAN Applications

Text-based image generation via diffusion models



"People walking on a beach during sunrise, a reflection of the sun on the water, realistic"



"Flamingos standing on water, red sunset, pink-red water reflection, photo-realistic, 4k"

Dall-E 2 by OpenAI, April 2022

Lots of GAN Applications



"A photo of a Shiba Inu dog with a backpack riding a bike. It is wearing sunglasses and a beach hat."



"A blue jay standing on a large basket of rainbow macarons"



"A brain riding a rocketship heading towards the moon"

Imagen by Google, April 2022

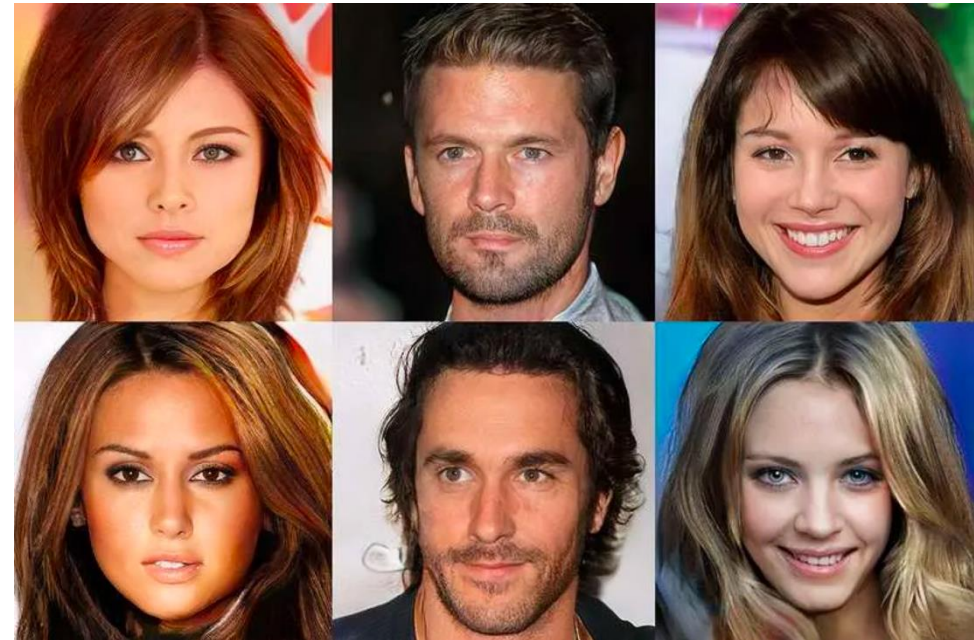
Lots of GAN Applications

Human Faces generated by **VAEs**



Celebrity faces [Radford 2015]

Human Faces generated by **GANs**

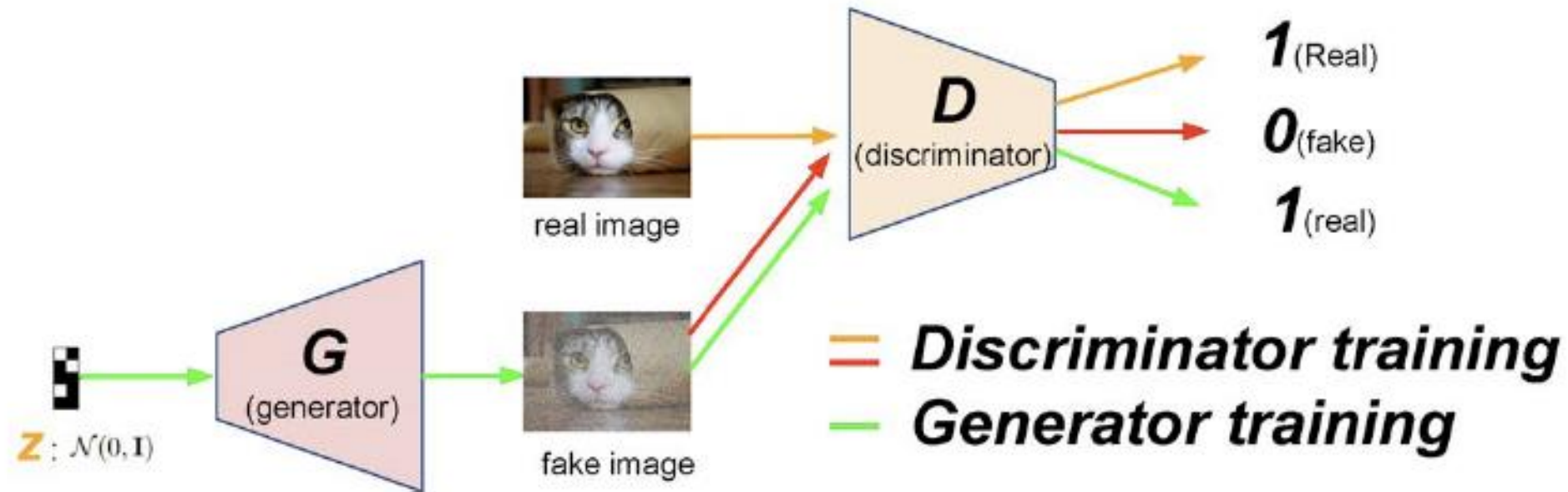


<https://becominghuman.ai/generative-adversarial-networks-gans-human-creativity-2fc61283f3f6>

Generative Adversarial Nets (GANs)

- The original purpose is to generate new data
- Classically for generating new images, but applicable to wide range of domains
- Learns the training set distribution and can generate new images that have never been seen before
- Similar to VAE, and in contrast to e.g., autoregressive models or RNNs (generating one word at a time), GANs generate the whole output all at once

GAN Training



Discriminator: $\max_D \mathcal{L}_D = \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x} \sim G(\mathbf{z}), \mathbf{z} \sim p(\mathbf{z})} [\log(1 - D(\mathbf{x}))]$

Generator: $\min_G \mathcal{L}_G = \mathbb{E}_{\mathbf{x} \sim G(\mathbf{z}), \mathbf{z} \sim p(\mathbf{z})} [\log(1 - D(\mathbf{x}))]$.



GAN Training – An Adversarial Game

Discriminator: learns to become better at distinguishing real from generated images

Generator: learns to generate better images to fool the discriminator

GAN Training – Putting it together

Discriminator: $\max_D \mathcal{L}_D = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x} \sim G(\mathbf{z}), \mathbf{z} \sim p(\mathbf{z})} [\log(1 - D(\mathbf{x}))]$

Generator: $\min_G \mathcal{L}_G = \mathbb{E}_{\mathbf{x} \sim G(\mathbf{z}), \mathbf{z} \sim p(\mathbf{z})} [\log(1 - D(\mathbf{x}))]$.



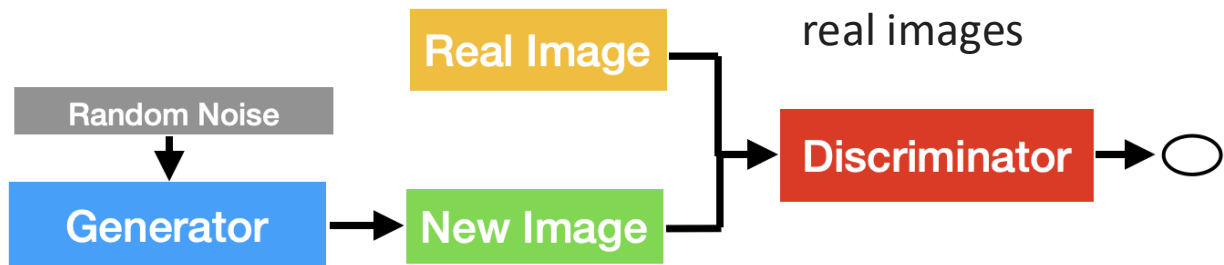
$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

GAN Training – Putting it together

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

Discriminator gradient for update (gradient ascent):

$$\nabla_{\mathbf{w}_D} \frac{1}{n} \sum_{i=1}^n \left[\underbrace{\log D(\mathbf{x}^{(i)})}_{\text{We want large probability on real images}} + \underbrace{\log(1 - D(G(\mathbf{z}^{(i)})))}_{\text{We want small probability on generated images}} \right]$$



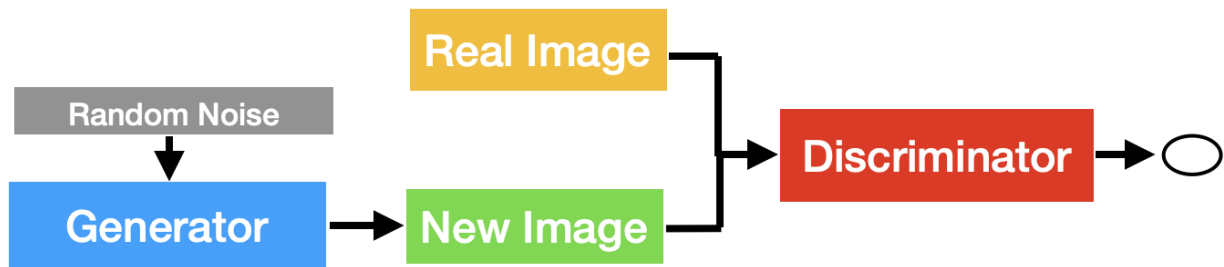
GAN Training – Putting it together

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

Generator gradient for update (gradient descent):

$$\nabla_{\mathbf{w}_G} \frac{1}{n} \sum_{i=1}^n \log \left(1 - D \left(G \left(\mathbf{z}^{(i)} \right) \right) \right)$$

Want discriminator to predict poorly on fake images



Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k , is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

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

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

end for

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

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

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets." In Advances in Neural Information Processing Systems, pp. 2672-2680. 2014.

GAN Training – Convergence?

- Converges when Nash-equilibrium (Game Theory concept) is reached in the minmax (zero-sum) game

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

- Nash-equilibrium in Game Theory is reached when the actions of one player won't change depending on the opponent's actions
- Here, this means that the GAN produces realistic images and the discriminator outputs random predictions (probabilities close to 0.5)

GAN Training Problems

- Oscillation between generator and discriminator loss
- Mode collapse (generator produces examples of a particular kind only)
- Discriminator is too strong, such that the gradient for the generator vanishes and the generator can't keep up

Instead of gradient descent with

$$\nabla_{\mathbf{w}_G} \frac{1}{n} \sum_{i=1}^n \log \left(1 - D \left(G \left(\mathbf{z}^{(i)} \right) \right) \right)$$

Do gradient ascent with

$$\nabla_{\mathbf{w}_G} \frac{1}{n} \sum_{i=1}^n \log \left(D \left(G \left(\mathbf{z}^{(i)} \right) \right) \right)$$

“Non-saturating” GAN

GAN Training Problems

- Oscillation between generator and discriminator loss
- Mode collapse (generator produces examples of a particular kind only)
- Discriminator is too strong, such that the gradient for the generator vanishes and the generator can't keep up
- Discriminator is too weak, and the generator produces non-realistic images that fool it too easily (rare problem, though)
- Sensitive to learning rate and other hyper parameters



GAN Training Problems

- Lots of Tips & Tricks:

<https://github.com/soumith/ganhacks>

Deep Convolutional GAN

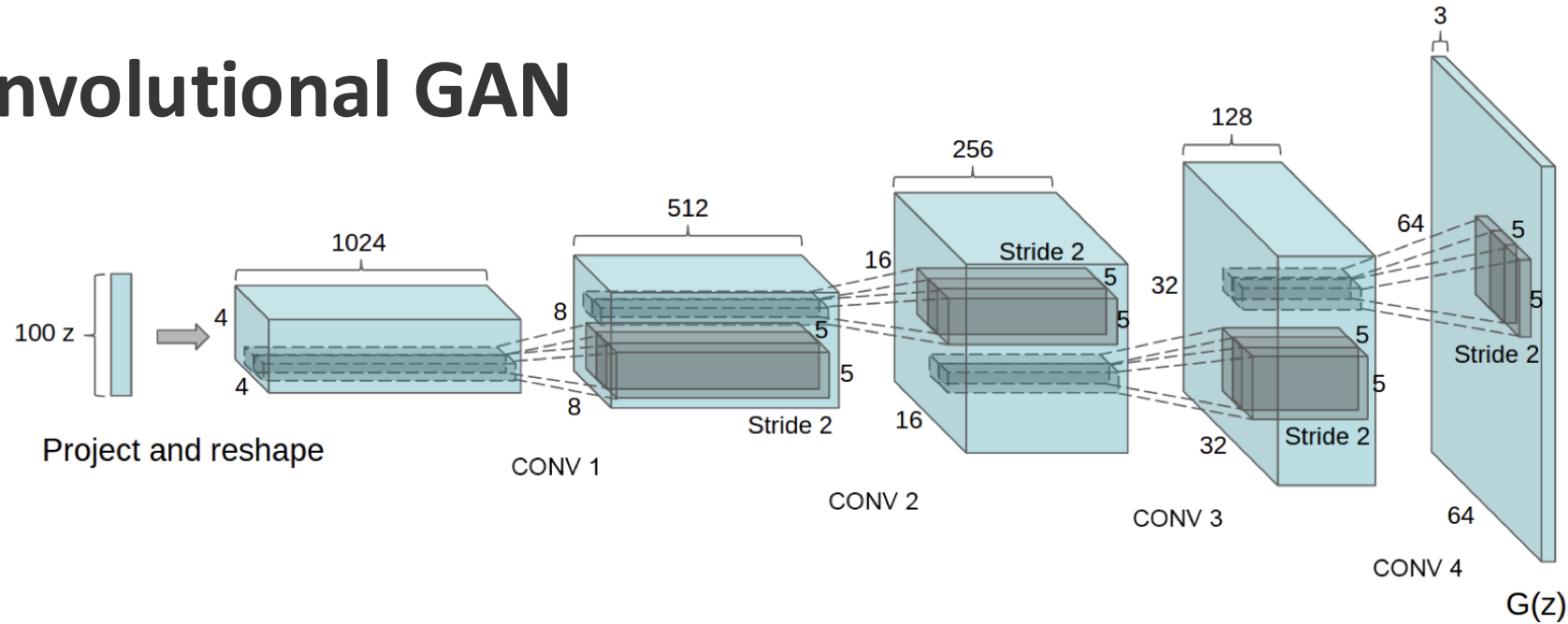


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64×64 pixel image. Notably, no fully connected or pooling layers are used.

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.

Bonus

GANs and VAEs: A Unified View

GANs



- GAN: Implicit distribution over $x \sim p_\theta(x | y)$

$$p_\theta(\mathbf{x}|y) = \begin{cases} p_{g_\theta}(\mathbf{x}) & y = 0 \\ p_{data}(\mathbf{x}) & y = 1. \end{cases} \quad \begin{array}{l} \text{(distribution of generated images)} \\ \text{(distribution of real images)} \end{array}$$

- $x \sim p_{g_\theta}(x) \Leftrightarrow x = G_\theta(z), z \sim p(z | y = 0)$
- $x \sim p_{data}(x)$

GANs: Rewrite in Variational-EM format

- The familiar “Variational-EM” format:

$$\max_{\phi} \mathcal{L}_{\phi} = \mathbb{E}_{\mathbf{x}=G_{\theta}(\mathbf{z}), \mathbf{z} \sim p(\mathbf{z}|\mathbf{y}=0)} [\log(1 - D_{\phi}(\mathbf{x}))] + \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} [\log D_{\phi}(\mathbf{x})]$$

$$\begin{aligned} \max_{\theta} \mathcal{L}_{\theta} &= \mathbb{E}_{\mathbf{x}=G_{\theta}(\mathbf{z}), \mathbf{z} \sim p(\mathbf{z}|\mathbf{y}=0)} [\log D_{\phi}(\mathbf{x})] + \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} [\log(1 - D_{\phi}(\mathbf{x}))] \\ &= \mathbb{E}_{\mathbf{x}=G_{\theta}(\mathbf{z}), \mathbf{z} \sim p(\mathbf{z}|\mathbf{y}=0)} [\log D_{\phi}(\mathbf{x})] \end{aligned}$$

- Implicit distribution over $\mathbf{x} \sim p_{\theta}(\mathbf{x} | \mathbf{y})$:

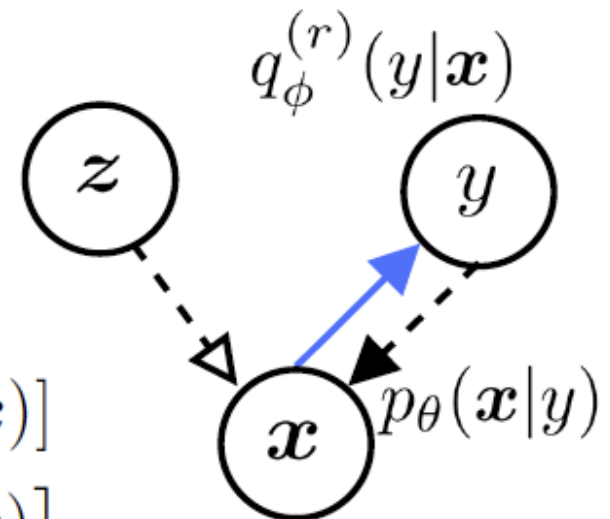
$$\mathbf{x} = G_{\theta}(\mathbf{z}), \mathbf{z} \sim p(\mathbf{z} | \mathbf{y} = 0)$$

- Discriminator distribution $q_{\phi}(\mathbf{y} | \mathbf{x})$:

$$q_{\phi}^r(\mathbf{y} | \mathbf{x}) = q_{\phi}(1 - \mathbf{y} | \mathbf{x})$$

$$\max_{\phi} \mathcal{L}_{\phi} = \mathbb{E}_{p_{\theta}(\mathbf{x}|\mathbf{y})p(\mathbf{y})} [\log q_{\phi}(\mathbf{y}|\mathbf{x})]$$

$$\max_{\theta} \mathcal{L}_{\theta} = \mathbb{E}_{p_{\theta}(\mathbf{x}|\mathbf{y})p(\mathbf{y})} [\log q_{\phi}^r(\mathbf{y}|\mathbf{x})]$$



Variational EM vs. GANs

- Variational EM

- Objectives

$$\max_{\phi} \mathcal{L}_{\phi, \theta} = \mathbb{E}_{q_{\phi}(z|x)} [\log p_{\theta}(x|z)] + KL(q_{\phi}(z|x) || p(z))$$

$$\max_{\theta} \mathcal{L}_{\phi, \theta} = \mathbb{E}_{q_{\phi}(z|x)} [\log p_{\theta}(x|z)] + KL(q_{\phi}(z|x) || p(z))$$

- Single objective for both θ and ϕ
 - Extra prior regularization by $p(z)$

- The **reconstruction term**:

- Maximize the conditional log-likelihood of x with the generative distribution $p_{\theta}(x | z)$ conditioning on the latent code z inferred by $q_{\phi}(z | x)$

- $p_{\theta}(x | z)$ is the generative model
- $q_{\phi}(z | x)$ is the inference model

- GAN

- Objectives

$$\max_{\phi} \mathcal{L}_{\phi} = \mathbb{E}_{p_{\theta}(x|y)p(y)} [\log q_{\phi}(y|x)]$$

$$\max_{\theta} \mathcal{L}_{\theta} = \mathbb{E}_{p_{\theta}(x|y)p(y)} [\log q_{\phi}^r(y|x)]$$

- Two objectives

- Maximize the conditional log-likelihood of y with the distribution $q_{\phi}(y | x)$ conditioning on data/generation x from $p_{\theta}(x | y)$
- $q_{\phi}(y | x)$ is the generative model
- $p_{\theta}(x | y)$ is the inference model

GANs vs VAEs: A Symmetry

Hu et al. "[Unifying Deep Generative Models](#)"

	GANs (InfoGAN)	VAEs
Generative distribution	$p_{\theta}(\mathbf{x} y) = \begin{cases} p_{g_{\theta}}(\mathbf{x}) & y = 0 \\ p_{data}(\mathbf{x}) & y = 1. \end{cases}$	$p_{\theta}(\mathbf{x} \mathbf{z}, y) = \begin{cases} p_{\theta}(\mathbf{x} \mathbf{z}) & y = 0 \\ p_{data}(\mathbf{x}) & y = 1. \end{cases}$
Discriminator distribution	$q_{\phi}(y \mathbf{x})$	$q_{*}(y \mathbf{x})$, perfect, degenerated
$\mathbf{z}$ -inference model	$q_{\eta}(\mathbf{z} \mathbf{x}, y)$ of InfoGAN	$q_{\eta}(\mathbf{z} \mathbf{x}, y)$
KLD to minimize	$\min_{\theta} \text{KL} (p_{\theta}(\mathbf{x} y) q^r(\mathbf{x} \mathbf{z}, y))$ $\sim \min_{\theta} \text{KL}(P_{\theta} Q)$	$\min_{\theta} \text{KL} \left(q_{\eta}(\mathbf{z} \mathbf{x}, y) q_{*}^r(y \mathbf{x}) p_{\theta}(\mathbf{z}, y \mathbf{x}) \right)$ $\sim \min_{\theta} \text{KL}(Q P_{\theta})$

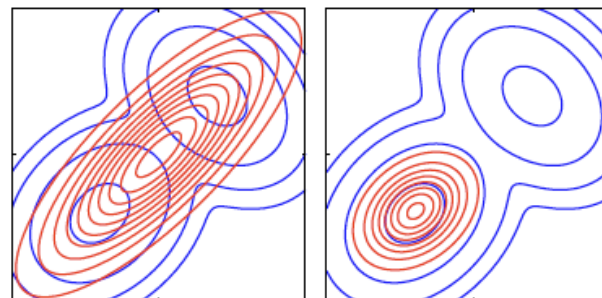


GANs vs VAEs: A Symmetry

Hu et al. "[Unifying Deep Generative Models](#)"

	GANs (InfoGAN)	VAEs
KLD to minimize	$\min_{\theta} \text{KL}(p_{\theta}(\mathbf{x} \mathbf{y}) q^r(\mathbf{x} \mathbf{z}, \mathbf{y}))$ $\sim \min_{\theta} \text{KL}(P_{\theta} Q)$	$\min_{\theta} \text{KL}(q_{\eta}(\mathbf{z} \mathbf{x}, \mathbf{y})q_{*}^r(\mathbf{y} \mathbf{x}) p_{\theta}(\mathbf{z}, \mathbf{y} \mathbf{x}))$ $\sim \min_{\theta} \text{KL}(Q P_{\theta})$

- Asymmetry of KLDs inspires combination of GANs and VAEs
 - GANs: $\min_{\theta} \text{KL}(P_{\theta} || Q)$ tends to missing mode
 - VAEs: $\min_{\theta} \text{KL}(Q || P_{\theta})$ tends to cover regions with small values of p_{data}



Mode covering

Mode missing

GANs and Mode Collapse

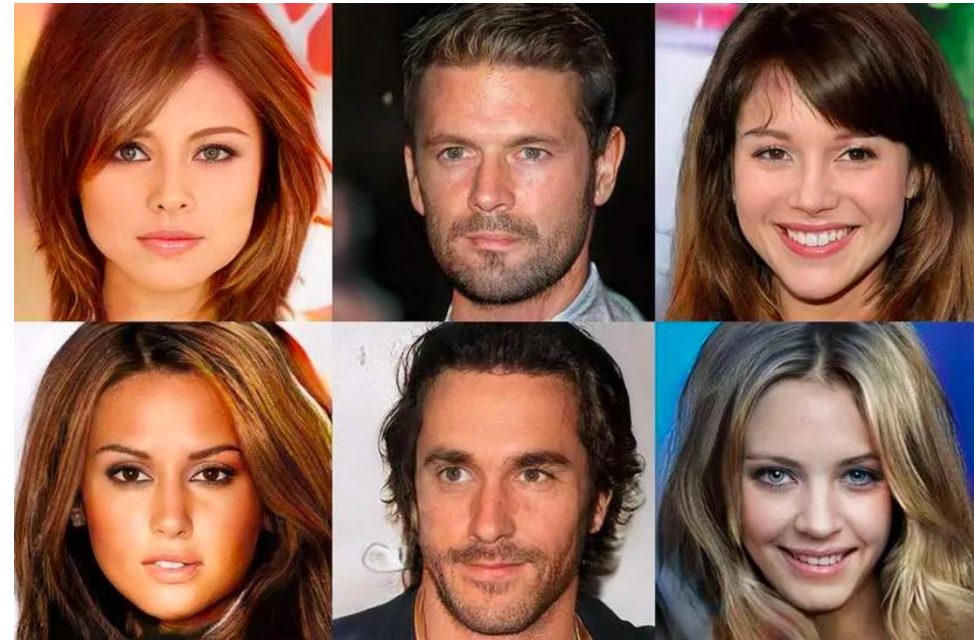


Human Faces generated by **VAEs**



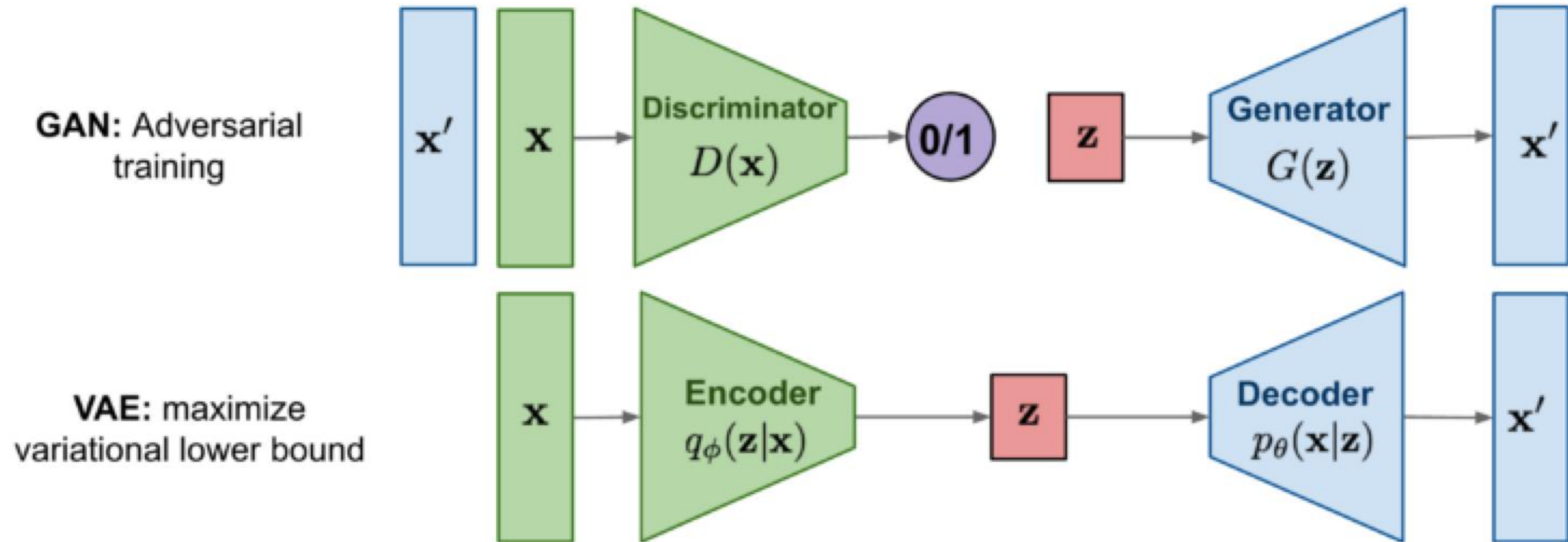
Celebrity faces [Radford 2015]

Human Faces generated by **GANs**



<https://becominghuman.ai/generative-adversarial-networks-gans-human-creativity-2fc61283f3f6>

What to remember



Source: <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

Questions?

