Deep Generative Modeling

Ava Soleimany
MIT 6.S191
January 25, 2022
Which face is real?

A  B  C
Supervised vs unsupervised learning

**Supervised Learning**

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

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

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

**Unsupervised Learning**

**Data:** \(x\)
- \(x\) is data, no labels!

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

**Examples:** Clustering, feature or dimensionality reduction, etc.
Supervised vs unsupervised learning

**Supervised Learning**

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

*Goal:* Learn function to map  
$x \rightarrow y$

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

**Unsupervised Learning**

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

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

*Examples:* Clustering, feature or dimensionality reduction, etc.
**Goal:** Take as input training samples from some distribution and learn a model that represents that distribution.

**Density Estimation**

**Sample Generation**

How can we learn $P_{model}(x)$ similar to $P_{data}(x)$?
Why generative models? Debiasing

Capable of uncovering **underlying features** in a dataset

Homogeneous skin color, pose

Diverse skin color, pose, illumination

How can we use this information to create fair and representative datasets?
Why generative models? Outlier detection

- **Problem:** How can we detect when we encounter something new or rare?
- **Strategy:** Leverage generative models, detect outliers in the distribution
- Use outliers during training to improve even more!

95% of Driving Data:
1. sunny, 2. highway, 3. straight road

Detect outliers to avoid unpredictable behavior when training

- Edge Cases
- Harsh Weather
- Pedestrians
Latent variable models

Autoencoders and Variational Autoencoders (VAEs)

Generative Adversarial Networks (GANs)
What is a latent variable?

Myth of the Cave
What is a latent variable?

Can we learn the true explanatory factors, e.g. latent variables, from only observed data?
Autoencoders
Autoencoders: background

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

“Encoder” learns mapping from the data, \( x \), to a low-dimensional latent space, \( z \).

Why do we care about a low-dimensional \( z \)?
Autoencoders: background

How can we learn this latent space?
Train the model to use these features to **reconstruct the original data**

“Decoder” learns mapping back from latent space, $z$, to a reconstructed observation, $\hat{x}$
Autoencoders: background

How can we learn this latent space?
Train the model to use these features to **reconstruct the original data**

\[ L(x, \hat{x}) = \| x - \hat{x} \|^2 \]

Loss function doesn’t use any labels!
Autoencoders: background

How can we learn this latent space? Train the model to use these features to reconstruct the original data.

\[ \mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2 \]

Loss function doesn’t use any labels!
Dimensionality of latent space → reconstruction quality

Autoencoding is a form of compression!
Smaller latent space will force a larger training bottleneck

2D latent space
5D latent space
Ground Truth
Autoencoders for representation learning

**Bottleneck hidden layer** forces network to learn a compressed latent representation

**Reconstruction loss** forces the latent representation to capture (or encode) as much “information” about the data as possible

**Autoencoding** = Automatically encoding data; “Auto” = self-encoding
Variational Autoencoders (VAEs)
Traditional autoencoders
VAEs: key difference with traditional autoencoder
VAEs: key difference with traditional autoencoder

Variational autoencoders are a probabilistic twist on autoencoders! Sample from the mean and standard deviation to compute latent sample
VAE optimization

Encoder computes: $q_\phi(z|x)$

Decoder computes: $p_\theta(x|z)$
VAE optimization

Encoder computes: $q_\phi(z|x)$

Decoder computes: $p_\theta(x|z)$

$L(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(regularization term)}$
VAE optimization

Encoder computes: $q_\phi(z|x)$

Decoder computes: $p_\theta(x|z)$

$L(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(regularization term)}$
VAE optimization

Encoder computes: \( q_{\phi}(z|x) \)

Decoder computes: \( p_{\theta}(x|z) \)

\[ \mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(regularization term)} \]
VAE optimization

Encoder computes: $q_{\phi}(z|x)$

Decoder computes: $p_{\theta}(x|z)$

$L(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(regularization term)}$
Priors on the latent distribution

\[ D \left( q_\phi(z|x) \parallel p(z) \right) \]

Inferred latent distribution \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \ quadratic\ mean\ and\ variance\ distribution.\}

\[ p(z) = N(\mu = 0, \sigma^2 = 1) \]

- Encourages encodings to distribute encodings evenly around the center of the latent space.
- Penalize the network when it tries to “cheat” by clustering points in specific regions (i.e., by memorizing the data).
Priors on the latent distribution

\[ D \left( q_\phi(z|x) \parallel p(z) \right) \]

\[ = - \frac{1}{2} \sum_{j=0}^{k-1} \left( \sigma_j + \mu_j^2 - 1 - \log \sigma_j \right) \]

Common choice of prior – Normal Gaussian:

\[ p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1) \]

- Encourages encodings to distribute encodings evenly around the center of the latent space.
- Penalize the network when it tries to “cheat” by clustering points in specific regions (i.e., by memorizing the data).
Intuition on regularization and the Normal prior

What properties do we want to achieve from regularization? 🤔

1. **Continuity**: points that are close in latent space ➔ similar content after decoding
2. **Completeness**: sampling from latent space ➔ “meaningful” content after decoding

---

Point in latent space not meaningfully decoded

Close in latent space, but not similarly decoded

Not regularized

Points close in latent space, similarly and meaningfully decoded

Regularized
Intuition on regularization and the Normal prior

1. **Continuity**: points that are close in latent space \(\rightarrow\) similar content after decoding
2. **Completeness**: sampling from latent space \(\rightarrow\) “meaningful” content after decoding

Encoding as a distribution does not guarantee these properties!

- Small variances \(\rightarrow\) Pointed distributions
- Different means \(\rightarrow\) Discontinuities
- Not regularized

Normal prior \(\rightarrow\)
continuity + completeness

Center means
Regularize variances

Regularized
Intuition on regularization and the Normal prior

1. **Continuity**: points that are close in latent space $\Rightarrow$ similar content after decoding
2. **Completeness**: sampling from latent space $\Rightarrow$ “meaningful” content after decoding

Regularization with Normal prior helps enforce **information gradient** in the latent space.
VAE computation graph

Encoder computes: $q_\phi(z|x)$

Decoder computes: $p_\theta(x|z)$

$L(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(regularization term)}$
VAE computation graph

**Problem:** We cannot backpropagate gradients through sampling layers!

Encoder computes: $q_\phi(z|x)$
Decoder computes: $p_\theta(x|z)$

$$\mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(regularization term)}$$
Reparameterizing the sampling layer

Key Idea:

\[ z \sim \mathcal{N}(\mu, \sigma^2) \]

Consider the sampled latent vector \( z \) as a sum of

- a fixed \( \mu \) vector;
- and fixed \( \sigma \) vector, scaled by random constants drawn from the prior distribution

\[ z = \mu + \sigma \odot \epsilon \]

where \( \epsilon \sim \mathcal{N}(0,1) \)
Reparametrizing the sampling layer

Deterministic node

Stochastic node

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

Backprop

Original form
Reparameterizing the sampling layer

Deterministic node

Stochastic node

Original form

Reparameterized form

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

\[ z = g(\phi, x, \epsilon) \]

\[ \frac{\partial f}{\partial z} \]

\[ \frac{\partial f}{\partial \phi} \]

\[ \sim \mathcal{N}(0,1) \]
VAEs: Latent perturbation

Slowly increase or decrease a **single latent variable**
Keep all other variables fixed

Different dimensions of \( z \) encodes **different interpretable latent features**
VAEs: Latent perturbation

Ideally, we want latent variables that are uncorrelated with each other.

Enforce diagonal prior on the latent variables to encourage independence.

Disentanglement
Latent space disentanglement with $\beta$-VAEs

Standard VAE loss:

$$
L(\theta, \phi; x, z, \beta) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x) \parallel p(z))
$$

- **Reconstruction term**
- **Regularization term**

$\beta > 1$: constrain latent bottleneck, encourage efficient latent encoding $\rightarrow$ disentanglement

Head rotation (azimuth)

Smile also changing!

Smile relatively constant!

Standard VAE ($\beta = 1$)  \hspace{1cm} $\beta$-VAE ($\beta = 250$)
Why latent variable models? Debiasing

Capable of uncovering **underlying latent variables** in a dataset

Homogeneous skin color, pose VS Diverse skin color, pose, illumination

How can we use latent distributions to create fair and representative datasets?
VAE summary

1. Compress representation of world to something we can use to learn
VAE summary

1. Compress representation of world to something we can use to learn
2. Reconstruction allows for unsupervised learning (no labels!)
VAE summary

1. Compress representation of world to something we can use to learn
2. Reconstruction allows for unsupervised learning (no labels!)
3. Reparameterization trick to train end-to-end
VAE summary

1. Compress representation of world to something we can use to learn
2. Reconstruction allows for unsupervised learning (no labels!)
3. Reparameterization trick to train end-to-end
4. Interpret hidden latent variables using perturbation
1. Compress representation of world to something we can use to learn
2. Reconstruction allows for unsupervised learning (no labels!)
3. Reparameterization trick to train end-to-end
4. Interpret hidden latent variables using perturbation
5. Generating new examples
Generative Adversarial Networks (GANs)
What if we just want to sample?

Idea: don’t explicitly model density, and instead just sample to generate new instances.

Problem: want to sample from complex distribution – can’t do this directly!

Solution: sample from something simple (e.g., noise), learn a transformation to the data distribution.
Generative Adversarial Networks (GANs) are a way to make a generative model by having two neural networks compete with each other.

The **discriminator** tries to identify real data from fakes created by the generator.

The **generator** turns noise into an imitation of the data to try to trick the discriminator.
Intuition behind GANs

**Generator** starts from noise to try to create an imitation of the data.

![Diagram showing the process of generating fake data](image)
Intuition behind GANs

**Discriminator** looks at both real data and fake data created by the generator.
Intuition behind GANs

**Discriminator** looks at both real data and fake data created by the generator:

- **Discriminator**
- **Generator**

- **Real data**
- **Fake data**
Intuition behind GANs

**Discriminator** tries to predict what's real and what's fake.

\[ P(\text{real}) = 1 \]

Real data  
Fake data
Intuition behind GANs

**Discriminator** tries to predict what's real and what's fake.

\[ P(\text{real}) = 1 \]

**Real data**  **Fake data**
Intuition behind GANs

**Discriminator** tries to predict what's real and what's fake.

\[ P(\text{real}) = 1 \]

- Real data
- Fake data

---

*MIT 6.8191 Introduction to Deep Learning*

@MITDeepLearning
Intuition behind GANs

**Discriminator** tries to predict what's real and what's fake.

\[ P(\text{real}) = 1 \]

**Real data**  **Fake data**
Intuition behind GANs

**Generator** tries to improve its imitation of the data.

**Discriminator**

\[ P(real) = 1 \]

- **Real data**
- **Fake data**
**Intuition behind GANs**

**Generator** tries to improve its imitation of the data.

\[ P(\text{real}) = 1 \]

Diagram:
- **Discriminator**
  - Real data
  - Fake data
- **Generator**
  - Real data
  - Fake data
Intuition behind GANs

**Generator** tries to improve its imitation of the data.

**Discriminator**

\[ P(\text{real}) = 1 \]

Real data

Fake data
Intuition behind GANs

**Discriminator** tries to predict what's real and what's fake.

\[ P(\text{real}) = 1 \]

---

MIT 6.S191

6.5191 Introduction to Deep Learning

@introtodeeplearning.com  @MITDeepLearning

1/25/22
Intuition behind GANs

**Discriminator** tries to predict what’s real and what’s fake.

Discriminator

\[ P(real) = 1 \]

Real data  Fake data
Intuition behind GANs

**Discriminator** tries to predict what's real and what's fake.

Discriminator

$$P(\text{real}) = 1$$

Generator

Real data

Fake data
Intuition behind GANs

**Discriminator** tries to predict what's real and what's fake.

\[ P(\text{real}) = 1 \]
Intuition behind GANs

**Generator** tries to improve its imitation of the data.

\[ P(\text{real}) = 1 \]
Intuition behind GANs

**Generator** tries to improve its imitation of the data.

\[ P(\text{real}) = 1 \]
Intuition behind GANs

**Generator** tries to improve its imitation of the data.

\[ P(\text{real}) = 1 \]

- Real data
- Fake data
Intuition behind GANs

**Discriminator** tries to identify real data from fakes created by the generator.

**Generator** tries to create imitations of data to trick the discriminator.

\[ P(\text{real}) = 1 \]

---

**Real data**  **Fake data**
Training GANs

\(G\) tries to synthesize fake instances that fool \(D\)

\(D\) tries to identify the synthesized instances

**Training:** adversarial objectives for \(D\) and \(G\)

**Global optimum:** \(G\) reproduces the true data distribution
Training GANs: loss function

\[
\arg\max_D \mathbb{E}_{z,x} \left[ \log D(G(z)) + \log (1 - D(x)) \right]
\]

- \(D\) tries to identify the synthesized images
- \(X_{\text{real}}\) and \(X_{\text{fake}}\) are the input images for the discriminator

Fake

Real
Training GANs: loss function

\[ \arg \min_G \mathbb{E}_{z, x} \left[ \log D(G(z)) + \log (1 - D(x)) \right] \]
Training GANs: loss function

\[ \arg \min_G \max_D \mathbb{E}_{z, x} \left[ \log D(G(z)) + \log (1 - D(x)) \right] \]

**G** tries to synthesize fake images that fool the best **D**
Generating new data with GANs

After training, use generator network to create **new data** that's never been seen before.
GANs are distribution transformers

Gaussian noise $z \sim N(0,1)$

Trained generator

Learned target data distribution
GANs are distribution transformers

Gaussian noise
\( z \sim N(0,1) \)

Trained generator

Learned target data distribution

\( X \)
GANs are distribution transformers

Gaussian noise
\[ z \sim N(0,1) \]

\( Z \) ------→ \( G \) ------→ \( ? \) ------→ \( X \)

Trained generator

Learned target data distribution

Gans are distribution transformers
GANs: Recent Advances
Progressive growing of GANs
Progressive growing of GANs: results
StyleGAN(2): progressive growing + style transfer

Latent $z \in \mathcal{Z}$

Normalize

Mapping network $f$

FC

FC

FC

FC

FC

FC

FC

FC

FC

FC

w $\in \mathcal{W}$

Synthesis network $g$

Noise

Const 4x4x512

AdaIN

Conv 3x3

B

style

A

B

4x4

AdaIN

Conv 3x3

B

style

A

B

Upsample

Conv 3x3

B

style

A

B

AdaIN

Conv 3x3

B

style

A

B

AdaIN

Conv 3x3

B

style

A

B

8x8

...
GANs for image synthesis: latest results
GANs for image synthesis: latest results
Conditional GANs

What if we want to control the nature of the output, by **conditioning** on a label?
Conditional GANs and pix2pix: paired translation

The discriminator, D, classifies between fake and real pairs. The generator, G, learns to fool the discriminator.
Applications of paired translation

Labels to Street Scene

input

output
Paired translation: results

Map → Aerial View

Aerial View → Map

input  output  input  output
Paired translation: coloring from edges
CycleGAN: domain transformation

CycleGAN learns transformations across domains with unpaired data.

\[ D_X \quad G \quad D_Y \]

\[ X \quad F \quad Y \]
Distribution transformations

**GANs:**
- Gaussian noise
  \( z \sim N(0,1) \)
  
  \[ Z \rightarrow Y \]

**CycleGANs:**
- Gaussian noise \( \rightarrow \) target data manifold

  \[ X \rightarrow Y \]

- data manifold \( X \rightarrow \) data manifold \( Y \)
CycleGAN: transforming speech

Audio waveform (A)

Spectrogram image (A)

Audio waveform (B)

Spectrogram image (B)
Deep Generative Modeling: Summary

Autoencoders and Variational Autoencoders (VAEs)

Learn lower-dimensional latent space and sample to generate input reconstructions

Generative Adversarial Networks (GANs)

Competing generator and discriminator networks
6.S191: Introduction to Deep Learning

Lab 2: Computer Vision

Link to download labs:
http://introtodeeplearning.com/#schedule

1. Open the lab in Google Colab
2. Start executing code blocks and filling in the #TODOs