Lecture 18

Image Synthesis
Image classification

image $x$ \rightarrow \text{Classifier} \rightarrow \text{"Fish"}
Image classification

image $x$  $\rightarrow$ Classifier $\rightarrow$ “Fish”  $\rightarrow$ label $y$
Image classification

image \( x \) \rightarrow \text{Classifier} \rightarrow \text{"Fish"}

label \( y \)
Image classification

: image \( x \) → Classifier → "Duck" label \( y \)
Image synthesis

“Duck” \rightarrow \text{Generator} \rightarrow \text{image } x

label y
Image synthesis

“Fish” \rightarrow \text{Generator} \rightarrow \text{image} \ x

label y
Image translation

User sketch

Translator

Photo
Image translation

Google Map → Translator → Satellite photo
Image translation

BW image → Translator → Color image
Image synthesis via **generative modeling**

\[ P(X|Y = y) \]

X is high-dimensional!

Model of high-dimensional structured data

In vision, this is usually what we are interested in!
What can you do with generative models?

1. Image synthesis

2. Structured prediction

3. Domain mapping

4. (Representation learning)

5. (Model-based intelligence)
1. Image synthesis
2. Structured prediction
3. Domain mapping

[Images: https://ganbreeder.app/]

Image synthesis
Procedural graphics
PROBABALISTIC PLANTS

- roll 5 dice with 4 faces
- possible outcomes: "L", "7716"

- Phase 1: "A" = 6.72
  - "B" = 6.30
  - "C" = 2.35
  - "D" = 3.64
  - "E" = 3.09

- Phase 2: "F" = 4.53
  - "G" = 0.04
Image synthesis from “noise”

\[ z \sim p(z) \]

\[ x = G(z) \]

\[ G : \mathcal{Z} \rightarrow \mathcal{X} \]

\[ z \sim p(z) \]

\[ x = G(z) \]
Learning a generative model

- **Learner**
  - Objective
  - Hypothesis space
  - Optimizer

- **Input samples** → **Latent variables** → **Generated samples**

Learning a density model

Data → Learner → Density

- Objective
- Hypothesis space
- Optimizer

Normalized distribution

$p: \mathcal{X} \rightarrow [0, 1]$


Normalized distribution (some models output unnormalized energy functions)
Max likelihood objective
\[
\max_{\theta} \mathbb{E}_{x \sim p_{\text{data}}} [\log p_\theta(x)]
\]

Considering only Gaussian fits
\[
p_\theta(x) = \mathcal{N}(x; \mu, \sigma)
\]
\[
\theta = [\mu, \sigma]
\]

Closed form optimum:
\[
\mu = \frac{1}{N} \sum_{i=1}^{N} x^{(i)} \quad \sigma^2 = \frac{1}{N} \sum_{i=1}^{N} (x^{(i)} - \mu)^2
\]
Case study #1: Fitting a Gaussian to data

Data → Learner

Objective
\[
\max_{\theta} \mathbb{E}_{x \sim p_{\text{data}}} [\log p_{\theta}(x)]
\]

Hypothesis space
\[
p(x) = \mathcal{N}(x; \mu, \sigma)
\]

Optimizer
\[
\mu = \frac{1}{N} \sum_{i=1}^{N} x^{(i)} \quad \sigma^2 = \frac{1}{N} \sum_{i=1}^{N} (x^{(i)} - \mu)^2
\]

“max likelihood”

Density
\[
p : \mathcal{X} \rightarrow [0, 1]
\]
Case study #2: learning a deep generative model

Data → Learner

Objective
Usually max likelihood
Hypothesis space
Deep net
Optimizer
SGD

Density
$p : \mathcal{X} \rightarrow [0, 1]$
Case study #2: learning a deep generative model

Models that provide a sampler but no density are called **implicit generative models**
Deep generative models are distribution transformers

Prior distribution \( p(z) \)

Target distribution \( p(x) \)

\[ G \]
Deep generative models are distribution transformers

Gaussian noise

\[ z \sim \mathcal{N}(\bar{0}, 1) \]

Synthesized image
Deep generative models are distribution transformers

$$z \sim \mathcal{N}(0, 1)$$

Gaussian noise

Synthesized image
Autoencoder $\rightarrow$ Generative model

$x \xrightarrow{} z \sim p(z) \xrightarrow{} \hat{x}$
Variational Autoencoders (VAEs)
[Kingma & Welling, 2014; Rezende, Mohamed, Wierstra 2014]

Prior distribution

\[ p(z) \]

Target distribution

\[ p(x) \]
Mixture of Gaussians

Target distribution

\[ p_\theta(x) = \sum_{i=1}^{k} w_i \mathcal{N}(x; u_i, \Sigma_i) \]

\[ x \sim p_{\text{data}}(x) \]

\[ p_\theta(x) \]
Variational Autoencoders (VAEs)  
[Kingma & Welling, 2014; Rezende, Mohamed, Wierstra 2014]

Prior distribution $p(z)$  
Target distribution $p(G(x))$

Density model:
\[
p_{\theta}(x) = \int p(x|z; \theta)p(z)dz
\]
\[
p(x|z; \theta) \sim \mathcal{N}(x; G_\theta^\mu(x), G_\theta^\sigma(x))
\]

Sampling:
\[
z \sim p(z) \quad \epsilon \sim \mathcal{N}(0, 1)
\]
\[
x = G_\theta^\mu(z) + G_\theta^\sigma(z)\epsilon
\]
Variational Autoencoder (VAE)

Data $\rightarrow$ \textbf{Learner}

Objective

$$\max_{\theta} \mathbb{E}_{x \sim p_{\text{data}}} [\log p_{\theta}(x)]$$

Hypothesis space

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

$$x = G_\theta^\mu(z) + G_\theta^\sigma(z)\varepsilon$$

$$\rightarrow$$

Density

$$p_{\theta} : \mathcal{X} \rightarrow [0, 1]$$

Sampler

$$G_\theta : \mathcal{Z} \rightarrow \mathcal{X}$$
In order to optimize our model, we need to measure the likelihood it assigns to each datapoint $x$.

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

$$= p(x|z^{(1)})p(z^{(1)})dz + p(x|z^{(2)})p(z^{(2)})dz + p(x|z^{(3)})p(z^{(3)})dz + ...$$
In order to optimize our model, we need to measure the likelihood it assigns to each datapoint $x$.

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

$$= \sim 0+$$

$$\sim 0+$$

$$p(x|z^{(3)})p(z^{(3)})dz + \ldots$$
Current model of target distribution

Prior distribution

If only we knew $z^*$, we wouldn't need the integral...

$$p_\theta(x) = \int p(x|z; \theta)p(z)dz \approx p(x|z^*; \theta)p(z^*)$$
Prior distribution

Current model of target distribution

Prior distribution needs to be a distribution (typically set to Gaussian), but here we (incorrectly) treat it as deterministic for simplicity.

Technical note: for the continuous math to actually work out, \( z^* \sim E(x) \) needs to be a distribution (typically set to Gaussian), but here we (incorrectly) treat it as deterministic for simplicity.

If only we knew \( z^* \), we wouldn’t need the integral...

\[
p_\theta(x) = \int p(x|z; \theta)p(z)dz \\
\approx p(x|z^*; \theta)p(z^*)
\]

So, we simply try to predict \( z^* \) for the given \( x \)!

\[
z^* = E(x) \\
\arg \max_{E} p(x|E(x); \theta)p(E(x))
\]
Prior distribution

Current model of target distribution

If only we knew $z^*$, we wouldn’t need the integral...

\[
p_\theta(x) = \int p(x|z; \theta)p(z)dz \\
\approx p(x|z^*; \theta)p(z^*)
\]

So, we simply try to predict $z^*$ for the given $x$!

\[
z^* = E(x)
\]

(assuming unit Gaussian prior, isotropic Gaussian likelihood model)

\[
\arg\min_E \|G(E(x)) - x\|^2_2 + \|E(x)\|^2_2
\]
Autoencoder!

\[
\arg \min_{G,E} \left\| G(E(x)) - x \right\|_2^2 + \left\| E(x) \right\|_2^2
\]
Autoencoder!

$$\arg\min_{G, E} \mathbb{E}_{x, \epsilon}[\|G(E(x + \epsilon)) - x\|_2^2 + \|E(x + \epsilon)\|_2^2]$$
Classical Autoencoder

\[
\arg \min_{G,E} \mathbb{E}_x \left[ \| G(E(x)) - x \|_2^2 \right]
\]
Variational Autoencoder

\[
\arg \min_{G,E} \mathbb{E}_{x, \epsilon} \left[ \| G(E(x + \epsilon)) - x \|^2_2 + \| E(x + \epsilon) \|^2_2 \right]
\]
All of that math was basically just to make $z$ have a Gaussian distribution, so that we sample random images by inputing random Gaussian noise.
Generative Adversarial Networks (GANs)

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

Synthesized image $\mathbf{x}$
$G$ tries to synthesize fake images that fool $D$

$D$ tries to identify the fakes

[Goodfellow et al., 2014]
\[
\text{arg max}_D \mathbb{E}_{z,x} \left[ \log D(G(z)) + \log (1 - D(x)) \right]
\]

[Goodfellow et al., 2014]
\( G \) tries to synthesize fake images that fool \( D \):

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

[Goodfellow et al., 2014]
G tries to synthesize fake images that fool the best D:

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

[Goodfellow et al., 2014]
Training

G tries to synthesize fake images that fool D

D tries to identify the fakes

• Training: iterate between training D and G with backprop.
• Global optimum when G reproduces data distribution.

[Goodfellow et al., 2014]
Samples from BigGAN
[Brock et al. 2018]
Generative Adversarial Network

Data →

Learner

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

Hypothesis space
Deep nets G and D

Optimizer
Alternating SGD on G and D

Critic
\[ D : \mathcal{X} \rightarrow [0, 1] \]

Sampler
\[ G : \mathcal{Z} \rightarrow \mathcal{X} \]
Latent space (Gaussian)

Data space (Natural image manifold)

\[ \mathbf{z} \xrightarrow{} \mathbf{X} \]

[BigGAN, Brock et al. 2018]
Generative models organize the manifold of natural images

latent space

image space
VAEs
Pros: Cheap to sample, good coverage
Cons: Blurry samples (in practice)

GANs
Pros: Cheap to sample, fast to train, require little data
Cons: No likelihoods, bad coverage (mode collapse), finicky to train (minimax)

Other deep generative models:
**Autoregressive models, Normalizing flows, Energy-based models**

[adapted from slide by David Duvenaud]
1. Image synthesis
2. **Structured prediction**
3. Domain mapping

**Structured Prediction**
Data prediction problems ("structured prediction")

Semantic segmentation

Edge detection

Text-to-photo

Future frame prediction

[Long et al. 2015, …]

[Xie et al. 2015, …]

[Reed et al. 2014, …]

[Mathieu et al. 2016, …]

“this small bird has a pink breast and crown…”
Structured prediction

$X$ is high-dimensional

Model *joint* distribution of high-dimensional data: $P(X|Y = y)$

In vision this is usually what we are interested in

Unstructured: $\prod_i p(X_i|Y = y)$
Deep learning in 2012

Use a **hypothesis space** that can model complex structure (e.g., a CNN, nearest-neighbor)
Why deep learning

How do data science techniques scale with amount of data?

[Slide credit: Andrew Ng]
\[
\arg \min_{\mathcal{F}} \mathbb{E}_{x, y}[L(\mathcal{F}(x), y)]
\]

Hypothesis space

Objective function (loss)
Semantic Segmentation

Data

\[
\{ x, y \} \quad \rightarrow \quad \{ x, y \} \quad \rightarrow \quad f
\]

Learner

Objective

\[
f^* = \arg \min_{f \in \mathcal{F}} \sum_{i=1}^{N} H(y_i, \hat{y}_i)
\]

Hypothesis space

Convolutional neural net

Optimizer

Stochastic gradient descent

\[
x \in \mathbb{R}^{H \times W \times 3}
\]

\[
y \in \mathbb{R}^{H \times W \times K}
\]
Data
\[
\begin{align*}
\{x, y\} & \\
\{x, y\} & \\
\vdots & \\
x & \in \mathbb{R}^{H \times W \times 3} \\
y & \in \mathbb{R}^{H \times W \times 3}
\end{align*}
\]

Sat2Map

Learner

Objective
\[
\theta^* = \arg \min_{\theta} \sum_{i=1}^{N} \left( f_\theta(x)_i - y_i \right)^2
\]

Hypothesis space

Convolutional neural net

Optimizer

Stochastic gradient descent

\[
\rightarrow f
\]
Structured prediction

Use an **objective** that can model structure! (e.g., a graphical model, a GAN, etc)
Generator $G$ takes input $x$ and outputs $G(x)$. 

$\begin{array}{c}
\text{X} \\
\downarrow \\
G \\
\uparrow \\
\text{Generator} \\
\rightarrow \\
G(x) \\
\end{array}$
G tries to synthesize fake images that fool D

D tries to identify the fakes
\[
\arg\max_D \mathbb{E}_{x,y} \left[ \log D(G(x)) + \log(1 - D(y)) \right]
\]
\( \textbf{G} \) tries to synthesize fake images that fool \( \textbf{D} \):

\[
\arg\min_{\textbf{G}} \mathbb{E}_{x,y} \left[ \log D(G(x)) + \log(1 - D(y)) \right]
\]
$\mathbf{G}$ tries to synthesize fake images that fool the best $\mathbf{D}$:

$$\arg \min_G \max_D \mathbb{E}_{x,y} \left[ \log D(G(x)) + \log(1 - D(y)) \right]$$
G’s perspective: D is a loss function.

Rather than being hand-designed, it is *learned* and *highly structured*. 
\[
\arg \min_G \max_D \mathbb{E}_{x,y} \left[ \log D(G(x)) + \log(1 - D(y)) \right]
\]
\[
\arg \min_G \max_D \mathbb{E}_{x,y} \left[ \log D(G(x)) + \log(1 - D(y)) \right]
\]
\[ \arg \min_G \max_D \mathbb{E}_{x,y} \left[ \log D(G(x)) + \log (1 - D(y)) \right] \]
arg min \max_G \mathbb{E}_{x,y}[ \log D(x, G(x)) + \log(1 - D(x, y)) ]

real or fake pair?
\[
\arg \min_G \max_D \mathbb{E}_{x,y} \left[ \log D(x, G(x)) + \log(1 - D(x, y)) \right]
\]
arg min \max_{G,D} \mathbb{E}_{x,y} [ \log D(x, G(x)) + \log(1 - D(x, y)) ]
\[
\arg \min_G \max_D \mathbb{E}_{x,y} \left[ \log D(x, G(x)) + \log(1 - D(x, y)) \right]
\]
Training Details: Loss function

Conditional GAN

\[ G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G). \]
Training Details: Loss function

Conditional GAN

\[ G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G). \]

Stable training + fast convergence

[c.f. Pathak et al. CVPR 2016]
Why deep learning

Performance vs. Amount of data

Deep learning

Older learning algorithms

How do data science techniques scale with amount of data?

[Slide credit: Andrew Ng]
Why structured objectives
(cartoon)
Why structured objectives
(cartoon)

- DL w/ structured objective (e.g., GANs, generative models)
- DL w/ unstructured objective (e.g., least-squares regression)
- Older learning algorithms

Amount of data

Performance
Input

Unstructured prediction (L1)
Training data

\[ x \rightarrow y \]

\[ \{ \text{shoes} \} \rightarrow \{ \text{shoes} \} \]

\[ \{ \text{boots} \} \rightarrow \{ \text{boots} \} \]

\[ \{ \text{ sneakers} \} \rightarrow \{ \text{ sneakers} \} \]

\[ \vdots \]

[HED, Xie & Tu, 2015]
#edges2cats [Chris Hesse]
1. Image synthesis
2. Structured prediction
3. Domain mapping

Domain mapping

[Includes slides from Jun-Yan Zhu, Taesung Park]
[Cartoon: The Computer as a Communication Device, Licklider & Taylor 1968]
Unpaired data

\[
\begin{align*}
\{X_i\} & , \{Y_i\} \\
\{X_i\} & , \{Y_i\} \\
\{X_i\} & , \{Y_i\} \\
\vdots & \\
\end{align*}
\]

Paired data

\[
\begin{align*}
\{x_i\} & , \{y_i\} \\
\{x_i\} & , \{y_i\} \\
\{x_i\} & , \{y_i\} \\
\vdots & \\
\end{align*}
\]
arg min_{G} max_{D} \mathbb{E}_{x,y} [ \log D(x, G(x)) + \log(1 - D(x, y)) ]
No input-output pairs!
\[
\arg \min_G \max_D \mathbb{E}_{x,y} \left[ \log D(G(x)) + \log(1 - D(y)) \right]
\]

Usually loss functions check if output matches a target \textit{instance}.

GAN loss checks if output is part of an admissible \textit{set}. 
Nothing to force output to correspond to input
CycleGAN, or there and back aGAN

[Zhu*, Park* et al. 2017], [Yi et al. 2017], [Kim et al. 2017]
CycleGAN, or there and back aGAN

\[ X \xrightarrow{D_X} Y \xleftarrow{D_Y} X \]
Cycle Consistency Loss

\[ \|F(G(x)) - x\|_1 \]
Cycle Consistency Loss

\[ x \xrightarrow{G} \hat{y} \xrightarrow{F} \hat{x} \]

\[ y \xrightarrow{F} \hat{x} \xrightarrow{G} \hat{y} \]

\[ \| F(G(x)) - x \|_1 \quad \| G(F(y)) - y \|_1 \]
Paired translation

Training data

\{ \begin{align*} x_i & \quad y_i \end{align*} \}

\begin{align*} X \quad \hat{y}_i \quad Y \end{align*}

Input

Result

Objective

regression error

Unpaired translation

Training data

\{ \begin{align*} \ldots & \quad \ldots \end{align*} \}

\begin{align*} X \quad x_i \quad Y \end{align*}

Input

Result

Objective

cycle-consistency error

[“pix2pix”, Isola, Zhu, Zhou, Efros, 2017]

[“CycleGAN”, Zhu*, Park*, Isola, Efros, 2017]
<table>
<thead>
<tr>
<th>Input</th>
<th>Monet</th>
<th>Van Gogh</th>
<th>Cezanne</th>
<th>Ukiyo-e</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Image" /></td>
<td><img src="image2" alt="Image" /></td>
<td><img src="image3" alt="Image" /></td>
<td><img src="image4" alt="Image" /></td>
<td><img src="image5" alt="Image" /></td>
</tr>
<tr>
<td><img src="image6" alt="Image" /></td>
<td><img src="image7" alt="Image" /></td>
<td><img src="image8" alt="Image" /></td>
<td><img src="image9" alt="Image" /></td>
<td><img src="image10" alt="Image" /></td>
</tr>
<tr>
<td><img src="image11" alt="Image" /></td>
<td><img src="image12" alt="Image" /></td>
<td><img src="image13" alt="Image" /></td>
<td><img src="image14" alt="Image" /></td>
<td><img src="image15" alt="Image" /></td>
</tr>
<tr>
<td><img src="image16" alt="Image" /></td>
<td><img src="image17" alt="Image" /></td>
<td><img src="image18" alt="Image" /></td>
<td><img src="image19" alt="Image" /></td>
<td><img src="image20" alt="Image" /></td>
</tr>
</tbody>
</table>
GANs

Gaussian

Target distribution

\( Z \) → \( Y \)
What would it look like if…?
What would it look like if…?

MRI → CT

[Wolterink et al, 2017]

Sim → “Real”

[Hoffman et al, 2018]