Generative Models

BOSTON UNIVERSITY
MACHINE INTELLIGENCE
COMMUNITY

Devin de Hueck, Darcy Meyer, Duy Nguyen
10/16/2019

A Recap

1. Feed-forward neural networks
2. How we train neural networks
3. Convolutional neural networks

What will be covered today

1. Generative vs. Discriminative Models
2. Autoencoders - high level
3. Generative Adversarial Models (GANs)
4. Training GANs

Generative vs. Discriminative Models

Discriminative Models

- Model learns the decision boundary to discriminate the data
- Estimates:

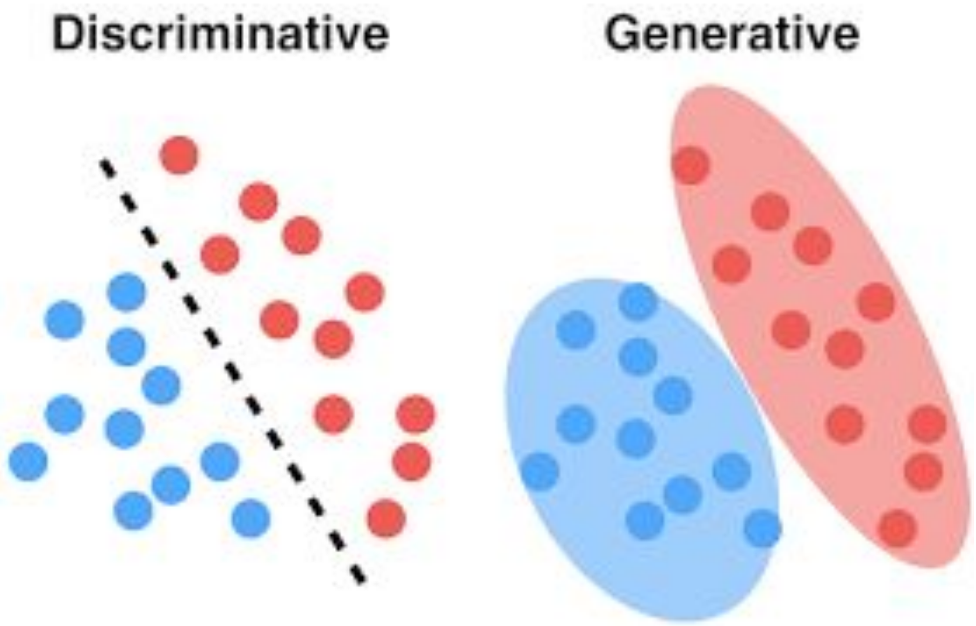
$$P(y|x)$$

Generative Models

- Model learns the probability distribution of the data in order to generate samples from the data.
- Estimates:

$$P(x|y)$$

A Comparison

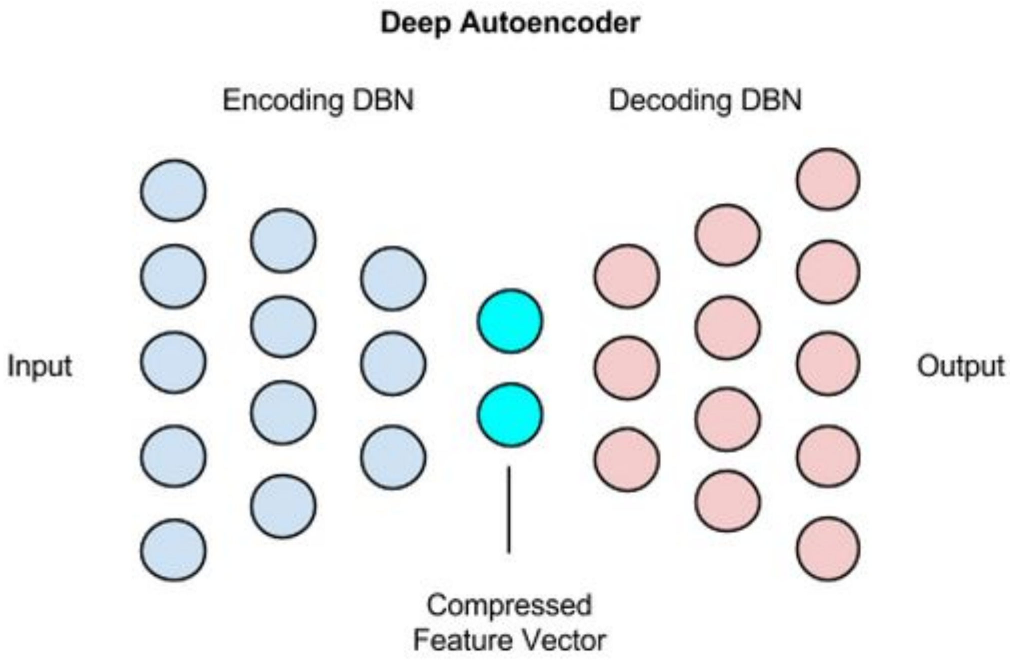


Autoencoders

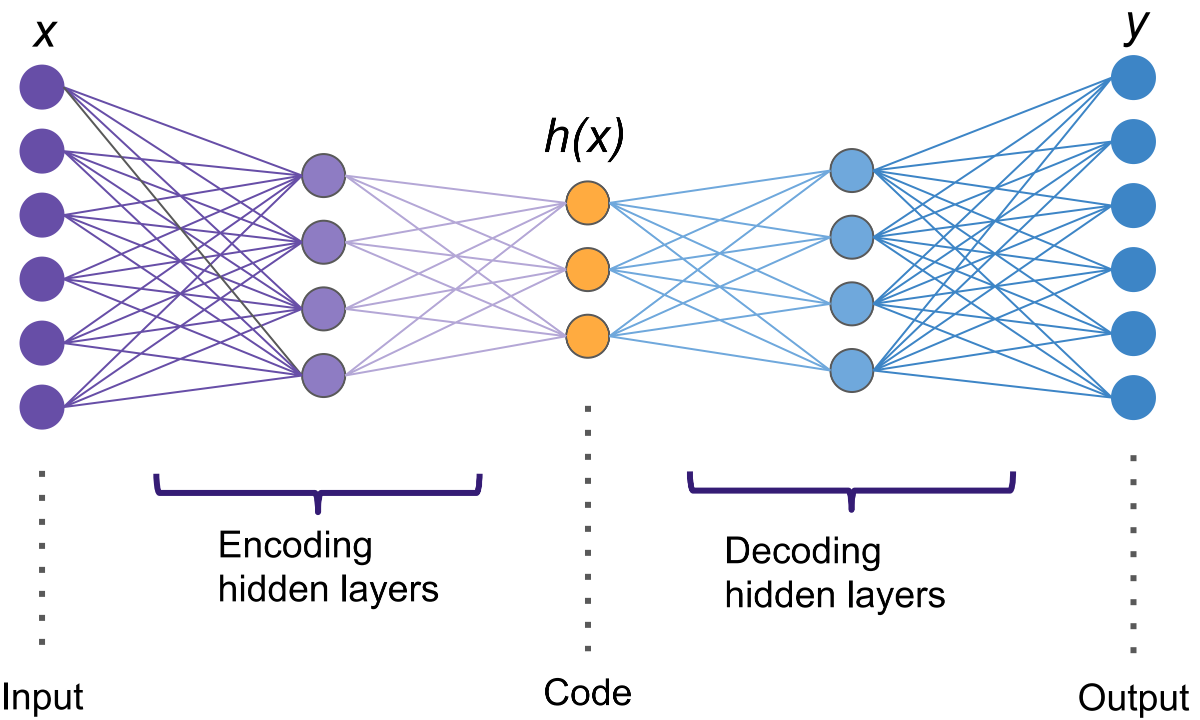
A Generative Model

Autoencoder Intuition

- Introduce a bottle neck that compresses the input into a *latent-space representation.*
- The *encoder* turns the input into the latent-space $h = f(x)$
- The *decoder* reconstruct the input from the latent-space representation $r = g(h)$



Autoencoder Latent Space



Source: [MIC 2017 Unsupervised Learning]

Applications of Autoencoders

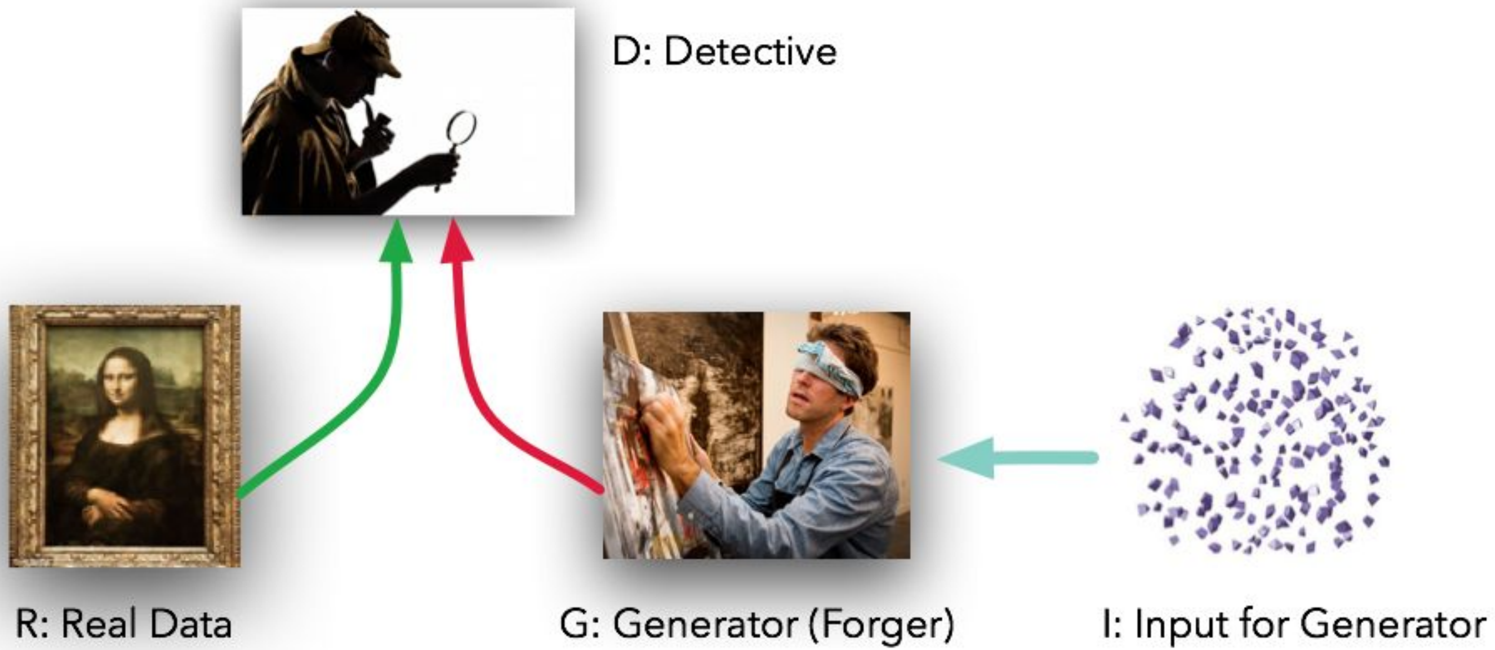
- Anomaly detection
- Image reconstruction
- Denoising
- Dimensionality reduction

Autoencoders for Image Generation

[Going Beyond GAN? New DeepMind VAE Model Generates High Fidelity Human Faces]

Generative Adversarial Networks (GANs)

GAN Intuition - What are GANs?

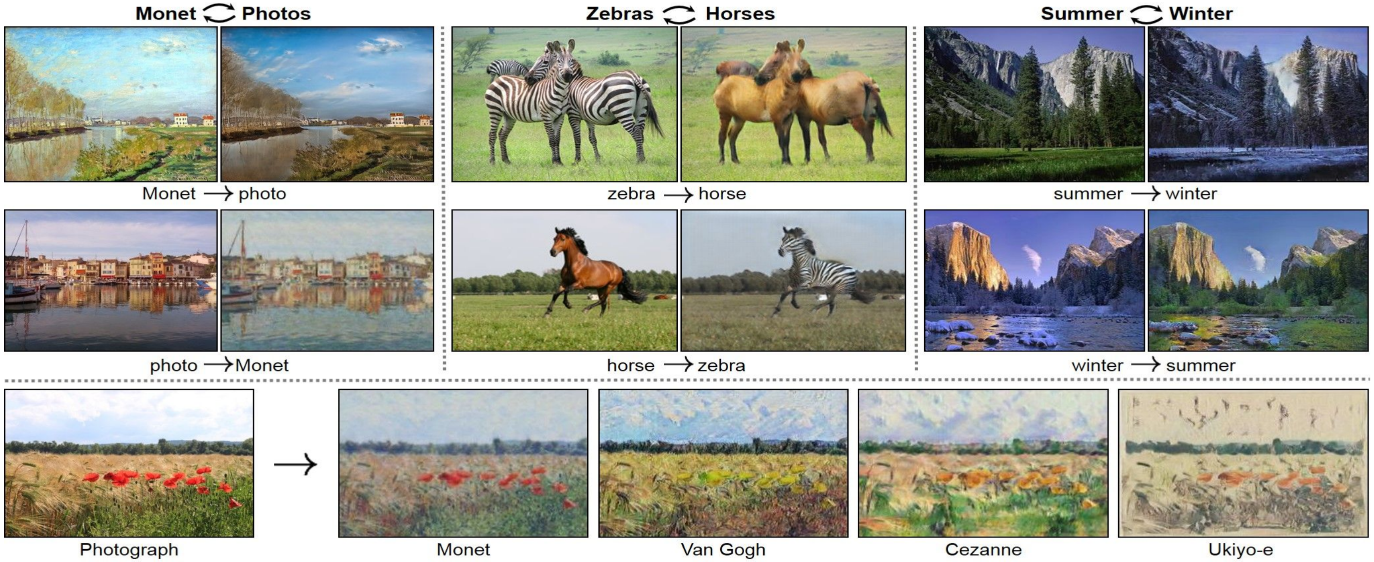


History of GANs

2014	2015	2016	2017
GAN	DCGAN	INFOGAN	CycleGAN
The original paper	Using convolutional networks in a GAN	Learning disentangled representations - vector arithmetic properties	Image to Image translation

And Many More!

Applications of GANs - Image to Image Translation



https://github.com/junyanz/CycleGAN

Applications of GANs - Face Generation

https://arxiv.org/pdf/1710.10196.pdf

Applications of GANs - Style Transfer

https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Gatys_Image_Style_Transfer_CVPR_2016_paper.pdf

How do GANs Work?

Two Opposing Models

The Forger
(The Generator)

The Detective
(The Discriminator)

Some Definitions

$\boldsymbol{x}$: space in which examples reside (space that Generator outputs to, and the Discriminator discriminates in)

$\boldsymbol{z}$: some space (space that the Generator samples from)

p_{g}: the distribution in $\boldsymbol{x}$ of the outputs of the generator

p_{data}: the distribution in $\boldsymbol{x}$ of the actual data

The Generator - G(z)

Generator $G(z)$: Given random input from $\boldsymbol{z}$, output an example in $\boldsymbol{x}$

G outputs examples in a distribution p_{g} in $\boldsymbol{x}$

The Discriminator - D(x)

Discriminator $D(\boldsymbol{x})$: Given an example in $\boldsymbol{x}$, output probability it is a real example

D outputs probability that the example came from p_{data} (the real examples) rather than p_{g} (the generated examples)

GAN Definition

Goal of Generator: map $\boldsymbol{z}$ to a distribution p_{g} in $\boldsymbol{x}$

- We want p_g to converge to p_{data}

Goal of Discriminator: determine whether $\boldsymbol{x}$ comes from p_{data} or p_{g}

- Accuracy of D is ½ when p_{g} has converged to p_{data}

To motivate this we define a value function:

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

The Generator's Goal

Create convincing generated examples

- Value function is small when the Discriminator predicts that the generated example is from the data
- G wants to minimize the value function

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

This is small when $D(G(z))$ is close to 1 (when the discriminator thinks the generated example is from the data)

The Discriminator's Goal

Correctly distinguish between real and generated examples

- Discriminator: Value function is large when D predicts that examples from the data are from the data, and examples from the generator are from the generator (i.e. it makes correct predictions)
- D wants to **maximize** the value function

$$\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\underbrace{[\log D(\boldsymbol{x})]}\ + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\underbrace{[\log(1 - D(G(\boldsymbol{z})))]}$$

This is large when $D(x)$ is close to 1 (when the discriminator identifies the real examples as real)

This is large when $D(G(z))$ is close to 0 (when the discriminator identifies the generated example as generated)

GAN Definition

2-player minimax game:

- D tries to correctly classify real examples and generated examples
- G tries to fool D by creating generated examples that are mistaken for real examples

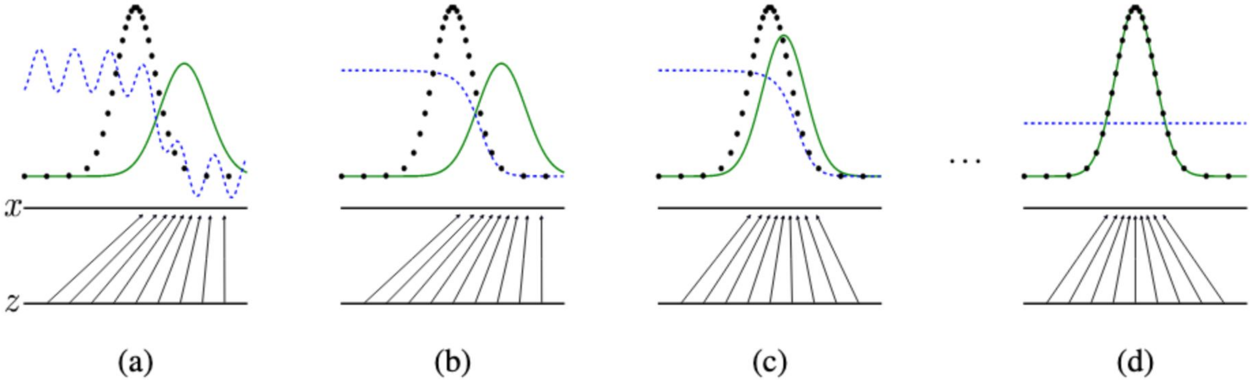
G tries to **minimize** and D tries to **maximize** the value function:

$$\underbrace{\min_G \max_D}_{\substack{\text{Minimize over G,} \\ \text{maximize over D}}} V(D,G) = \underbrace{\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})]}_{\substack{\text{Large when D assigns} \\ \text{correct label}}} + \underbrace{\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]}_{\substack{\text{Small when} \\ \text{G fools D}}}$$

GAN Convergence

- Black: p_{data} (real examples)
- Green: p_g (generated examples)
- Blue: D's prediction

p_G converges to p_{data}, and the D's prediction converges to ½ (can't distinguish between real and generated)



Training GANs

Training GANs - The Algorithm

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

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

  end for

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

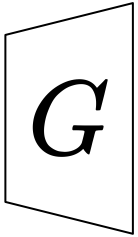
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

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

The Algorithm - Updating the Discriminator

1. Sample minibatch of m noise samples $\{z_1, \ldots, z_m\}$ from a random distribution.
2. Then run samples through the generator (G) - to get fake examples

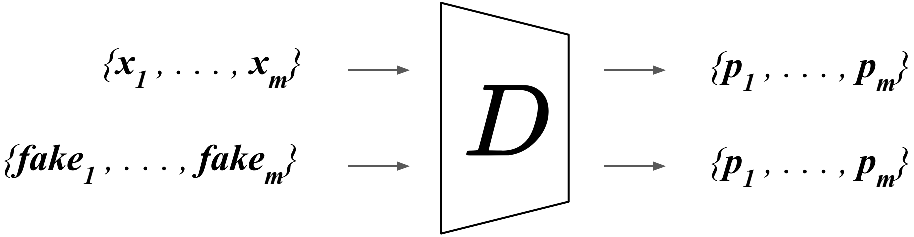
$\{z_1, \ldots, z_m\} \longrightarrow$  $\longrightarrow \{fake_1, \ldots, fake_m\}$

3. Sample minibatch of m examples $\{x_1, \ldots, x_m\}$ from the true dataset

Real	Fake
$\{x_1, \ldots, x_m\}$	$\{fake_1, \ldots, fake_m\}$

The Algorithm - Updating the Discriminator

4. Get probability scores for real and fake examples



Real

$\{p_1, \ldots, p_m\}$

Fake

$\{p_1, \ldots, p_m\}$

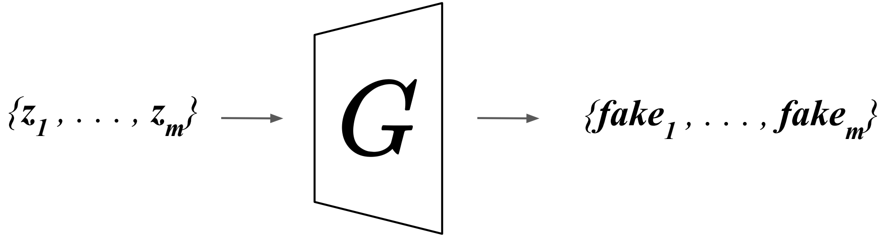
The Algorithm - Updating the Discriminator

5. Use probability scores to update the discriminator's parameters

$$\theta_G := \theta_G - \alpha \cdot \nabla_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} \left[\log \overbrace{D\left(\boldsymbol{x}^{(i)}\right)}^{\substack{\text{Real} \\ \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m\}}} + \log\left(1 - \overbrace{D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)}^{\substack{\text{Fake} \\ \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m\}}}\right) \right]$$

The Algorithm - Updating the Generator

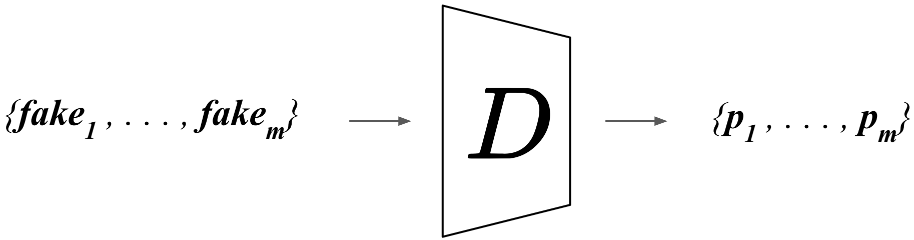
6. Sample minibatch of $\boldsymbol{m}$ noise samples $\{z_1, \ldots, z_m\}$ from a random distribution.
7. Then run samples through the generator ($\boldsymbol{G}$) - to get fake examples



Fake
$\{fake_1, \ldots, fake_m\}$

The Algorithm - Updating the Generator

8. Get probability scores for new fake examples



Fake

$\{p_1, \ldots, p_m\}$

The Algorithm - Updating the Generator

9. Use probability scores to update the generator's parameters

$$\theta := \theta - \alpha \cdot \nabla_\theta \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - \underbrace{D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)}_{\substack{\text{Fake} \\ \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m\}}}\right)$$

Training GANs - The Algorithm - Recap

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

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

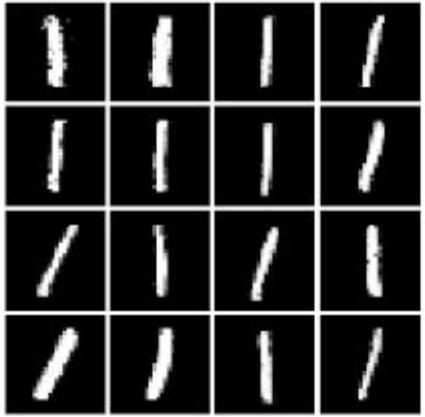
  end for
  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

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

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

Training GANs - Issues

- Non-Convergence
- Mode Collapse
- Diminished Gradient
- Sensitive to Hyperparameters!



Mode Collapse on MNIST

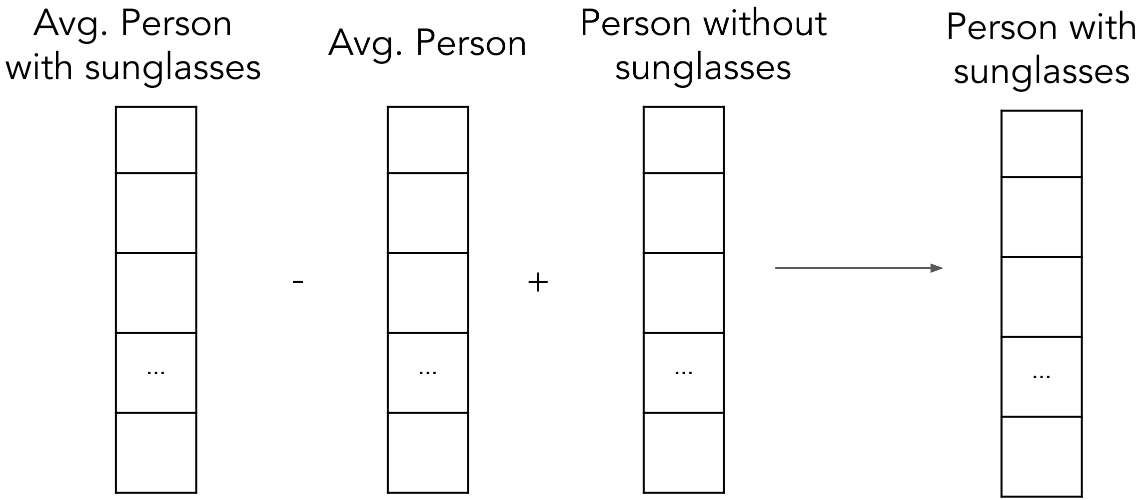
https://medium.com/@jonathan_hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b

An Interesting Property of GANs - Latent Vector Arithmetic

(b) Presence or absence of glasses

https://arxiv.org/pdf/1606.03657.pdf

An Interesting Property of GANs - Latent Vector Arithmetic



Project Ideas

Open Questions

Open Questions about Generative Adversarial Networks

What we'd like to find out about GANs that we don't know yet.

Problem 1	What are the trade-offs between GANs and other generative models?
Problem 2	What sorts of distributions can GANs model?
Problem 3	How can we Scale GANs beyond image synthesis?
Problem 4	What can we say about the global convergence of the training dynamics?
Problem 5	How should we evaluate GANs and when should we use them?
Problem 6	How does GAN training scale with batch size?
Problem 7	What is the relationship between GANs and adversarial examples?

Coding Example

Link to Collab Notebook and Github - https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html

References & Further Reading

1. Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.
2. Chen, Xi, et al. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets." Advances in neural information processing systems. 2016.
3. Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).

M I C

Thank you for coming!