Generative Models:
PixelCNN / PixelRNN
Variational AutoEncoders (VAEs)
Administrative

- Milestone Report is due EOD 11/7 NO GRACE PERIOD
- HW3 due EOD 10/24 (grace period ends EOD 10/26)
- HW4 release 10/26, due 11/14
Recap: Computer Vision Tasks

- **Classification**
  - CAT
  - No spatial extent

- **Semantic Segmentation**
  - GRASS, CAT, TREE, SKY
  - No objects, just pixels

- **Object Detection**
  - DOG, DOG, CAT
  - Multiple Object

- **Instance Segmentation**
  - DOG, DOG, CAT
  This image is CC0 public domain
Semantic Segmentation
Object Detection: Multiple Objects

Each image needs a different number of outputs!

CAT: \((x, y, w, h)\) 4 numbers

DOG: \((x, y, w, h)\)
DOG: \((x, y, w, h)\) 12 numbers

CAT: \((x, y, w, h)\)

DUCK: \((x, y, w, h)\) Many numbers!

DUCK: \((x, y, w, h)\)

....
Region Proposals: Selective Search

- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU

Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014
Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014
“Slow” R-CNN

Predict “corrections” to the RoI: 4 numbers: (dx, dy, dw, dh)

Classify regions with SVMs

Forward each region through ConvNet

Warped image regions (224x224 pixels)

Regions of Interest (RoI) from a proposal method (~2k)

Problem: Very slow! Need to do ~2k independent forward passes for each image!

Idea: Pass the image through convnet before cropping! Crop the conv feature instead!

Figure copyright Ross Girshick, 2015; source. Reproduced with permission.
Cropping Features: RoI Align

Sample at regular points in each subregion using bilinear interpolation

No “snapping”!

Max-pool within each subregion

Region features (here 512 x 2 x 2; In practice e.g 512 x 7 x 7)

Cropping Features:
RoI Align

Project proposal onto features

He et al, “Mask R-CNN”, ICCV 2017
Region Proposal Network

Input Image (e.g. 3 x 640 x 480)

CNN

Example: 20 x 15 anchor box uniformly sampled on the feature map

Image features (e.g. 512)

At each point, predict whether the corresponding anchor contains an object (binary classification)

Anchor is an object? 1 x 20 x 15

Conv
**Faster R-CNN:**
Make CNN do proposals!

Faster R-CNN is a **Two-stage object detector**

**First stage:** Run once per image
- Backbone network
- Region proposal network

**Second stage:** Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset

Do we really need the second stage?
Mask R-CNN

He et al, "Mask R-CNN", arXiv 2017
Mask R-CNN: Very Good Results!

He et al, "Mask R-CNN", ICCV 2017
3D Shape Prediction: Mesh R-CNN

Gkioxari et al., Mesh RCNN, ICCV 2019
What if all we have are data without label?

We have lots of raw data (e.g., Internet)!
Can we still learn useful things without labels?
Generative Models
Supervised vs Unsupervised Learning

**Supervised Learning**

**Data:** (x, y)

x is data, y is label

**Goal:** Learn a *function* to map x -> y

**Examples:** Classification, regression, object detection, semantic segmentation, image captioning, etc.
Supervised vs Unsupervised Learning

Supervised Learning

**Data**: (x, y)
x is data, y is label

**Goal**: Learn a *function* to map x -> y

**Examples**: Classification, regression, object detection, semantic segmentation, image captioning, etc.

This image is CC0 public domain
Supervised vs Unsupervised Learning

Supervised Learning

**Data**: (x, y)
x is data, y is label

**Goal**: Learn a *function* to map x -> y

**Examples**: Classification, regression, object detection, semantic segmentation, image captioning, etc.

A cat sitting on a suitcase on the floor

Image captioning

Caption generated using neuraltalk2
Image - CC0 Public domain
Supervised vs Unsupervised Learning

Supervised Learning

**Data:** \((x, y)\)
\(x\) is data, \(y\) is label

**Goal:** Learn a *function* to map \(x \rightarrow y\)

**Examples:** Classification, regression, object detection, semantic segmentation, image captioning, etc.
Supervised vs Unsupervised Learning

Supervised Learning

Data: \((x, y)\)
x is data, y is label

Goal: Learn a \textit{function} to map \(x \rightarrow y\)

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.
Supervised vs Unsupervised Learning

Unsupervised Learning

**Data:** $x$
Just data, **no labels!**

**Goal:** Learn some underlying hidden *structure* of the data

**Examples:** Clustering, dimensionality reduction, density estimation, etc.
Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden structure of the data

Examples: Clustering, dimensionality reduction, density estimation, etc.
Unsupervised Learning

**Data:** $x$
Just data, no labels!

**Goal:** Learn some underlying hidden *structure* of the data

**Examples:** Clustering, dimensionality reduction, density estimation, etc.

Principal Component Analysis (Dimensionality reduction)
Supervised vs Unsupervised Learning

Unsupervised Learning

**Data:** $x$
Just data, no labels!

**Goal:** Learn some underlying hidden *structure* of the data

**Examples:** Clustering, dimensionality reduction, density estimation, etc.

1-d density estimation

2-d density estimation

Modeling $p(x)$
Supervised vs Unsupervised Learning

**Supervised Learning**

**Data:** (x, y)

x is data, y is label

**Goal:** Learn a *function* to map x -> y

**Examples:** Classification, regression, object detection, semantic segmentation, image captioning, etc.

**Unsupervised Learning**

**Data:** x

Just data, no labels!

**Goal:** Learn some underlying hidden *structure* of the data

**Examples:** Clustering, dimensionality reduction, density estimation, etc.
Generative Modeling

Given training data, generate new samples from same distribution

Training data $\sim p_{\text{data}}(x)$

Objectives:
1. Learn $p_{\text{model}}(x)$ that approximates $p_{\text{data}}(x)$
2. Sampling new $x$ from $p_{\text{model}}(x)$
Generative Modeling

Given training data, generate new samples from same distribution

Training data ~ $p_{\text{data}}(x)$

Formulate as density estimation problems:
- **Explicit density estimation**: explicitly define and solve for $p_{\text{model}}(x)$, e.g., a high-dimensional Gaussian Mixture Model (GMM)
- **Implicit density estimation**: learn model that can sample from $p_{\text{model}}(x)$ without explicitly defining it.
Why Generative Models?

- Realistic samples for artwork, super-resolution, colorization, etc.
- Learn useful features for downstream tasks such as classification.
- Getting insights from high-dimensional data (physics, medical imaging, etc.)
- Modeling physical world for simulation and planning (robotics and reinforcement learning applications)
- Many more ...

Figures from L-R are copyright: (1) Alec Radford et al. 2016; (2) Phillip Isola et al. 2017. Reproduced with authors permission (3) BAIR Blog.
Taxonomy of Generative Models

Generative models

Explicit density

Tractable density

Fully Visible Belief Nets
- NADE
- MADE
- PixelRNN/CNN
- NICE / RealNVP
- Glow
- Fjord

Approximate density

Variational

Variational Autoencoder

Denoising Diffusion Models

Implicit density

Approximate density

Markov Chain

Markov Chain

Markov Chain

Direct

GAN

Boltzmann Machine

GSN

Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.
Taxonomy of Generative Models

Today and the next lecture: discuss 4 most popular types of generative models

- Fully Visible Belief Nets
  - NADE
  - MADE
  - PixelRNN/CNN
  - NICE / RealNVP
  - Glow
  - Ffjord

- Variational Autoencoder
- Denoising Diffusion Models

Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.
PixelRNN and PixelCNN
(A very brief overview)
Fully visible belief network (FVBN)

Explicit density model

\[ p(x) = p(x_1, x_2, \ldots, x_n) \]

- Likelihood of image \( x \)
- Joint likelihood of each pixel in the image
Fully visible belief network (FVBN)

Explicit density model

Use chain rule to decompose likelihood of an image $x$ into product of 1-d distributions:

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})$$

- Likelihood of image $x$
- Probability of i’th pixel value given all previous pixels

Then maximize likelihood of training data
Fully visible belief network (FVBN)

Explicit density model

Use chain rule to decompose likelihood of an image $x$ into product of 1-d distributions:

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})$$

E.g. softmax over 0-255

Then maximize likelihood of training data

Complex distribution over pixel values $\Rightarrow$ Express using a neural network!
Recurrent Neural Network

$p(x_i|x_1, \ldots, x_{i-1})$
PixelRNN  [van der Oord et al. 2016]

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)
PixelRNN  [van der Oord et al. 2016]

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)
PixelRNN \cite{van_der_Oord_2016}

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)
PixelRNN [van der Oord et al. 2016]

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)

Drawback: sequential generation is slow in both training and inference!
PixelCNN [van der Oord et al. 2016]

Still generate image pixels starting from corner

Dependency on previous pixels now modeled using a CNN over context region (masked convolution)

Figure copyright van der Oord et al., 2016. Reproduced with permission.
PixelCNN [van der Oord et al. 2016]

Still generate image pixels starting from corner

Dependency on previous pixels now modeled using a CNN over context region (masked convolution)

Training is faster than PixelRNN (can parallelize convolutions since context region values known from training images)

Generation is still slow:
For a 32x32 image, we need to do forward passes of the network 1024 times for a single image
Generation Samples

32x32 CIFAR-10

32x32 ImageNet

Figures copyright Aaron van der Oord et al., 2016. Reproduced with permission.
PixelRNN and PixelCNN

Improving PixelCNN performance
- Gated convolutional layers
- Short-cut connections
- Discretized logistic loss
- Multi-scale
- Training tricks
- Etc…

See
- Van der Oord et al. NIPS 2016
- Salimans et al. 2017
  (PixelCNN++)

Pros:
- Can explicitly compute likelihood p(x)
- Easy to optimize
- Good samples

Con:
- Sequential generation => slow
Taxonomy of Generative Models

- Explicit density
  - Tractable density
    - Fully Visible Belief Nets
      - NADE
      - MADE
      - PixelRNN/CNN
      - NICE / RealNVP
      - Glow
      - Fjord
  - Approximate density
    - Variational
    - Variational Autoencoder
    - Boltzmann Machine
    - Denoising Diffusion Models
- Implicit density
  - Markov Chain
    - GAN
    - GSN
  - Variational Markov Chain
  - Boltzmann Machine
  - Denoising Diffusion Models

Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.
Variational Autoencoders (VAE)
So far...

PixelR/CNNs define tractable density function, optimize likelihood of training data:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, ..., x_{i-1})$$
So far...

PixelR/CNNs define tractable density function, optimize likelihood of training data:

\[
p_{\theta}(x) = \prod_{i=1}^{n} p_{\theta}(x_i | x_1, \ldots, x_{i-1})
\]

Variational Autoencoders (VAEs) define intractable density function with latent \( z \):

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

No dependencies among pixels, can generate all pixels at the same time!
So far...

PixelR/CNNs define tractable density function, optimize likelihood of training data:

\[ p_\theta(x) = \prod_{i=1}^{\mathbf{n}} p_\theta(x_i|x_1, \ldots, x_{i-1}) \]

Variational Autoencoders (VAEs) define intractable density function with latent \( z \):

\[ p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz \]

No dependencies among pixels, can generate all pixels at the same time!
Latent variable \( z \) that captures important factors of variations in dataset

Cannot optimize (maximum likelihood estimation) directly, derive and optimize lower bound on likelihood instead
So far...

PixelR/CNNs define tractable density function, optimize likelihood of training data:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i|x_1, \ldots, x_{i-1})$$

Variational Autoencoders (VAEs) define intractable density function with latent $z$:

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

No dependencies among pixels, can generate all pixels at the same time!

Latin variable $z$ that captures important factors of variations in dataset

Cannot optimize (maximum likelihood estimation) directly, derive and optimize lower bound on likelihood instead
Some background first: Autoencoders

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data
Some background first: Autoencoders

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

\( z \) usually smaller than \( x \) (dimensionality reduction)

Q: Why dimensionality reduction?
Some background first: Autoencoders

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

$\mathbf{z}$ usually smaller than $\mathbf{x}$
(dimensionality reduction)

Q: Why dimensionality reduction?

A: Want features to capture the “essence” of the dataset. Think of compression.
Some background first: Autoencoders

How to learn this feature representation?

Train such that features can be used to reconstruct original data “Autoencoding” - encoding input itself

**Encoder**: 4-layer conv

**Decoder**: 4-layer upconv

Input data

Features

Reconstructed input data

Reconstructed data

Encoder: 4-layer conv
Decoder: 4-layer upconv

Input data
Some background first: Autoencoders

Train such that features can be used to reconstruct original data

L2 Loss function:

$$\|x - \hat{x}\|^2$$

Doesn’t use labels!

Encoder: 4-layer conv
Decoder: 4-layer upconv

Input data

Features

Decoder

Reconstructed data
Some background first: Autoencoders

After training, throw away decoder
Some background first: Autoencoders

Transfer from large, unlabeled dataset to small, labeled dataset.

Encoder can be used to initialize a supervised model.

Loss function (Softmax, etc)

Predicted Label

Classifier

Features

Encoder

Input data

Fine-tune encoder jointly with classifier

Train for final task (sometimes with small data)

bird  plane  
dog  deer  truck
Some background first: Autoencoders

Autoencoders can reconstruct data, and can learn features to initialize a supervised model.

Features capture factors of variation in training data.

Ideally, knowing the space of $Z$ is sufficient to recover the *entire training set* through the decoder.
Some background first: Autoencoders

Autoencoders can reconstruct data, and can learn features to initialize a supervised model.

Features capture factors of variation in training data.

Ideally, knowing the space of $Z$ is sufficient to recover the entire training set through the decoder.

VAE: Model data distribution $p(x)$ through a probabilistic latent space $p(z)$ and a probabilistic decoder $p(x|z)$.

$$p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$$
Variational Autoencoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!
Variational Autoencoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

Assume training data \( \{x^{(i)}\}_{i=1}^{N} \) is generated from the distribution of unobserved (latent) representation \( z \)

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

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

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

Assume training data \( \{x^{(i)}\}_{i=1}^{N} \) is generated from the distribution of unobserved (latent) representation \( z \).

**Intuition** (remember from autoencoders!): \( x \) is an image, \( z \) is latent code used to generate \( x \).

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

Sample from true prior
\[
z^{(i)} \sim p_{\theta^*}(z)
\]

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders

We want to estimate the true parameters $\theta^*$ of this generative model given training data $x$. $\theta^*$ includes both the decoder model parameters and the latent distribution.

Sample from true conditional

$\textbf{p}_{\theta^*}(x \mid z^{(i)})$

Sample from true prior

$z^{(i)} \sim \textbf{p}_{\theta^*}(z)$

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders

We want to estimate the true parameters $\theta^*$ of this generative model given training data $x$.

How should we represent this model?

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders

We want to estimate the true parameters $\theta^*$ of this generative model given training data $x$.

How should we represent this model?

Assume $p(z)$ is known and simple, e.g. isotropic Gaussian. Reasonable for latent attributes, e.g. pose, how much smile.

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders

We want to estimate the true parameters $\theta^*$ of this generative model given training data $x$.

How should we represent this model?

Assume $p(z)$ is known and simple, e.g. isotropic Gaussian. Reasonable for latent attributes, e.g. pose, how much smile.

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

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders

Sample from true conditional

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

Sample from true prior

\[ z^{(i)} \sim p_{\theta^*}(z) \]

Decoder network

We want to estimate the true parameters \( \theta^* \) of this generative model given training data \( x \).

How to train the model?

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders

We want to estimate the true parameters $\theta^*$ of this generative model given training data $x$.

How to train the model?

Learn model parameters to maximize likelihood of training data

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

Sample from true conditional

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

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

Sample from true prior

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders

We want to estimate the true parameters $\theta^*$ of this generative model given training data $x$.

How to train the model?

Learn model parameters to maximize likelihood of training data

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

Q: What is the problem with this?

Intractable!

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders: Intractability

Data likelihood: \( p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz \)

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders: Intractability

Data likelihood: \( p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz \)

Simple Gaussian prior

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders: Intractability

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

Decoder neural network

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders: Intractability

Data likelihood: $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$

Intractable to compute $p(x|z)$ for every $z$!
Variational Autoencoders: Intractability 😞 ✔ ✔

Data likelihood: $p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$

Intractable to compute $p(x|z)$ for every $z$!
Variational Autoencoders: Intractability

😢 ✔ ✔

Data likelihood: \( p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz \)

Can we do Monte Carlo sampling?

\[
\log p(x) \approx \log \frac{1}{k} \sum_{i=1}^{k} p(x|z^{(i)}), \text{ where } z^{(i)} \sim p(z)
\]
Variational Autoencoders: Intractability

Data likelihood: $p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$

Can we do Monte Carlo sampling?

$$\log p(x) \approx \log \frac{1}{k} \sum_{i=1}^{k} p(x|z^{(i)}), \text{ where } z^{(i)} \sim p(z)$$

We don’t know which $z$ corresponds to a sample $(x)$!
Most $z$’s will be sampled from where $p(x|z)$ is zero.

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders: Intractability 😞 ✓ ✓

Data likelihood: \( p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz \)

Can we do Monte Carlo sampling?

\[ \log p(x) \approx \log \frac{1}{k} \sum_{i=1}^{k} p(x|z^{(i)}), \text{ where } z^{(i)} \sim p(z) \]

Can we estimate posterior density?

\[ p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)} \]

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders: Intractability

Data likelihood:  \( p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz \)

Can we do Monte Carlo sampling?

\[ \log p(x) \approx \log \frac{1}{k} \sum_{i=1}^{k} p(x|z^{(i)}), \text{ where } z^{(i)} \sim p(z) \]

Can we estimate posterior density? Not quite, but …

\[ p_\theta(z|x) = p_\theta(x|z)p_\theta(z)/p_\theta(x) \]

Intractable data likelihood

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders: Intractability

Data likelihood: \[ p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz \]

Can we do Monte Carlo sampling?
\[ \log p(x) \approx \log \frac{1}{k} \sum_{i=1}^{k} p(x|z^{(i)}), \text{ where } z^{(i)} \sim p(z) \]

Can we estimate posterior density? Not quite, but …
\[ p_\theta(z|x) = p_\theta(x|z)p_\theta(z)/p_\theta(x) \]

VAE: We can use an approximate posterior (variational distribution) to form a *tractable lower bound* of the data likelihood \( p(x) \).

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders

$$
\log p_\theta(x^{(i)}) = \mathbf{E}_{z \sim q_\phi(z|x^{(i)})} \left[ \log p_\theta(x^{(i)}) \right] \\
(p_\theta(x^{(i)}) \text{ Does not depend on } z)
$$
Let's assume we can sample from some approximate posterior for now …

\[ \log p_{\theta}(x^{(i)}) = E_{z \sim q_{\phi}(z|x^{(i)})} \left[ \log p_{\theta}(x^{(i)}) \right] \quad (p_{\theta}(x^{(i)}) \text{ Does not depend on } z) \]
\[
\log p_\theta(x^{(i)}) = \mathbb{E}_{z \sim q_\phi(z | x^{(i)})} \left[ \log p_\theta(x^{(i)}) \right] \\
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \\
\text{\quad (Bayes’ Rule)} \quad P(B) = \frac{P(B | A)P(A)}{P(A | B)}
\]
Variational Autoencoders

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

\[
= \mathbf{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \quad \text{(Bayes’ Rule)}
\]

\[
= \mathbf{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z) q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)}) q_\phi(z | x^{(i)})} \right] \quad \text{(Multiply by constant)}
\]
Variational Autoencoders

\[
\log p_\theta(x^{(i)}) = \mathbb{E}_{z \sim q_\phi(z \mid x^{(i)})} \left[ \log p_\theta(x^{(i)}) \right] \\
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})} \right] \\
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})} \frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})} \right] \\
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} \mid z) \right] - \mathbb{E}_z \left[ \log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)} \right] + \mathbb{E}_z \left[ \log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})} \right] 
\]
Variational Autoencoders

\[
\log p_\theta(x^{(i)}) = E_{z \sim q_\phi(z|x^{(i)})} \left[ \log p_\theta(x^{(i)}) \right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)
\]

\[
= E_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \quad \text{(Bayes’ Rule)}
\]

\[
= E_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{q_\phi(z | x^{(i)})} \frac{q_\phi(z | x^{(i)})}{q_\phi(z | x^{(i)})} \right] \quad \text{(Multiply by constant)}
\]

\[
= E_z \left[ \log p_\theta(x^{(i)} | z) \right] - E_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z)} \right] + E_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] \quad \text{(Logarithms)}
\]

\[
= E_z \left[ \log p_\theta(x^{(i)} | z) \right] - \text{KL}(q_\phi(z | x^{(i)}) \parallel p_\theta(z)) + \text{KL}(q_\phi(z | x^{(i)}) \parallel p_\theta(z | x^{(i)}))
\]

Recall: \( \text{KL}(q \parallel p) = E_q[\log \frac{q}{p}] \)
Variational Autoencoders

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

\[ \quad = \mathbb{E}_z \left[ \log \left( \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})} \right) \right] \quad \text{(Bayes’ Rule)} \]

\[ \quad = \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})} \frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})} \right] \quad \text{(Multiply by constant)} \]

\[ \quad = \mathbb{E}_z \left[ \log p_\theta(x^{(i)} \mid z) \right] - \mathbb{E}_z \left[ \log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)} \right] + \mathbb{E}_z \left[ \log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})} \right] \quad \text{(Logarithms)} \]

\[ \quad = \mathbb{E}_z \left[ \log p_\theta(x^{(i)} \mid z) \right] - D_{KL}(q_\phi(z \mid x^{(i)}) \| p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \| p_\theta(z \mid x^{(i)})) \]

\[ p_\theta(z|x) \text{ intractable (saw earlier), can’t compute this KL term :( But we know KL divergence always } \geq 0. \]
Variational Autoencoders

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

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \quad (\text{Bayes’ Rule})
\]

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{q_\phi(z | x^{(i)})} \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] \quad (\text{Multiply by constant})
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - \mathbb{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z)} \right] + \mathbb{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] \quad (\text{Logarithms})
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z)) + D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z | x^{(i)}))
\]

\[
\mathcal{L}(x^{(i)}, \theta, \phi) \geq 0\]

ELBO: Evidence Lower Bound
Variational inference: Optimize q(z|x) to approximate log[p(x)] by raising ELBO. Higher ELBO -> lower KL(q(z|x)||p(z|x))
Variational Autoencoders

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

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)}|z)p_\theta(z)}{p_\theta(z|x^{(i)})} \right] \quad \text{(Bayes’ Rule)}
\]

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)}|z)p_\theta(z)}{p_\theta(z|x^{(i)})} \frac{q_\phi(z|x^{(i)})}{q_\phi(z|x^{(i)})} \right] \quad \text{(Multiply by constant)}
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)}|z) \right] - \mathbb{E}_z \left[ \log \frac{q_\phi(z|x^{(i)})}{p_\theta(z)} \right] + \mathbb{E}_z \left[ \log \frac{q_\phi(z|x^{(i)})}{p_\theta(z|x^{(i)})} \right] \quad \text{(Logarithms)}
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)}|z) \right] - D_{KL}(q_\phi(z|x^{(i)}) || p_\theta(z))
\]

Minimize KL -> Make the approximate posterior more like the prior!
Use NN to model the approximate posterior.
Variational Autoencoders

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

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \quad \text{(Bayes’ Rule)}
\]

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})q_\phi(z | x^{(i)})} \right] \quad \text{(Multiply by constant)}
\]

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)}{p_\theta(z | x^{(i)})} \right] - \mathbb{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z)} \right] + \mathbb{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] \quad \text{(Logarithms)}
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) \| p_\theta(z))
\]

Decoder network gives \( p_\theta(x|z) \), can compute the expectation by sampling from the learned posterior. (need some trick to differentiate through sampling). This KL term (between Gaussians for encoder and \( z \) prior) has nice closed-form solution!
Variational Autoencoders

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

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \quad \text{(Bayes’ Rule)}
\]

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \frac{q_\phi(z | x^{(i)})}{q_\phi(z | x^{(i)})} \right] \quad \text{(Multiply by constant)}
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - \mathbb{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z)} \right] + \mathbb{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] \quad \text{(Logarithms)}
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))
\]
We want to maximize the data likelihood

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

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \quad \text{(Bayes’ Rule)}
\]

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \quad \text{(Multiply by constant)}
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - \mathbb{E}_z \left[ \log \frac{q_{\phi}(z | x^{(i)})}{p_\theta(z)} \right] + \mathbb{E}_z \left[ \log \frac{q_{\phi}(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] \quad \text{(Logarithms)}
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) \| p_\theta(z))
\]

\[
\mathcal{L}(x^{(i)}, \theta, \phi)
\]

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

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

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})} \right] \quad (\text{Bayes' Rule})
\]

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})} q_\phi(z \mid x^{(i)}) \right] \quad (\text{Multiply by constant})
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} \mid z) \right] - \mathbb{E}_z \left[ \log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)} \right] + \mathbb{E}_z \left[ \log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})} \right] \quad (\text{Logarithms})
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} \mid z) \right] - D_{KL}(q_\phi(z \mid x^{(i)}) \parallel p_\theta(z)) \quad \mathcal{L}(x^{(i)}, \theta, \phi)
\]

**Decoder:**
reconstruct the input data

**Encoder:**
make approximate posterior distribution close to prior

Sample \( z \) from the learned posterior (encoder) to train the decoder to reconstruct!

**Tractable lower bound** which we can take gradient of and optimize! (\( p_\theta(x \mid z) \) differentiable, KL term differentiable)
Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

\[ \mathcal{L}(x^{(i)}, \theta, \phi) \]
Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

\[
E_z \left[ \log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) \| p_\theta(z))
\]

Let’s look at computing the KL divergence between the estimated posterior and the prior given some data

Input Data \( \mathcal{X} \)
Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

\[ \mathbb{E}_z \left[ \log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) \parallel p_\theta(z)) \]

Encoder network

\[ q_\phi(z | x) \]

Input Data

\[ \mathcal{X} \]
Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

\[
\mathbb{E}_z \left[ \log p_{\theta}(x^{(i)} \mid z) \right] - D_{KL}(q_{\phi}(z \mid x^{(i)}) \mid \mid p_{\theta}(z))
\]

\[
\mathcal{L}(x^{(i)}, \theta, \phi)
\]

Make approximate posterior distribution close to prior

\[
D_{KL}(\mathcal{N}(\mu_z \mid x, \Sigma_z \mid x) \mid \mid \mathcal{N}(0, I))
\]

Have analytical solution

Input Data

Encoder network

\[
q_{\phi}(z \mid x)
\]

\[
\mu_z \mid x \\
\Sigma_z \mid x
\]
Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

$$L(x^{(i)}, \theta, \phi) = E_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) \mid \mid p_\theta(z))$$

Not part of the computation graph!

Sample $z$ from $z | x \sim \mathcal{N}(\mu_z | x, \Sigma_z | x)$

Make approximate posterior distribution close to prior

Encoder network $q_\phi(z | x)$

Input Data $x$
Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

Reparameterization trick to make sampling differentiable:

Sample \( \epsilon \sim \mathcal{N}(0, I) \)

\[ z = \mu_{z|x} + \epsilon \sigma_{z|x} \]

Sample \( z \) from \( z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) \)

Encoder network \( q_{\phi}(z|x) \)

Input Data \( X \)
Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

Reparameterization trick to make sampling differentiable:

Sample \( \epsilon \sim \mathcal{N}(0, I) \)

\[
\mathbf{z} = \mu_{z|x} + \epsilon \sigma_{z|x}
\]

Part of computation graph

Sample \( z \) from \( z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) \)

Encoder network

\( q_\phi(z|x) \)

Input Data

Input to the graph
Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

$$E_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))$$

Decoder network
$$p_\theta(x | z)$$

Sample z from
$$z \mid x \sim \mathcal{N}(\mu_z \mid x, \Sigma_z \mid x)$$

Encoder network
$$q_\phi(z \mid x)$$

Input Data
Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

Maximize likelihood of original input being reconstructed

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

Decoder network

\[ p_{\theta}(x | z) \]

Sample z from

\[ z | x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) \]

Encoder network

\[ q_{\phi}(z | x) \]

Input Data

\[ \mathcal{X} \]
Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

\[
\mathbb{E}_z [\log p_\theta(x^{(i)} | z)] - \lambda D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))
\]

Hyperparameter to weigh the strength of the prior matching objective
Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

$$E_z[\log p_\theta(x^{(i)}|z)] - \lambda D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z))$$

For every minibatch of input data: compute this forward pass, and then backprop!

Decoder network

$$p_\theta(x|z)$$

Sample z from

$$z|x \sim \mathcal{N}(\mu_z|x, \Sigma_z|x)$$

Encoder network

$$q_\phi(z|x)$$

Input Data

$$\mathcal{L}(x^{(i)}, \theta, \phi)$$
Variational Autoencoders: Generating Data!

Our assumption about data generation process

Sample from true conditional

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

Sample from true prior

\[ z^{(i)} \sim p_{\theta^*}(z) \]

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders: Generating Data!

Our assumption about data generation process:

1. Sample from true conditional $p_{\theta^*}(x \mid z^{(i)})$
2. Sample from true prior $z^{(i)} \sim p_{\theta^*}(z)$

Now given a trained VAE:

- Use decoder network & sample $z$ from prior!
- Sample $x|z$ from $x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z})$
- Decoder network $p_\theta(x|z)$
- $\Sigma_{x|z}$
- Sample $z$ from $z \sim \mathcal{N}(0, I)$

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders: Generating Data!

Use decoder network. Now sample $z$ from prior!

Sample $x | z$ from $x | z \sim \mathcal{N}(\mu_{x | z}, \Sigma_{x | z})$

Decoder network

$p_{\theta}(x | z)$

Sample $z$ from $z \sim \mathcal{N}(0, I)$

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders: Generating Data!

Use decoder network. Now sample $z$ from prior!

$$\hat{x}$$

Sample $x\mid z$ from $x\mid z \sim \mathcal{N}(\mu_{x\mid z}, \Sigma_{x\mid z})$

Decoder network $p_\theta(x\mid z)$

Sample $z$ from $z \sim \mathcal{N}(0, I)$

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Data manifold for 2-d $z$
Variational Autoencoders: Generating Data!

Diagonal prior on $z$ => independent latent variables

Different dimensions of $z$ encode interpretable factors of variation

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders: Generating Data!

Diagonal prior on $z$
$\Rightarrow$ independent latent variables

Different dimensions of $z$ encode interpretable factors of variation

Also good feature representation that can be computed using $q_\phi(z|x)$!

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders: Generating Data!

32x32 CIFAR-10

Labeled Faces in the Wild

Variational Autoencoders

Probabilistic spin to traditional autoencoders => allows generating data
Defines an intractable density => derive and optimize a (variational) lower bound

Pros:
- Principled approach to generative models
- Latent space z is interpretable and may be useful for other downstream tasks.

Cons:
- Samples are blurry
- KL weights are hard to tune
- Latent distributions are aggressive representation bottlenecks that may limit the expressiveness of the model.

Can be made more powerful by making VAE hierarchical (multiple layers of latents).
Diffusion model (denoising diffusion) can be thought of a type of hierarchical VAE!
Next Time: Denoising Diffusion