

# Generative Adversarial Networks

---

A gentle introduction and application for domain adaptation

**Mariana Bento**  
Assistant Professor  
Biomedical Engineering

F2024



UNIVERSITY OF  
**CALGARY**

# Keep Up with the Presentation!!!

---

- We are going to have questions along the slides :O
  - Please answer them on the chat or unmute: let's discuss
- Please count how many times this pic appears on the slides
  - (I know, I am “borrowing” one of Wallace’s great idea)
- At the end of the presentation, please share:
  - Something you learn
  - Share your knowledge (teach us something)



# Outline

---

- Generative Adversarial Networks (GANs)
  - What are GANs?
  - How GANs work?
  - Types of GANs and applications
- Domain adaptation

# What are GANs?

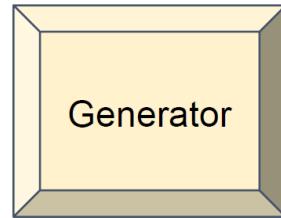
---

- GANs are unsupervised deep learning methods
- GANs are considered one of the greatest deep learning breakthroughs in recent years
- They all operate under the same principle of having modules with adversarial (*i.e.*, competing objectives)

# What are GANs?

---

- Two separate networks compete against each other: **generator** and **discriminator**
- This can be thought of as a game between a **counterfeiter** and an **art curator**



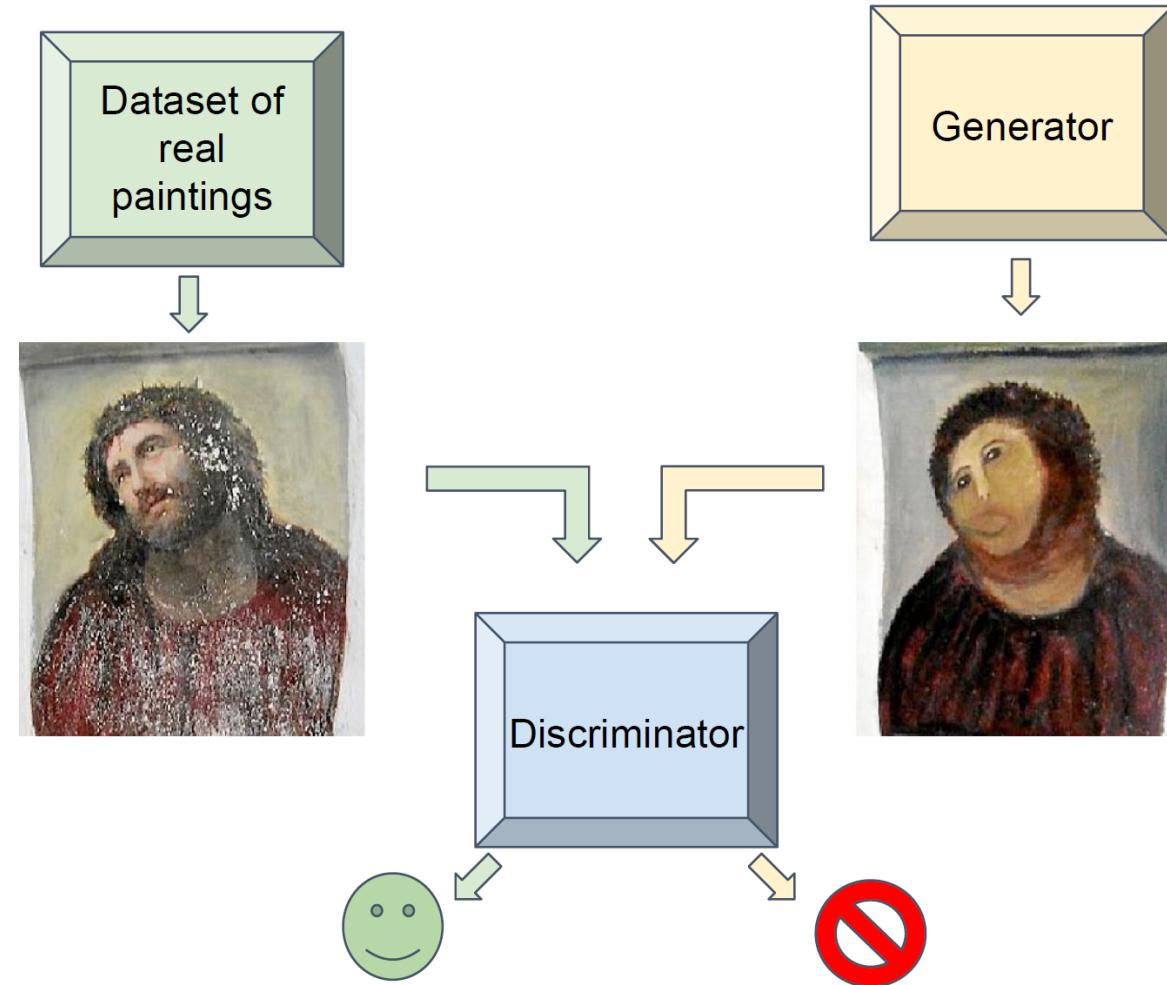
Generates samples that  
look convincingly real



Determines whether a  
sample is real or generated

# What are GANs?

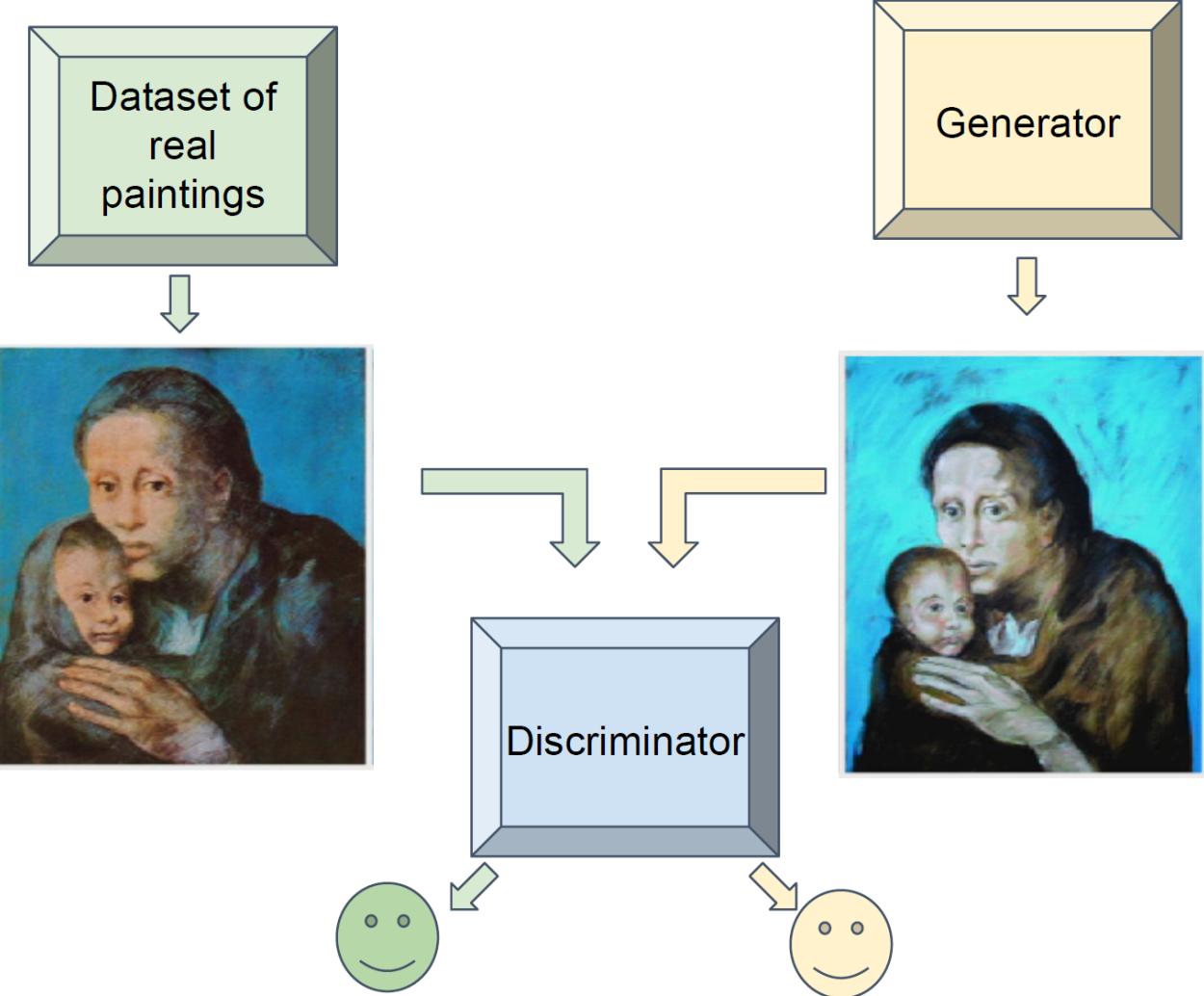
- The counterfeiter (generator) attempts to generate real looking images
  - At first the generator is pretty bad



<https://www.pri.org/stories/2012-08-25/amateur-restoration-botches-jesus-painting-spain>

# What are GANs?

- Eventually, the generator learns to fool the discriminator and can generate convincing images



<https://www.channel4.com/news/art-forgery-beltracchi-wolfgang-ernst-picasso-paraic-obrien>

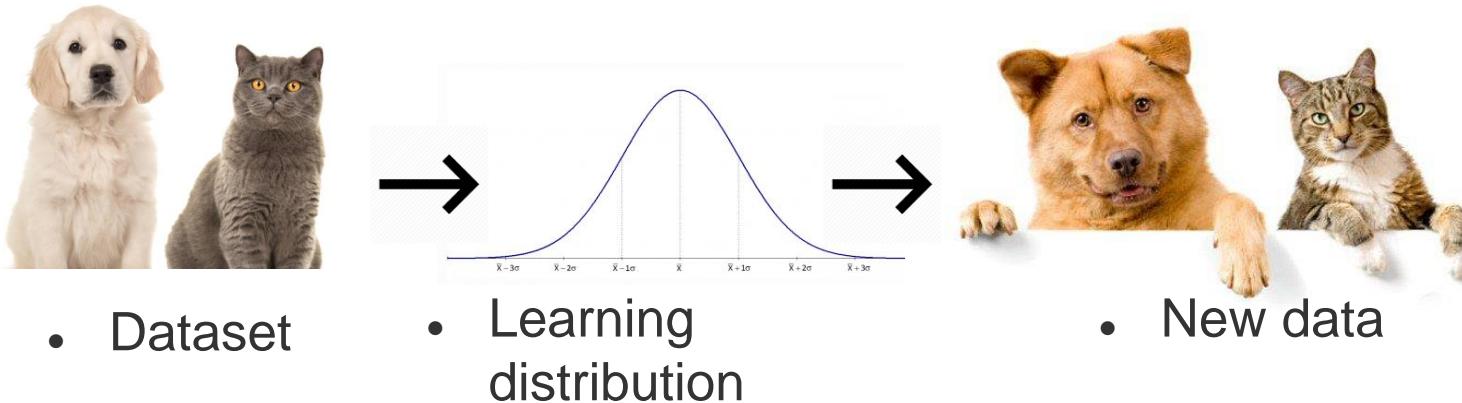
# Generative vs Discriminate Models

---

## 1) Discriminative models



## 2) Generative models



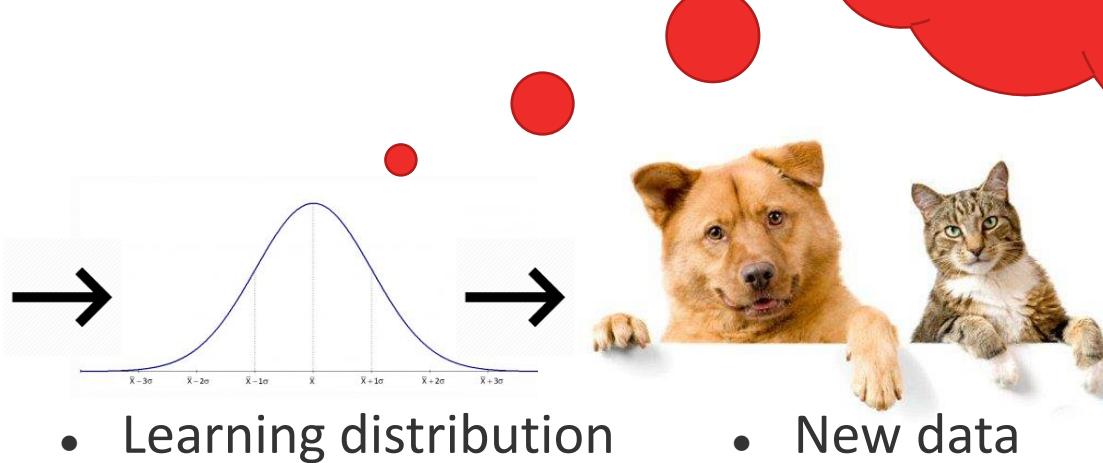
# Generative vs Discriminate Models

## 1) Discriminative models



GANs are generative models where the data distribution is learned implicitly

## 2) Generative models



- Dataset

- Learning distribution

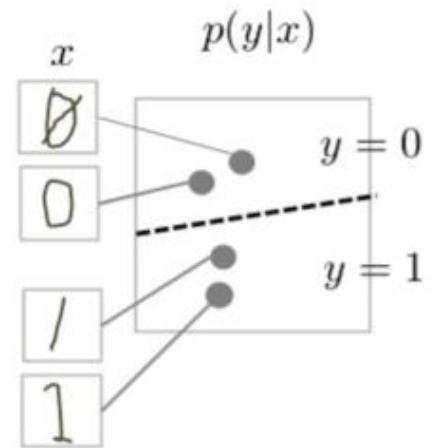
- New data

# Generative vs Discriminate Models

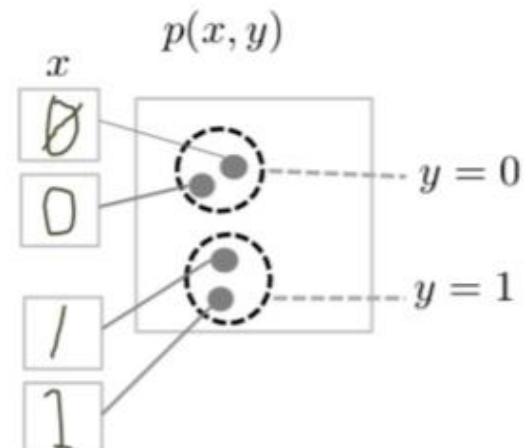


- The discriminative model tries to tell the difference between handwritten 0's and 1's by drawing a line in the data space
- The generative model tries to produce 1's and 0's that fall close to their real counterparts in the data space
  - Model the distribution throughout the data space

- Discriminative Model



- Generative Model



# Question #1

---

- You have IQ scores for 1000 people. You model the distribution of IQ scores with the following procedure:
  - Roll three six-sided dice
  - Multiply the roll by a constant  $w$
  - Repeat 100 times and take the average of all the results
  - You try different values for  $w$  until the result of your procedure equals the average of the real IQ scores. Is your model a generative model or a discriminative model?
- (a) Generative model
- (b) Discriminative model
- (c) Not enough information to tell

# Question #1

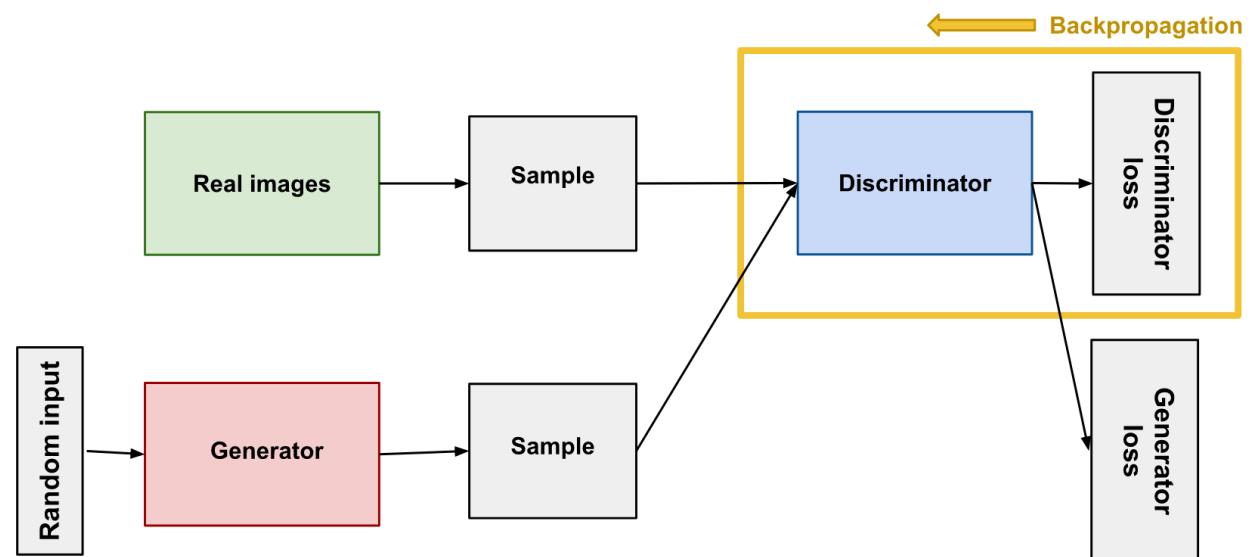
---

- You have IQ scores for 1000 people. You model the distribution of IQ scores with the following procedure
- **(a) Generative model:** with every roll you are effectively generating the IQ of an imaginary person. Furthermore, your generative model captures the fact that IQ scores are distributed normally (that is, on a bell curve)
- **(b) Discriminative model:** might try to classify an IQ as fake or real

# Discriminative Models

---

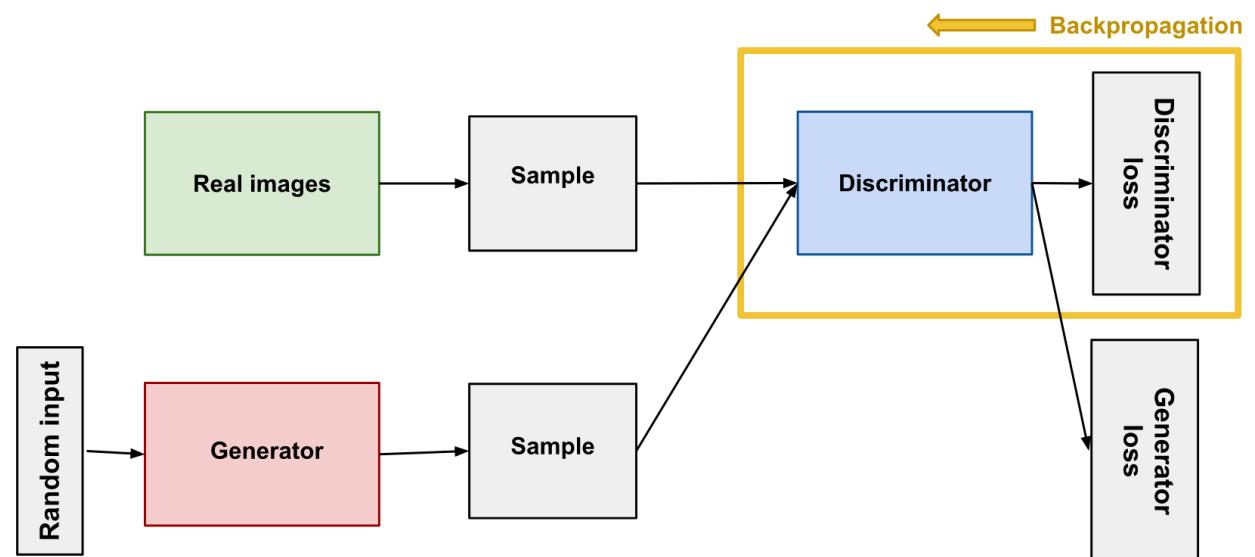
- The discriminator in a GAN is simply a classifier
- During discriminator training:
  - Classifies real data and fake data from the generator
  - The discriminator loss penalizes it for misclassification
  - Updates its weights through backpropagation



# Discriminative Models

---

- The discriminator connects to two loss functions:
  - During discriminator training, the discriminator ignores the generator loss and just uses the discriminator loss
  - The generator loss is used during generator training



# Generative Models

---

- Learns to create fake data by incorporating feedback from the discriminator
  - Learns to make the discriminator classify its output as real
- Generator training:
  - random input
  - generator network, which transforms the random input into a data instance
  - discriminator network, which classifies the generated data
  - discriminator output
  - generator loss, which penalizes the generator for failing to fool the discriminator



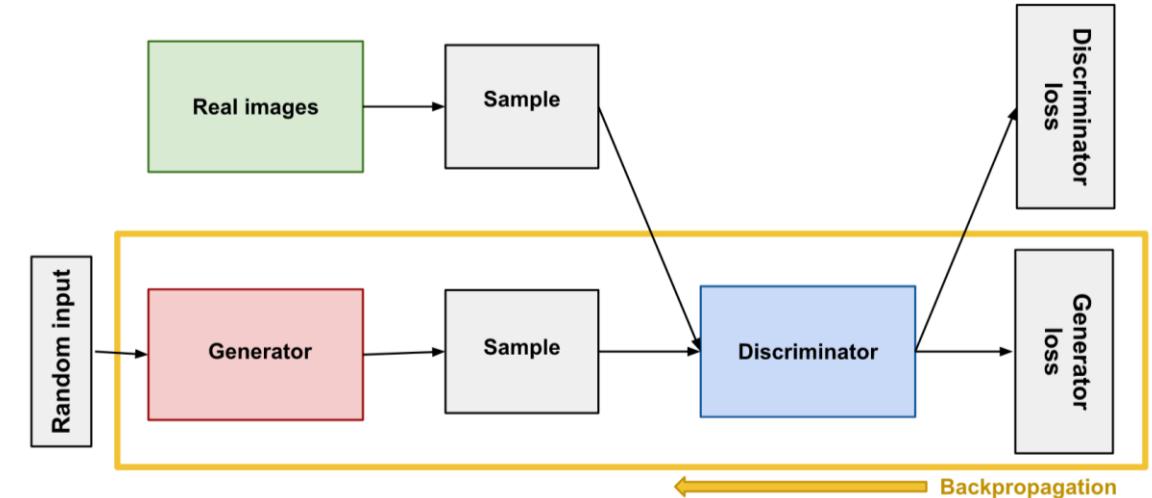
# Generative Models – Random Input

---

- GAN takes random noise as its input
- The generator transforms this noise into a meaningful output
- By introducing noise, the GANs can produce a wide variety of data, sampling from different places in the target distribution
- Experiments suggest that the distribution of the noise doesn't matter much, so we can choose something that's easy to sample from, like a uniform distribution

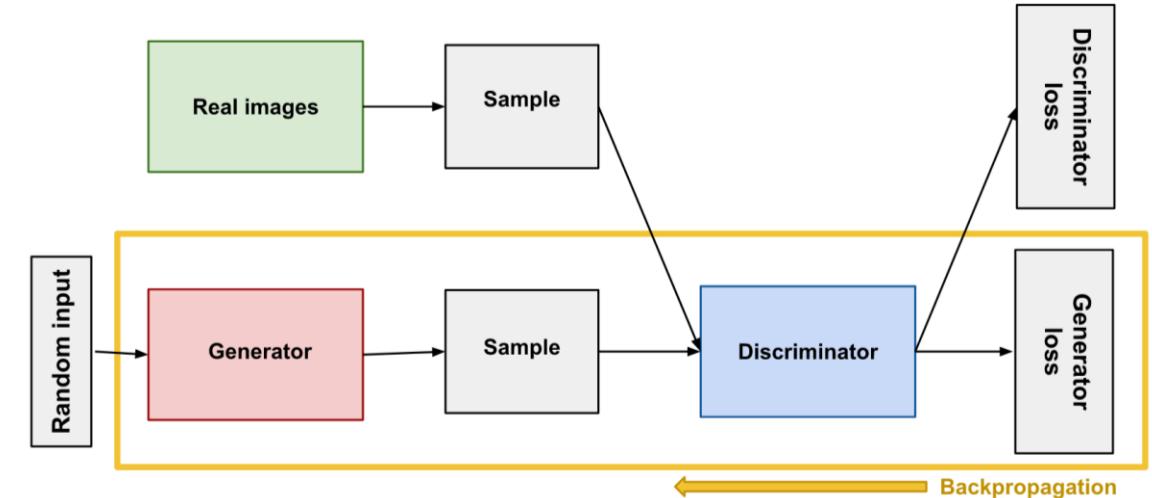
# Generative Models – Discriminator to train the generator

- The generator feeds into the discriminator
- *Discriminator* produces the output
- The generator loss penalizes the generator for producing a sample that the discriminator network classifies as fake
- **The discriminator do not change during generator training**
  - Trying to hit a moving target would be a harder problem for the generator



# Generative Models – Training

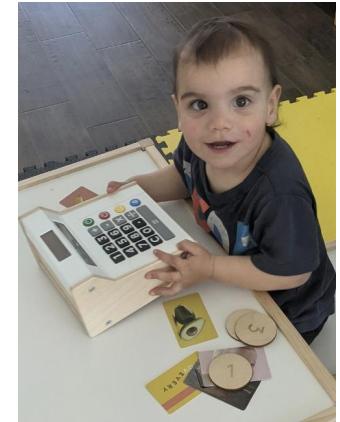
- Sample random noise
- Produce output from sampled random noise
- Get discriminator "Real" or "Fake" classification for generator output
- Calculate loss from discriminator classification
- Backpropagate through discriminator and generator to obtain gradients
  - Change only the generator weights



## Question #2

---

- GANs are trained by having a \_\_\_\_\_ and a \_\_\_\_\_, two separate networks that \_\_\_\_\_ each other:
  - (a) discriminator, generator, compete
  - (b) discriminator, generator, collaborate
  - (c) discriminator in the source data, discriminator in the target data, compete



## Question #2

---

- GANs are trained by having a \_\_\_\_\_ and a \_\_\_\_\_, two separate networks that \_\_\_\_\_ each other:
- **(a) discriminator, generator, compete**

There is a competition here !

Generator tries to make fakes that look real  
and fool the discriminator

Discriminator learns how to distinguish  
reals from fakes

# How do GANs Work?

---

- Mathematically, this can be expressed as a **MinMax** game, where the generator tries to **minimize** the objective function  $V$  while the discriminator tries to **maximize** it. Both networks are trained successively

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

$D(x)$  is a neural network that predicts how confident it is that the input image  $\mathbf{x}$  is real. “The **probability** that the sample is real”

$G(z)$  is a neural network that generates an image given a noise signal  $z$

# How do GANs Work?

---

- When training the **generator**, we want to **minimize** the term in blue, i.e. we want to maximize the error that  $D$  will make on a generated image  $G(z)$

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

- When training the **discriminator**, we want to **maximize** the terms in red, i.e. we want to minimize the error that  $D$  will make on a generated image  $G(z)$ .

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

# Question #3

---

- Select the true statement about GANs
  - (a) When training the generator, we want to minimize the error that the discriminator will make on a generated image
  - (b) When training the discriminator, we want to maximize the error that the discriminator will make on a generated image
  - (c) When training the discriminator, we want to minimize the error that the discriminator will make on a generated example

# Question #3

---

- Select the true statement about GANs
  - (a) When training the generator, we want to minimize the error that the discriminator will make on a generated image
  - (b) When training the discriminator, we want to maximize the error that the discriminator will make on a generated image
  - **(c) When training the discriminator, we want to minimize the error that the discriminator will make on a generated example**

## Question #4

---

- A typical GAN trains the generator and the discriminator simultaneously
  - True or False



## Question #4

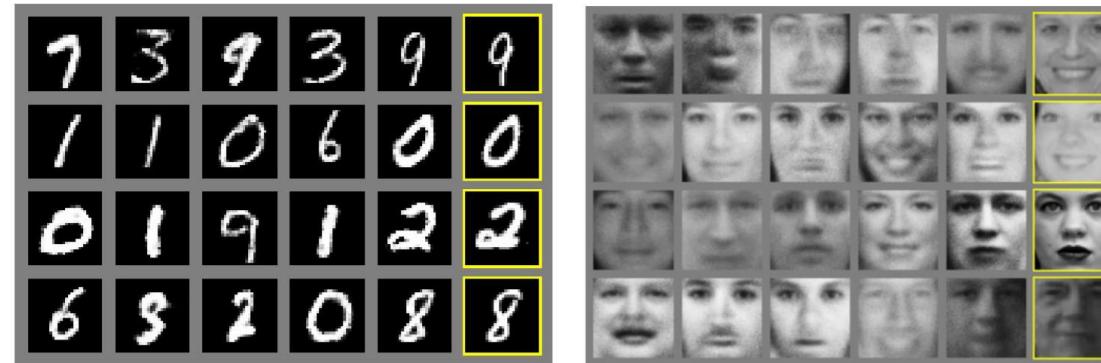
---

- A typical GAN trains the generator and the discriminator simultaneously
- True or False: A typical GAN alternates between training the discriminator and training the generator
- **The discriminator do not change during generator training**
  - Trying to hit a moving target would be a harder problem for the generator

# Types of GANs and applications

---

- In the original paper, both digits and faces were generated by the network and look convincingly real. The yellow boxes show the closest match to its generated neighbors in the training dataset



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

# DCGAN

- In a follow up paper, CNNs are used to generate the images

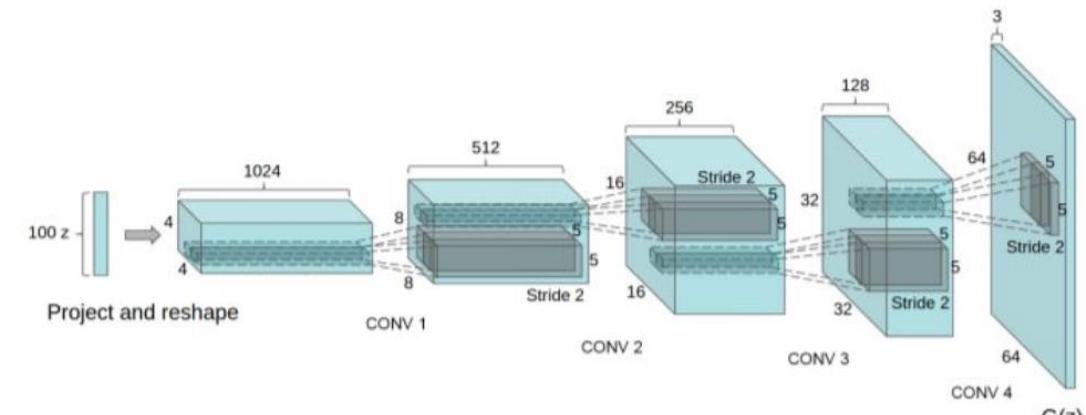
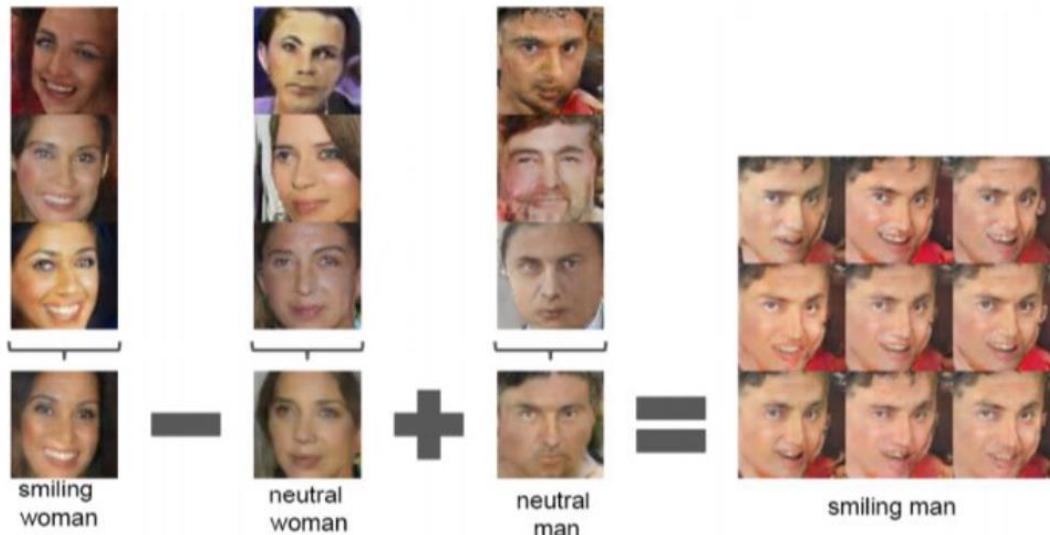


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution  $Z$  is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions)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

<https://arxiv.org/pdf/1511.06434.pdf>

# Image Synthesis

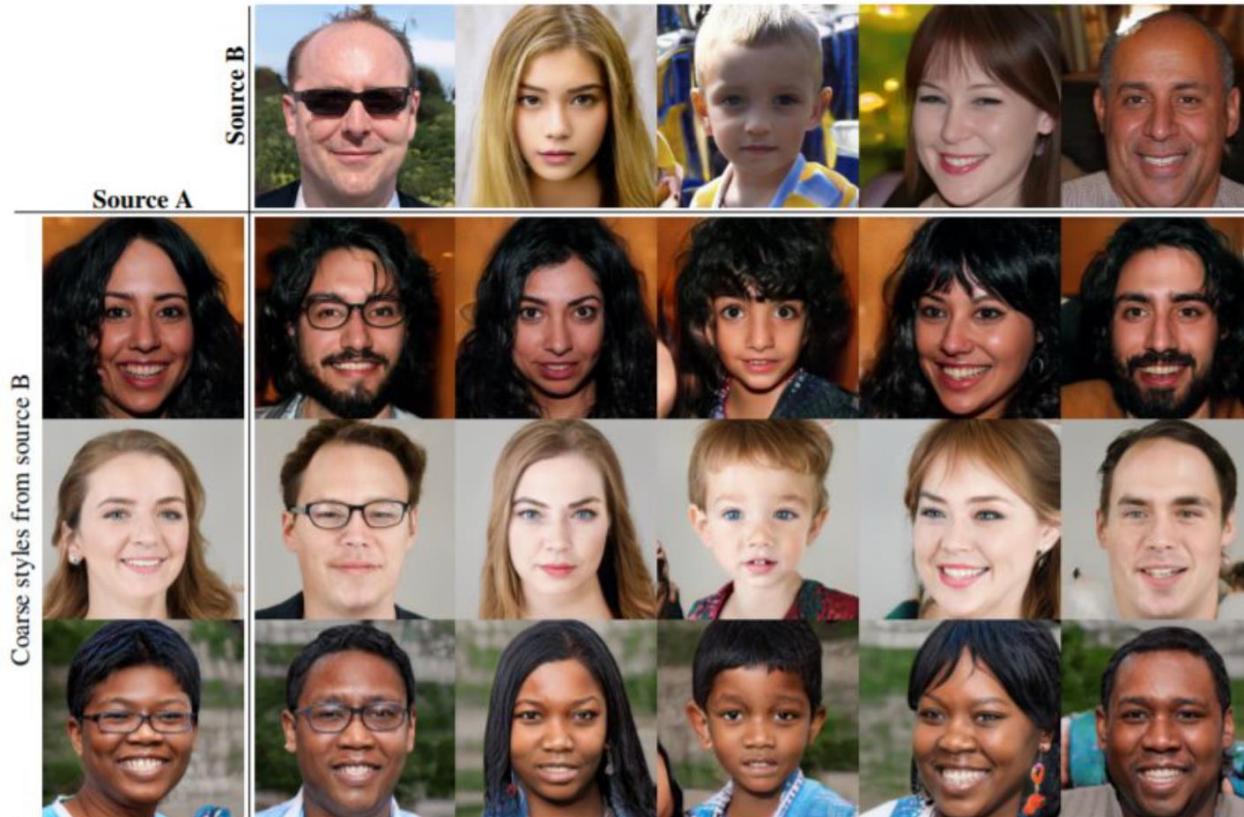
---

- We can teach networks to generate realistic images that have never been seen before by the network. Which face is fake?



# Style Transfer

---



<https://arxiv.org/pdf/1812.04948.pdf>

# Face Inpainting

---



- Chunks of an image are blacked out, and the system tries to fill in the missing chunks
- GAN outperformed other techniques for inpainting images of faces



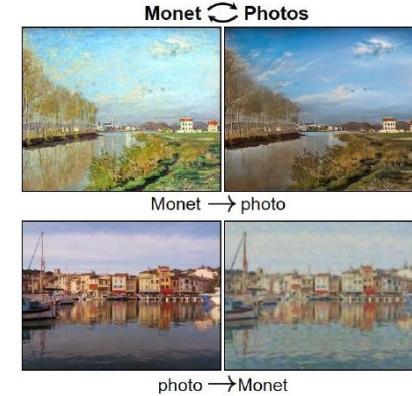
# CycleGAN

---

- CycleGan introduced the concept of **cycle-consistency** loss to their GANs. This allowed training generators that would translate from domain X to domain Y
- This allows map between arbitrary domains with realistic results



*horse → zebra*



<https://junyanz.github.io/CycleGAN/>  
<https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix>

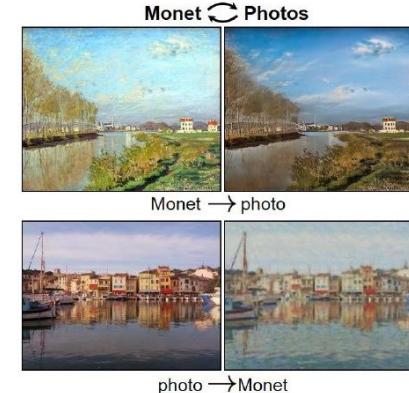
# CycleGAN

---

- The training data for the CycleGAN is simply two sets of images (a set of horse images and a set of zebra images). The system requires no labels or pairwise correspondences between images



*horse → zebra*



<https://junyanz.github.io/CycleGAN/>  
<https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix>

# GANs Limitations

---

- **Non-convergence:** the model parameters oscillate, and the model does not converge
- **Mode collapse:** the generator collapses and produces a limited number of different samples
- **Diminished gradient:** the discriminator is too good that the generator gradient vanishes and learns nothing
- **Highly sensitive** to the hyperparameter selections

# Summary

---

- GANs are unsupervised techniques
- They can be used to generate synthetic data that can potentially be used to train other deep learning models
- There are different GAN types, but they are all based on the principle of having two competing objectives
- GANs often face instabilities during training

# Current Experiments

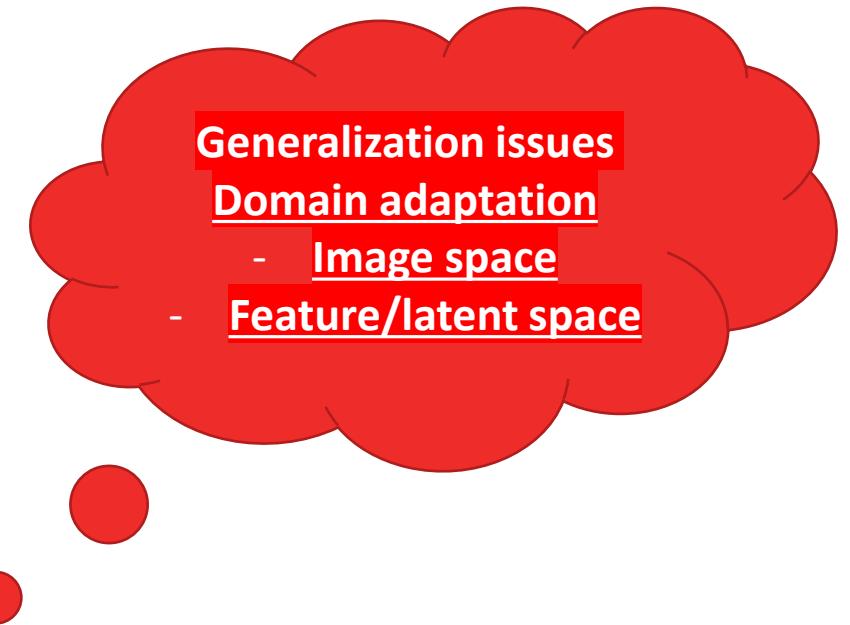
---

- ADNI : AD vs NC
- Experimental design
  - Set1\_ Training/validation
  - Set2\_ Testing
  - Set3\_ Secondary testing set (other public dataset CC359)
- Using a common transfer learning approach:
  - Accuracy on training/validation and testing (Set1 & Set2) are similar and high
  - Accuracy on secondary testing set (Set3) is low

# Current Experiments

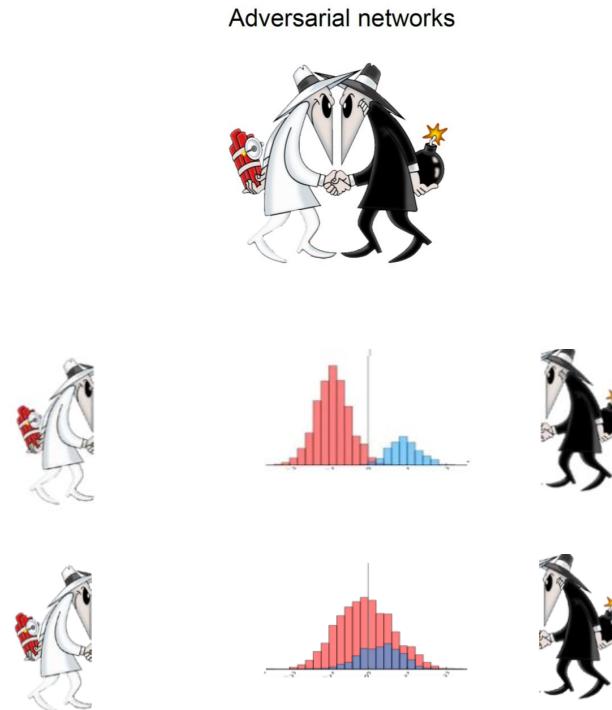
---

- ADNI : AD vs NC
- Experimental design
  - Set1\_ Training/validation
  - Set2\_ Testing
  - Set3\_ Secondary testing set (other public dataset CC359)
- Using a common transfer learning approach:
  - Accuracy on Training/Validation and Testing (Set1 & Set2) are similar and high
  - Accuracy on secondary testing set (Set3) is low

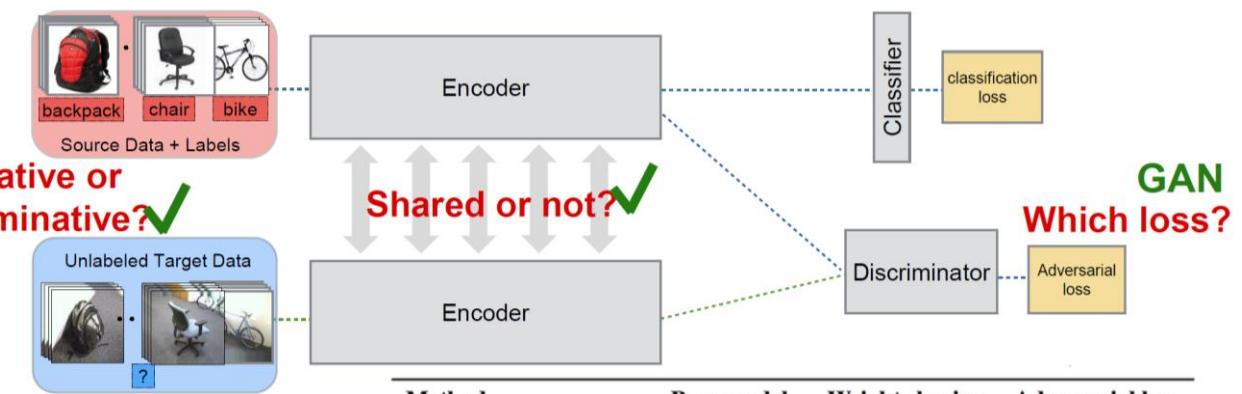


# Domain Adaptation - Feature Space

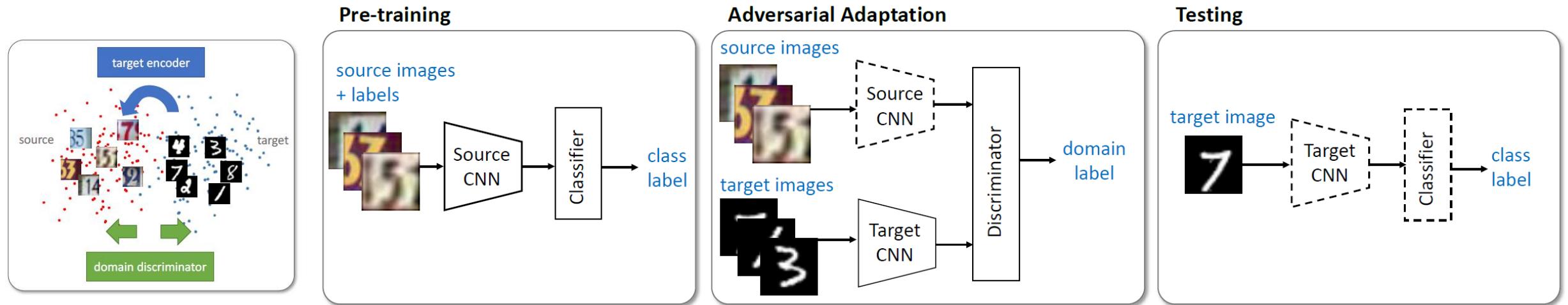
- Train a network to minimize classification loss AND confuse two domains



## Adversarial Discriminative Domain Adaptation (ADDA) (in submission)

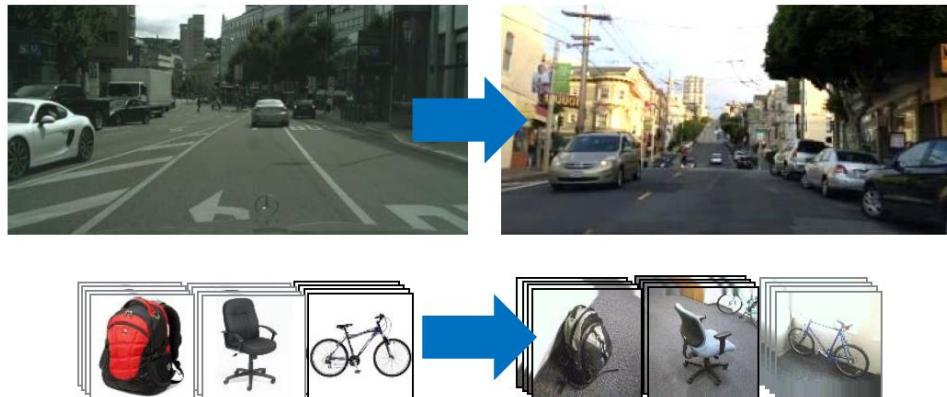


# Domain Adaptation using Adversarial Networks

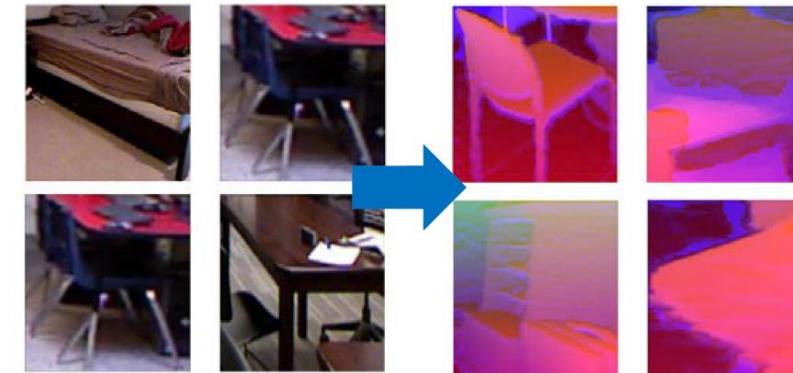


# Domain Adaptation using Adversarial Networks

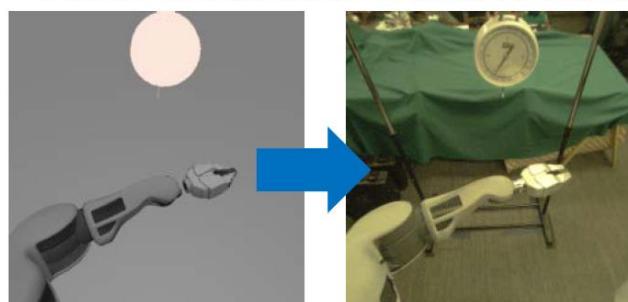
From dataset to dataset



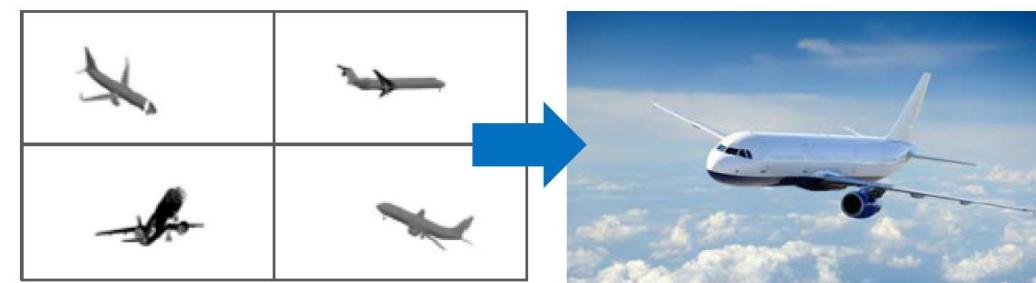
From RGB to depth



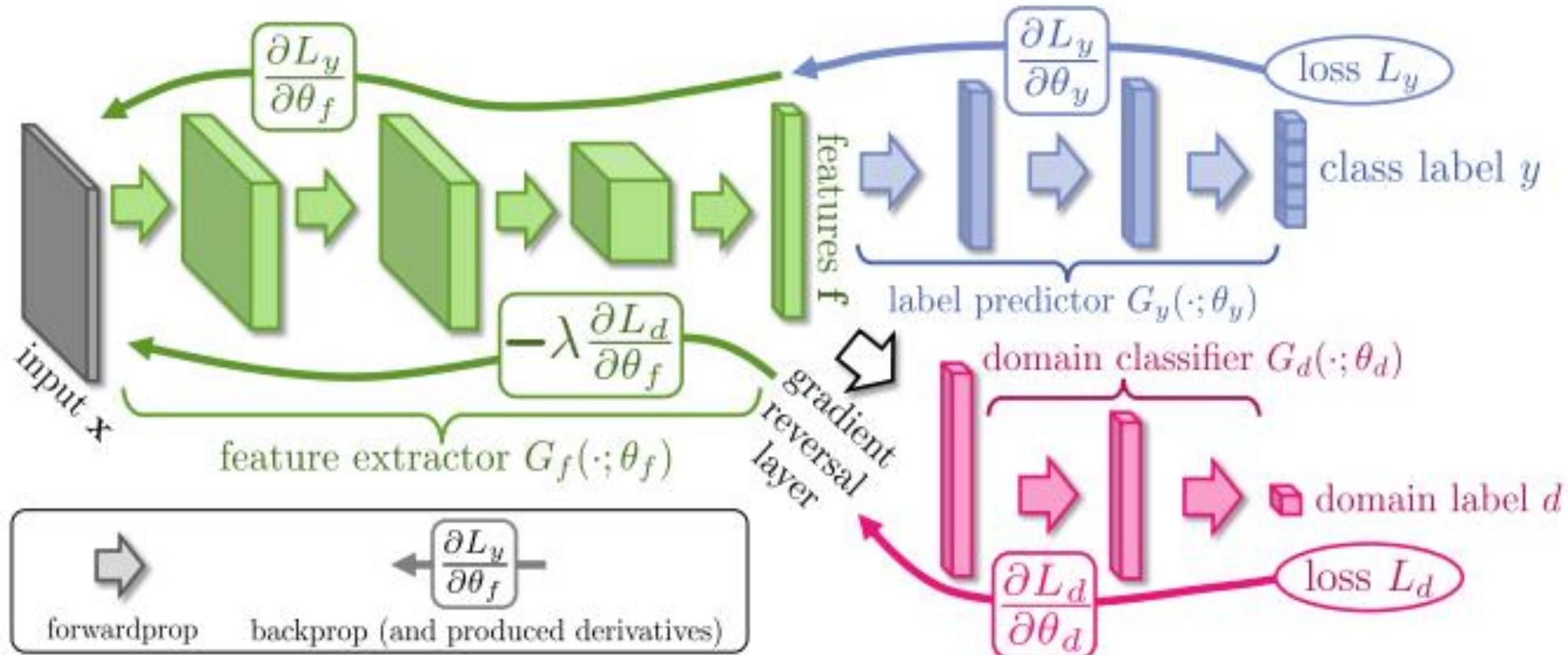
From simulated to real control



From CAD models to real images

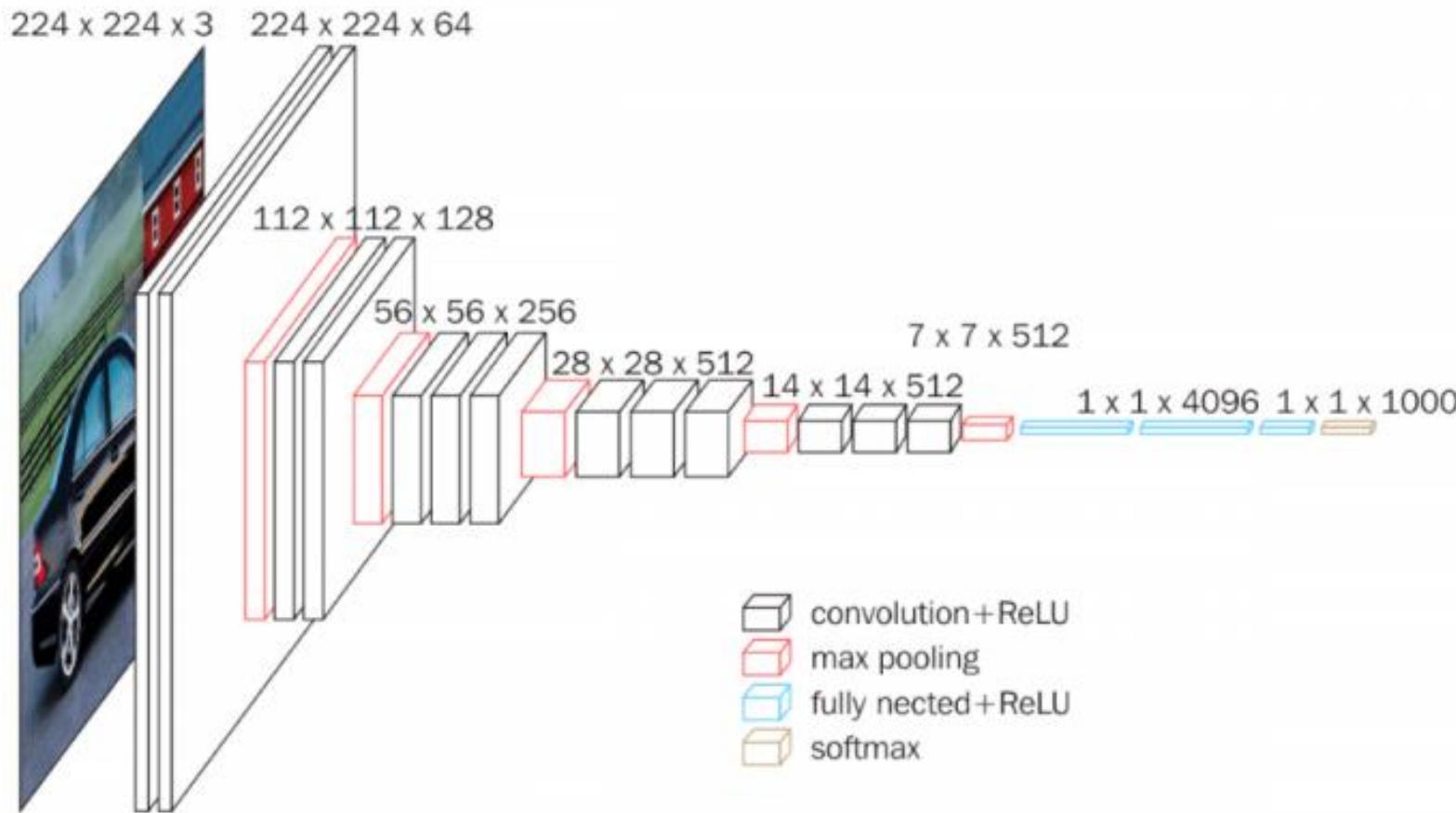


# Domain Adaptation – Gradient Reversal



# Domain Adaptation – Gradient Reversal

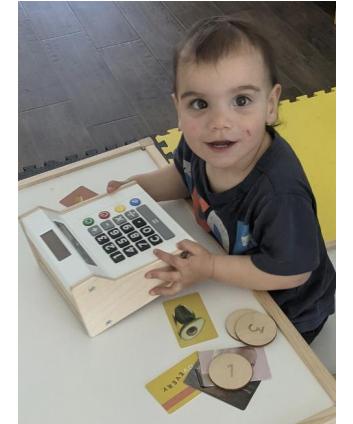
- VGG16 as a feature extraction



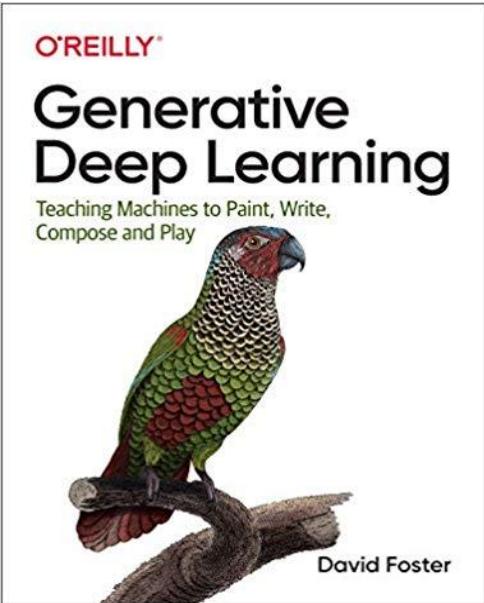
# Generalizable ML to Multicenter Datasets

---

- Experiment 1: common transfer learning
- Experiment 2: domain adaptation using adversarial networks
  - Adapt domains in the feature/latent space
- Experiment 3: domain adaptation using gradient reversal



# References



## Adversarial Discriminative Domain Adaptation

Eric Tzeng  
University of California, Berkeley  
etzeng@eecs.berkeley.edu

Kate Saenko  
Boston University  
saenko@bu.edu

Judy Hoffman  
Stanford University  
jhoffman@cs.stanford.edu

Trevor Darrell  
University of California, Berkeley  
trevor@eecs.berkeley.edu



A screenshot of a web browser showing a tutorial on Generative Adversarial Networks (GANs). The URL in the address bar is 'developers.google.com/machine-learning/gan'. The page has a dark header with the text 'Quebec Artificial Intelligence Institute' and the Mila logo. The main content area is titled 'Generative Adversarial Networks' and includes sections like 'Crash Course', 'Problem Framing', 'Data Prep', 'Clustering', 'Recommendation', 'Testing and Debugging', and 'GANs'. The 'GANs' section is currently selected. It contains an 'Introduction' video thumbnail featuring a man speaking. Below the video, there are four generated faces: a woman with red hair, a woman with blonde hair, a man with dark hair, and a man with light hair. The caption below the images reads 'Figure 1: Images generated by a GAN created by NVIDIA.' A note explains that GANs achieve realism by pairing a generator and a discriminator.

# Thank you!

---

# 7 JORGES TODAY!

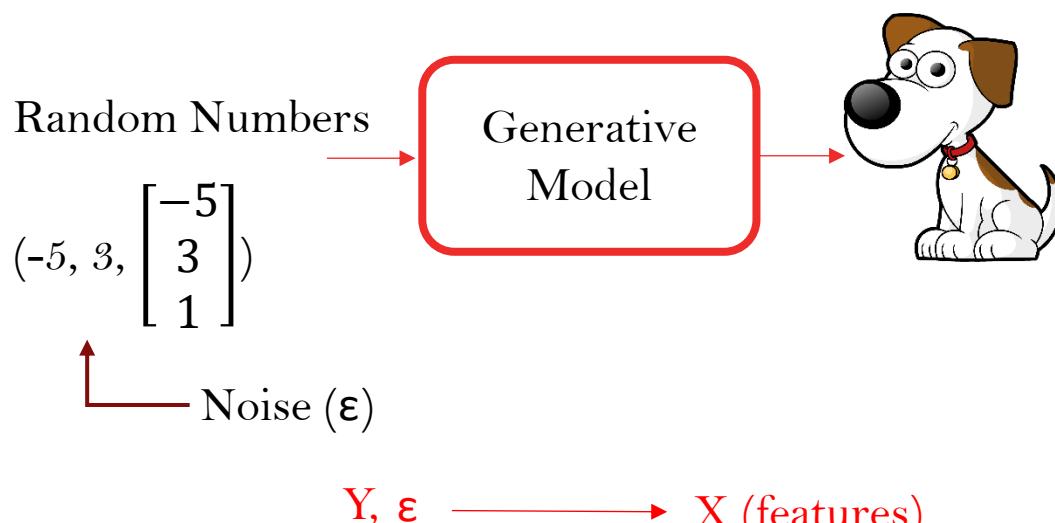
---



# Generative vs Discriminate models

- Generative Models

- Generate realistic representation for each class.



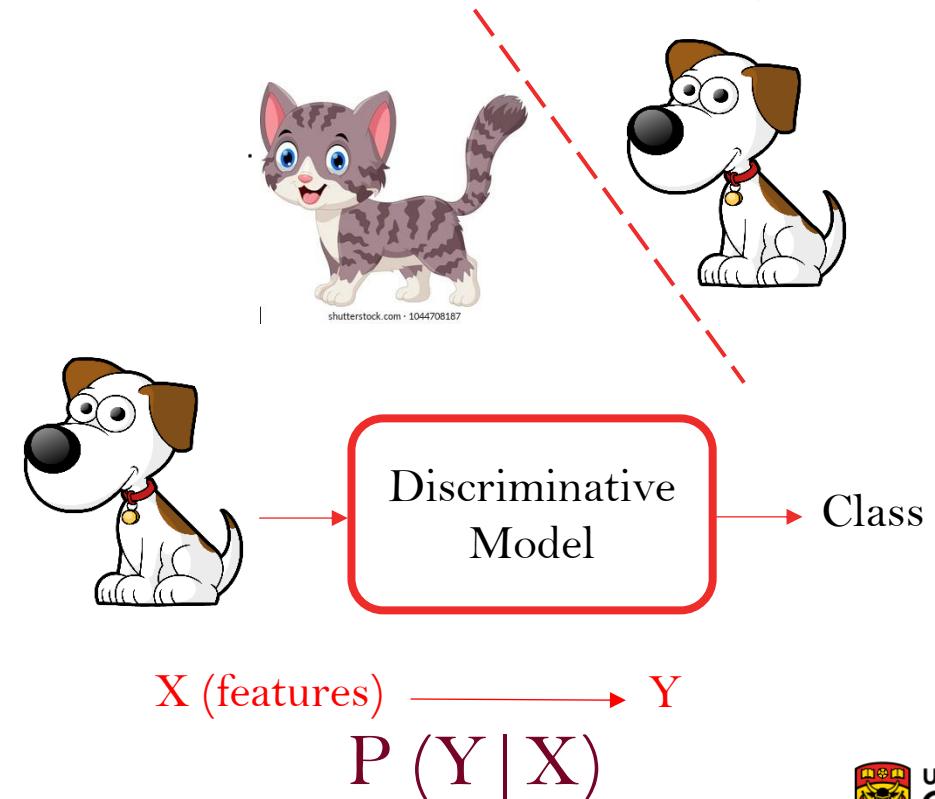
$$P(X|Y)$$

or

$$P(X)$$

- Discriminative Models

- Used for classification problem

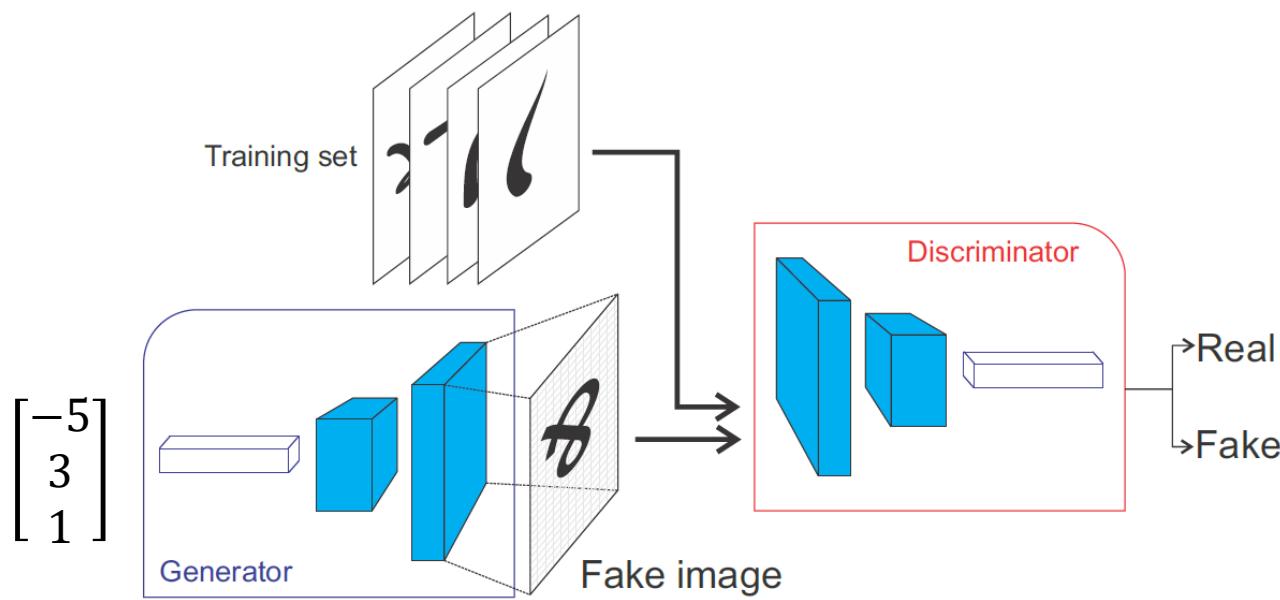


# GAN

To produce Realistic Presentation of different classes

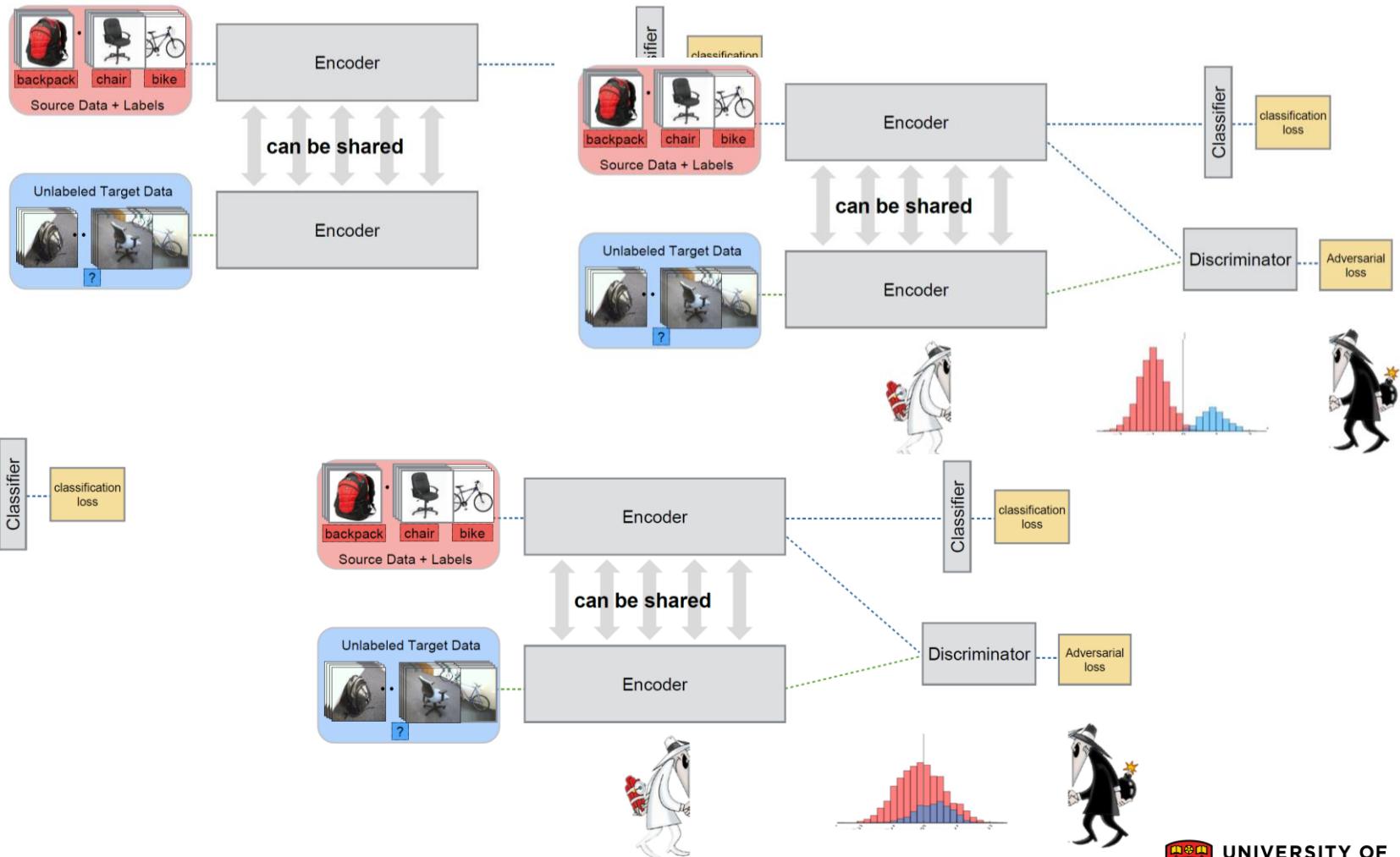
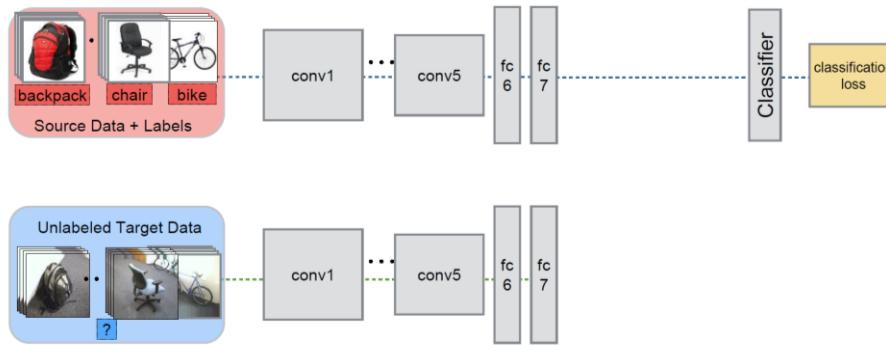


To distinguish real images from fake ones (produced by generator)



# Domain adaptation using adversarial networks

## Adversarial networks



# How do GANs work?

- Notice that the discriminator and generator are trained **iteratively**.

ned

Maximize

Minimize



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

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

**end for**

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

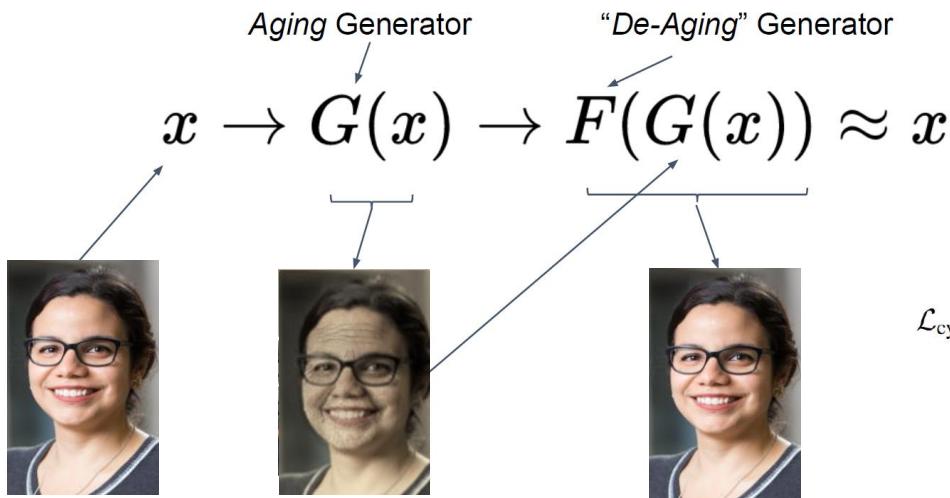
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(z^{(i)}))).$$

**end for**

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

# CycleGAN

- CycleGan introduced the concept of **cycle-consistency** loss to their GANs. This allowed training generators that would unpaired images from domain X to domain Y



$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)} [\|G(F(y)) - y\|_1].$$

<https://arxiv.org/pdf/1703.10593.pdf>  
<https://junyanz.github.io/CycleGAN/>

<https://arxiv.org/pdf/1703.10593.pdf>  
<https://junyanz.github.io/CycleGAN/>