Machine Learning

Prof. Adil Khan

Objectives

1. A quick recap of last week
2. What are Generative Models? How are they different from Discriminative Models? What are some applications of Generative Models?
3. Generative Adversarial Networks
   - What is Adversarial Learning?
   - What is a Generator? What is its objective? How is it motivated?
   - What is a Discriminator? What is its objective? How is it motivated?
   - What is the overall objective of GANs?
   - How are GANs trained?
4. Some important concepts related to GANs
5. Some important variants of GANs

Recap (1)

SPORTS

WORLD NEWS

Unsupervised Learning

- We have only predictors (a.k.a inputs, or features) but no labels

$$\mathbb{D} = \{(x_i)|x_i \in \mathbb{R}^p\}_{i=1}^m$$

The **goal of unsupervised learning** is to model the hidden patterns or underlying structure in the given input data in order to learn about the data

- This *underlying structure* is what we usually refer to as *groups* of data
- And these groups are what we refer to as ***Clusters***

Clusters

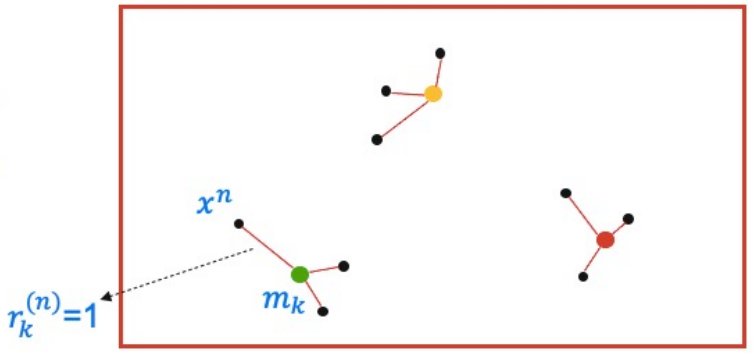
- Clusters are defined by a center and a spread
- And we assign a given point x^n to a cluster by computing its similarity to the center of the cluster m_k

Recap (2)

Objective

- Find cluster centers $\{\boldsymbol{m}_k\}_{k=1}^K$ and assignments $\{\boldsymbol{r}^{(n)}\}_{n=1}^N$ to minimize the sum of squared distances of data points $\{\boldsymbol{x}^{(n)}\}$ to their assigned centers

$\boldsymbol{x}^{(n)} \in \mathbb{R}^D$
$\boldsymbol{m}_k \in \mathbb{R}^D$
$\boldsymbol{r}^{(n)} \in \mathbb{R}^K$



$\boldsymbol{r}^{(n)} = [0, \cdots, 1, \cdots, 0]^T$

$$\min_{\{\mathbf{m}_k\},\{\mathbf{r}^{(n)}\}} J(\{\mathbf{m}_k\},\{\mathbf{r}^{(n)}\}) = \min_{\{\mathbf{m}_k\},\{\mathbf{r}^{(n)}\}} \sum_{n=1}^{N}\sum_{k=1}^{K} r_k^{(n)} ||\mathbf{m}_k - \mathbf{x}^{(n)}||^2$$

where $r_k^{(n)} = \mathbb{I}[\mathbf{x}^{(n)}$ is assigned to cluster $k]$, i.e., $\mathbf{r}^{(n)} = [0, .., 1, .., 0]^\top$

K-means Clustering

- Initialization: randomly initialize cluster centers
- The algorithm then iteratively alternates between the following two steps
  - Assignment Step: assign each data point to the closest center
  - Refitting Step: move each cluster center to the mean of the data points assigned to it

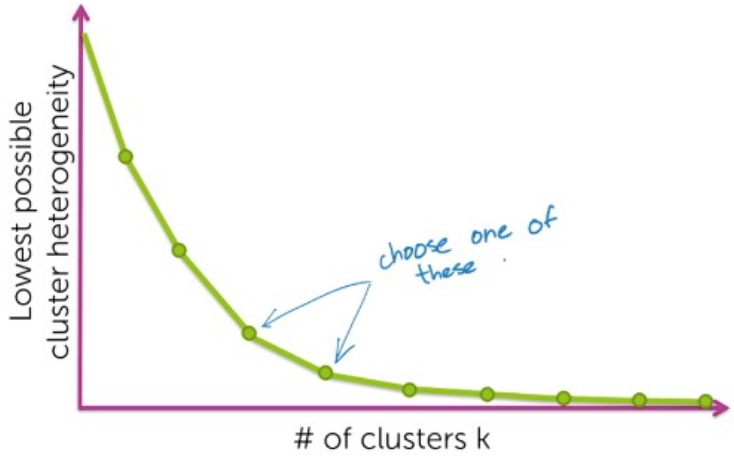
The objective is solved by alternating minimization

Recap (3)

k-means is sensitive to center initialization and we can get completely different solutions, by converging to a local mode

It can be improved by using a specific way of choosing centers

Choosing K – The Elbow Method



K-means++

Issues with K-means

1. Not knowing the optimum value of K
2. Also, k-means clustering does not work well when
   - We want to discover clusters of varying sizes, densities and shapes
   - We do not want to include noisy points (outliers) into any clusters

Recap (4)

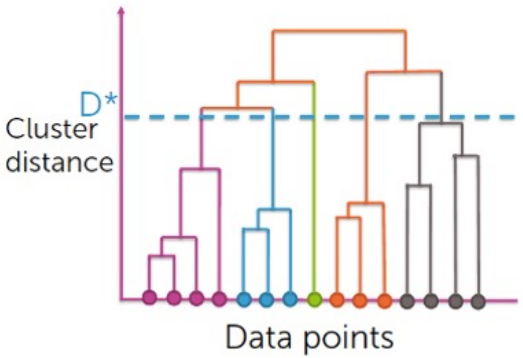
Hierarchical Clustering

- Top-down or Divisive Clustering
  - Start with all items in one cluster,
  - Split recursively
- Bottom-up or Agglomerative Clustering
  - Start with singeltons,
  - Merge clusters using some criteria

Top-down

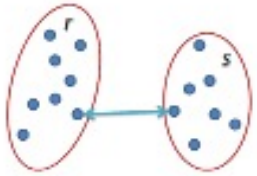
- Simple solution: run k-means recursively

Dendrogram



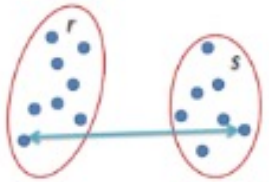
Bottom-up

- Merging clusters:
  1. Single linkage:



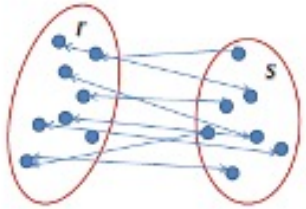
$$L(r,s) = \min(D(x_{ri}, x_{sj}))$$

  2. Complete linkage



$$L(r,s) = \max(D(x_{ri}, x_{sj}))$$

  3. Average linkage

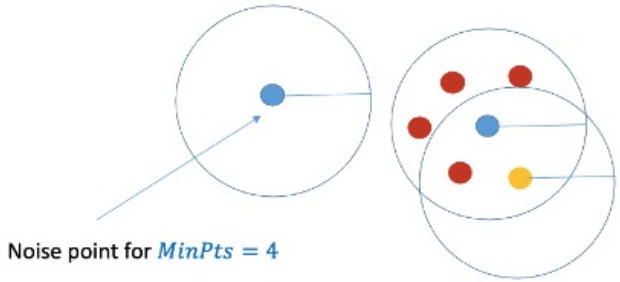


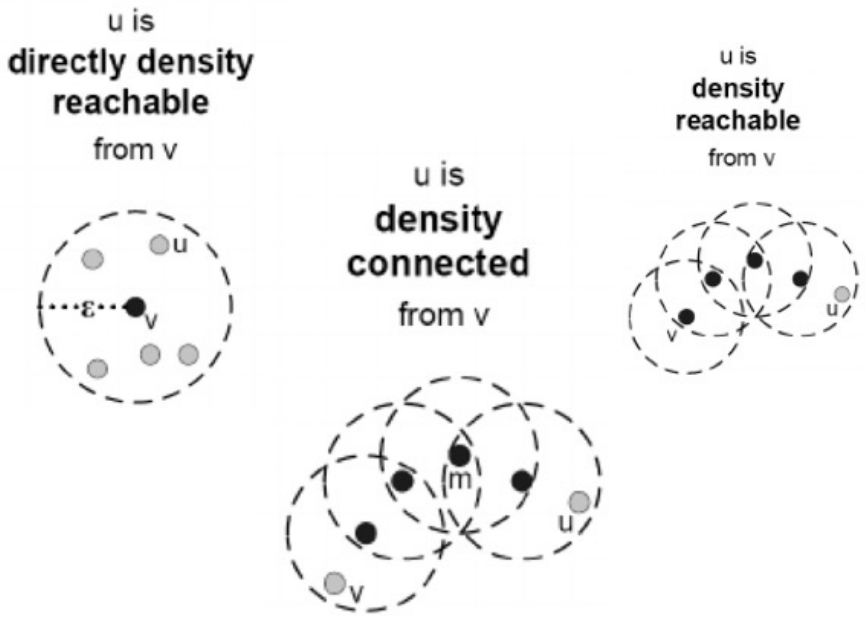
$$L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

Recap (5)

DBSCAN

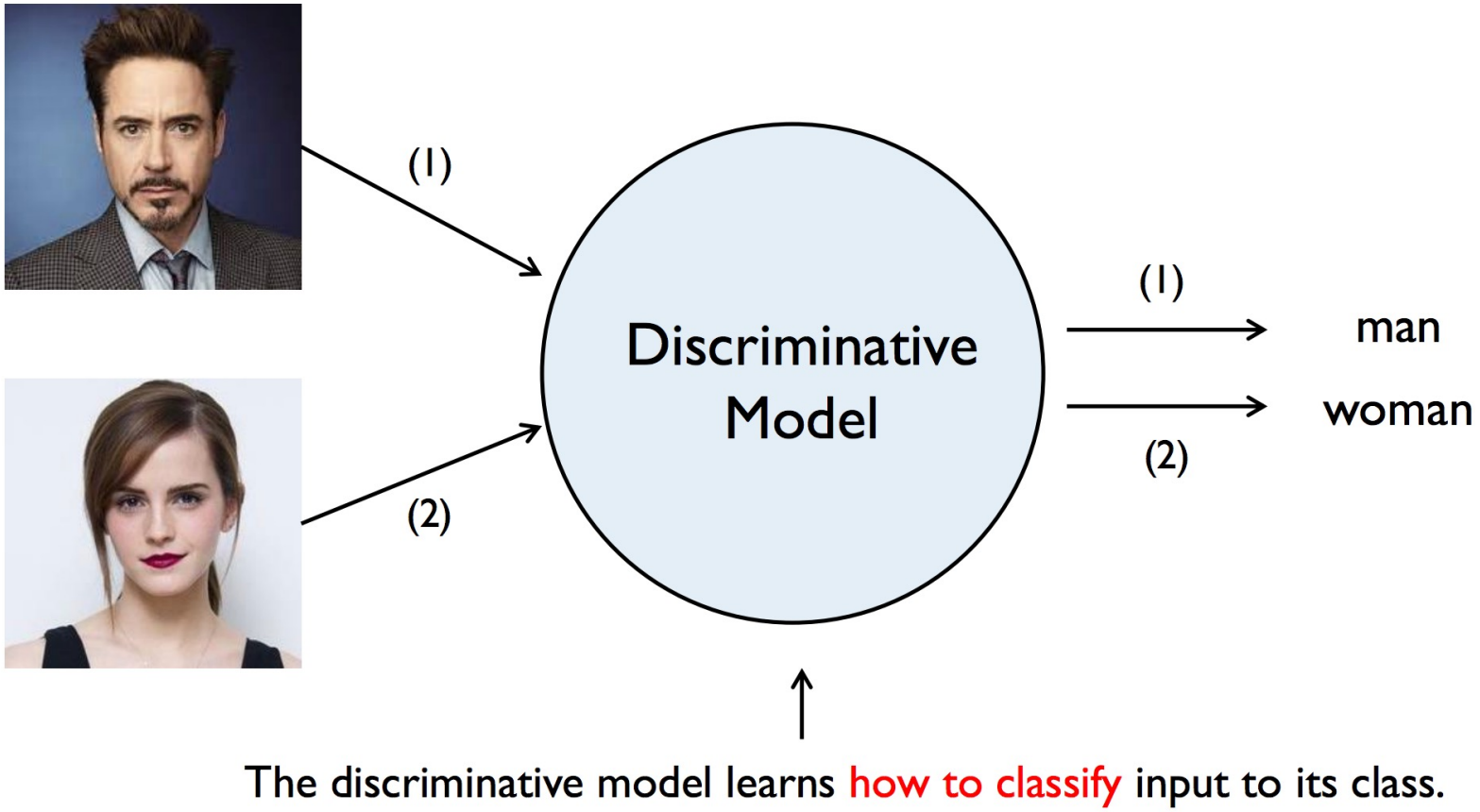
- Neighborhood of point p
- Number of points inside the neighborhood of p
- Core points
- Border points
- Noise points
- Reachability





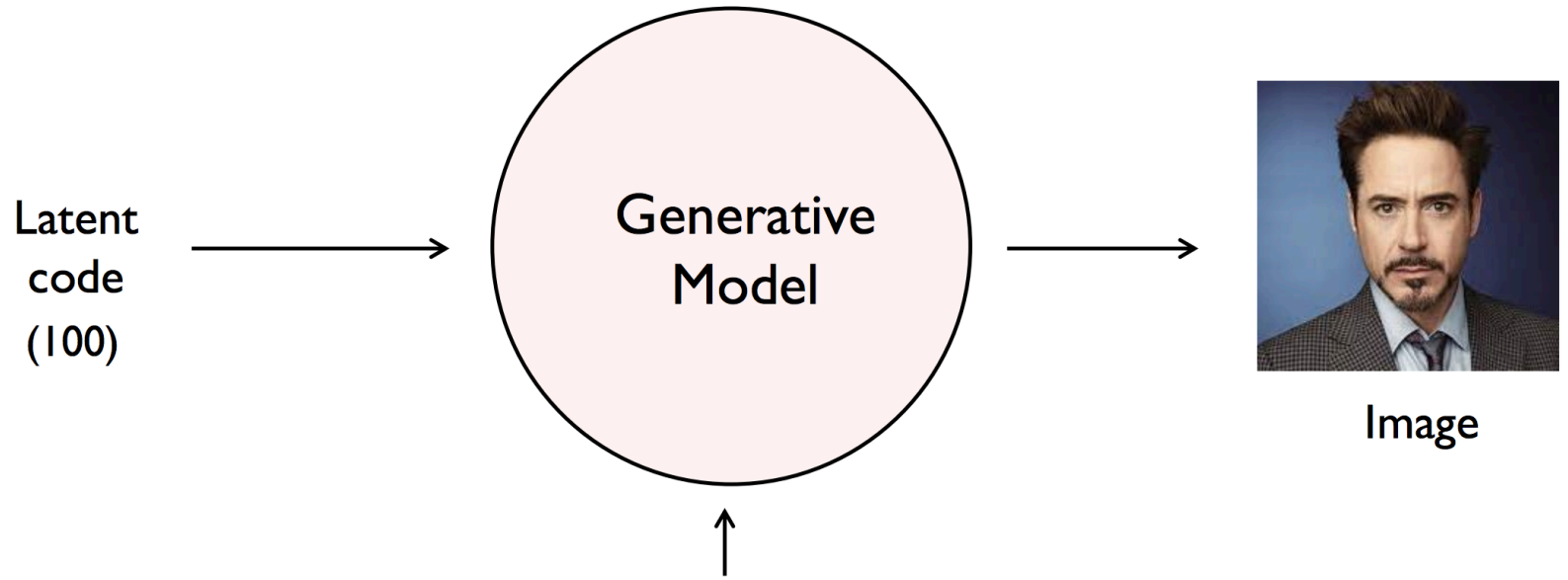
Generative Models

Discriminative Model



Source: Yunjey Choi, Korea University

Generative Model



Source: Yunjey Choi, Korea University

Data Distribution

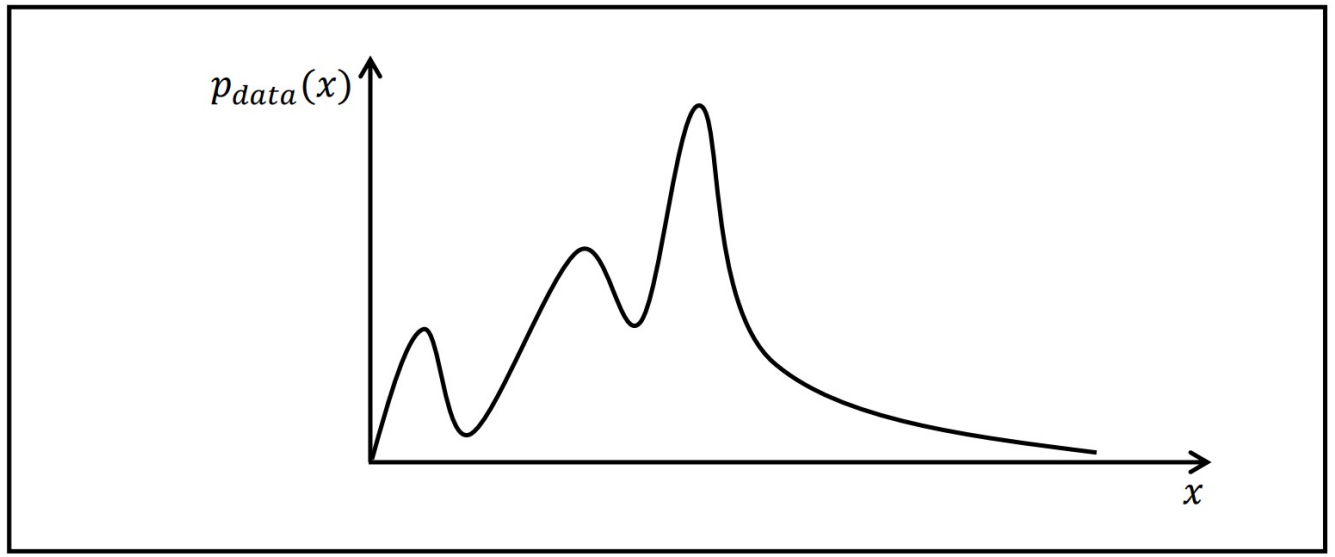
- Let x be a training image, which can be represented as a vector (e.g. 64x64x3 dimensional vector)
- Our dataset is a collection of such images

Source: Yunjey Choi, Korea University

Data Distribution (2)

Probability density function

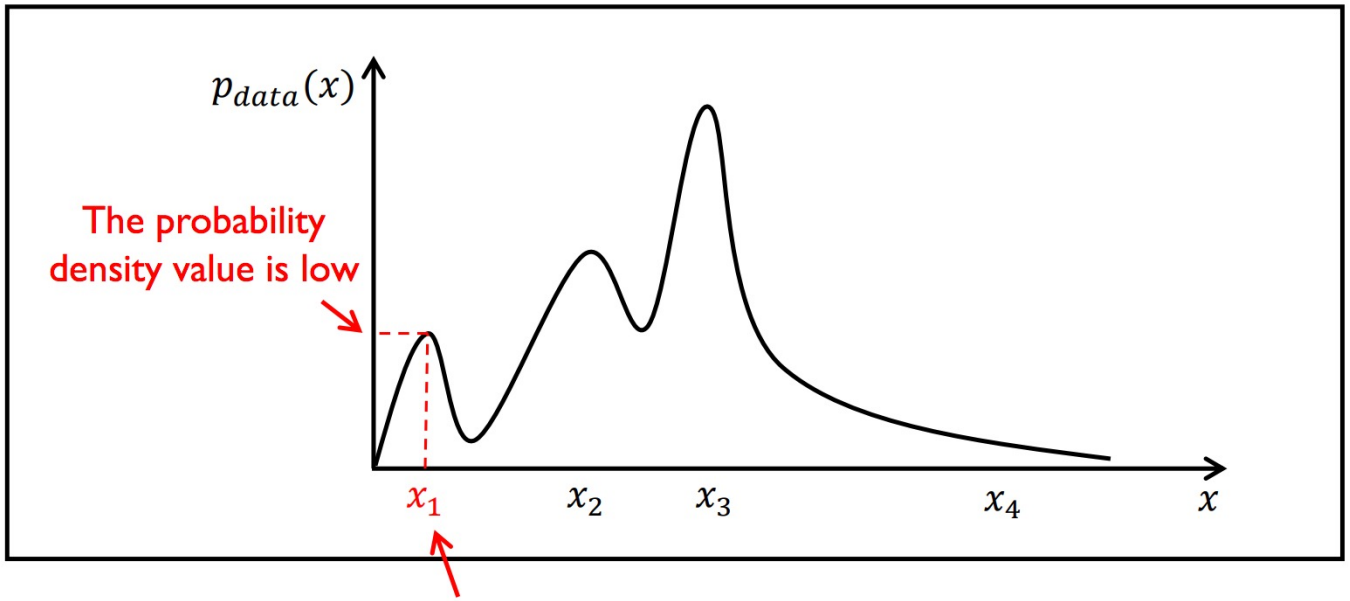
There is a $p_{data}(x)$ that represents the distribution of actual images.



Source: Yunjey Choi, Korea University

Data Distribution (3)

Our dataset may contain few images of men with glasses.

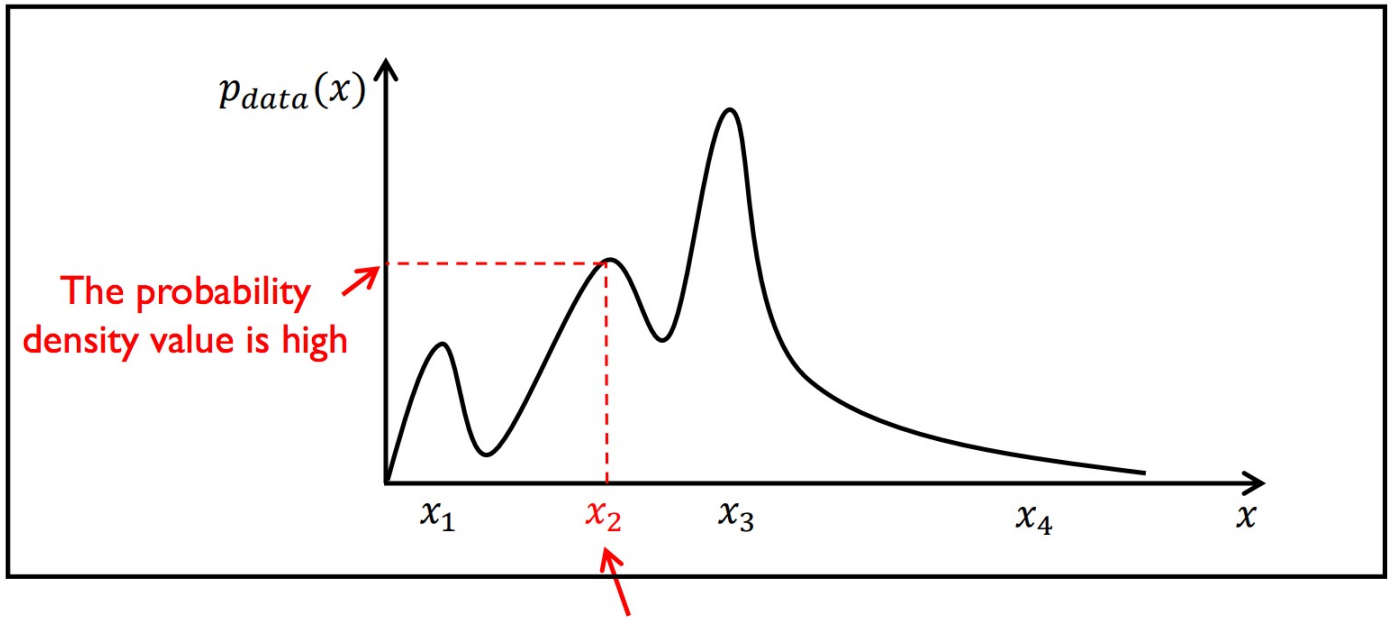


x_1 is a 64x64x3 high dimensional vector representing a man with glasses.

Source: Yunjey Choi, Korea University

Data Distribution (4)

Our dataset may contain many images of women with black hair.

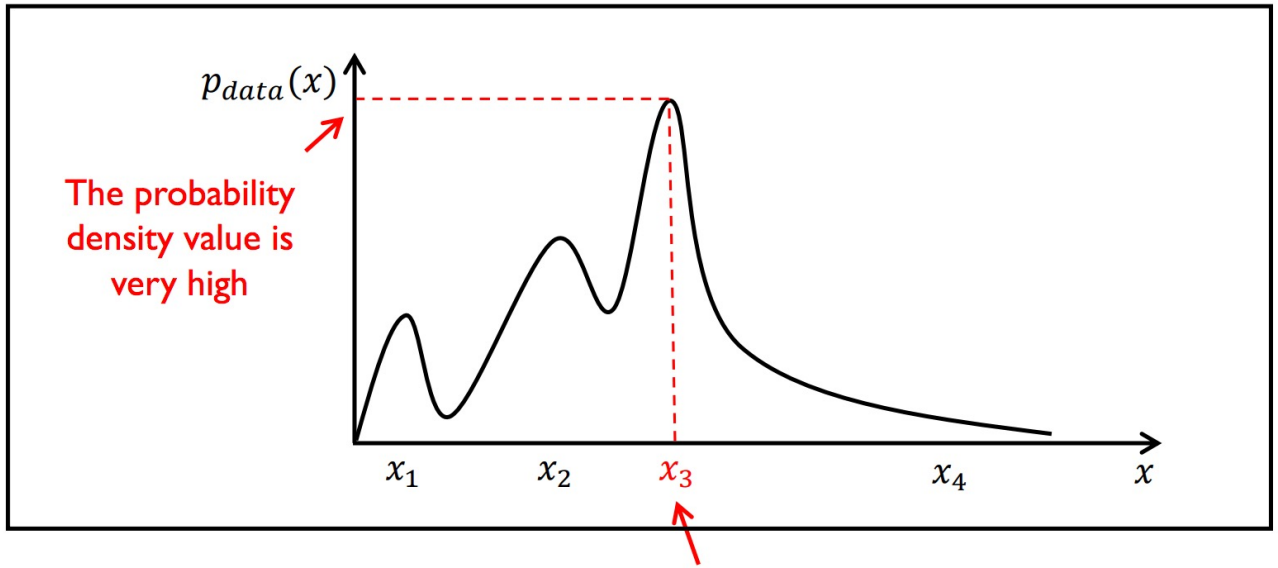


x_2 is a 64x64x3 high dimensional vector representing a woman with black hair.

Source: Yunjey Choi, Korea University

Data Distribution (5)

Our dataset may contain very many images of women with blonde hair.

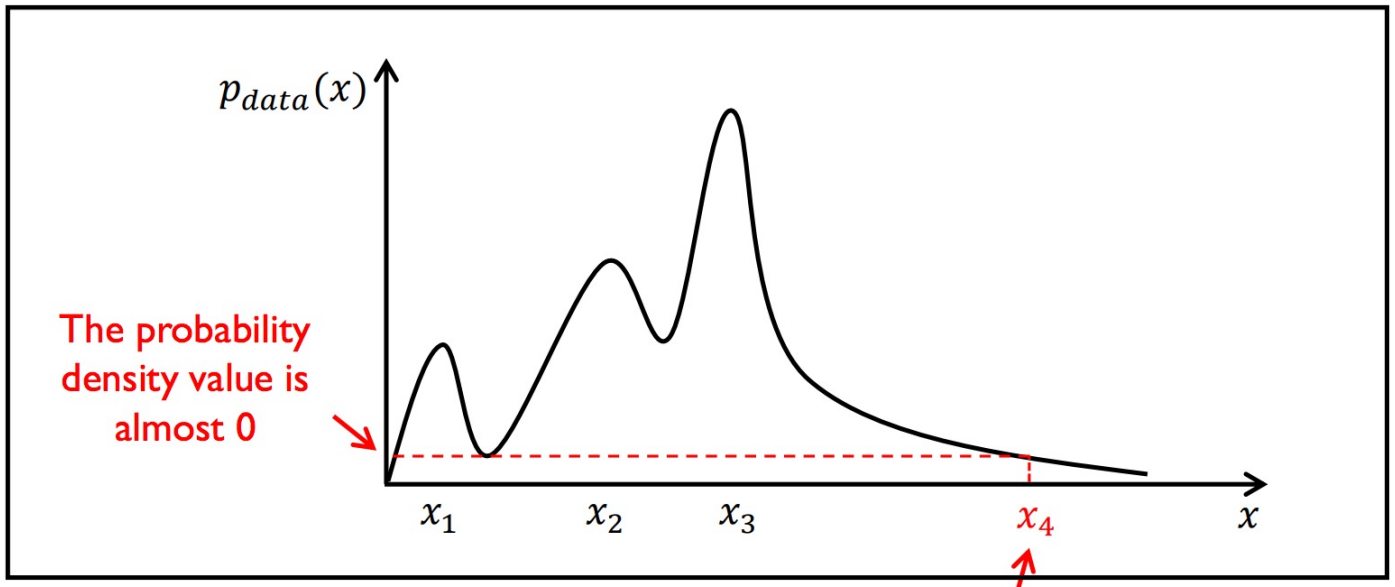


x_3 is a 64x64x3 high dimensional vector representing a woman with blonde hair.

Source: Yunjey Choi, Korea University

Data Distribution (6)

Our dataset may not contain these strange images.



x_4 is an 64x64x3 high dimensional vector representing very strange images.

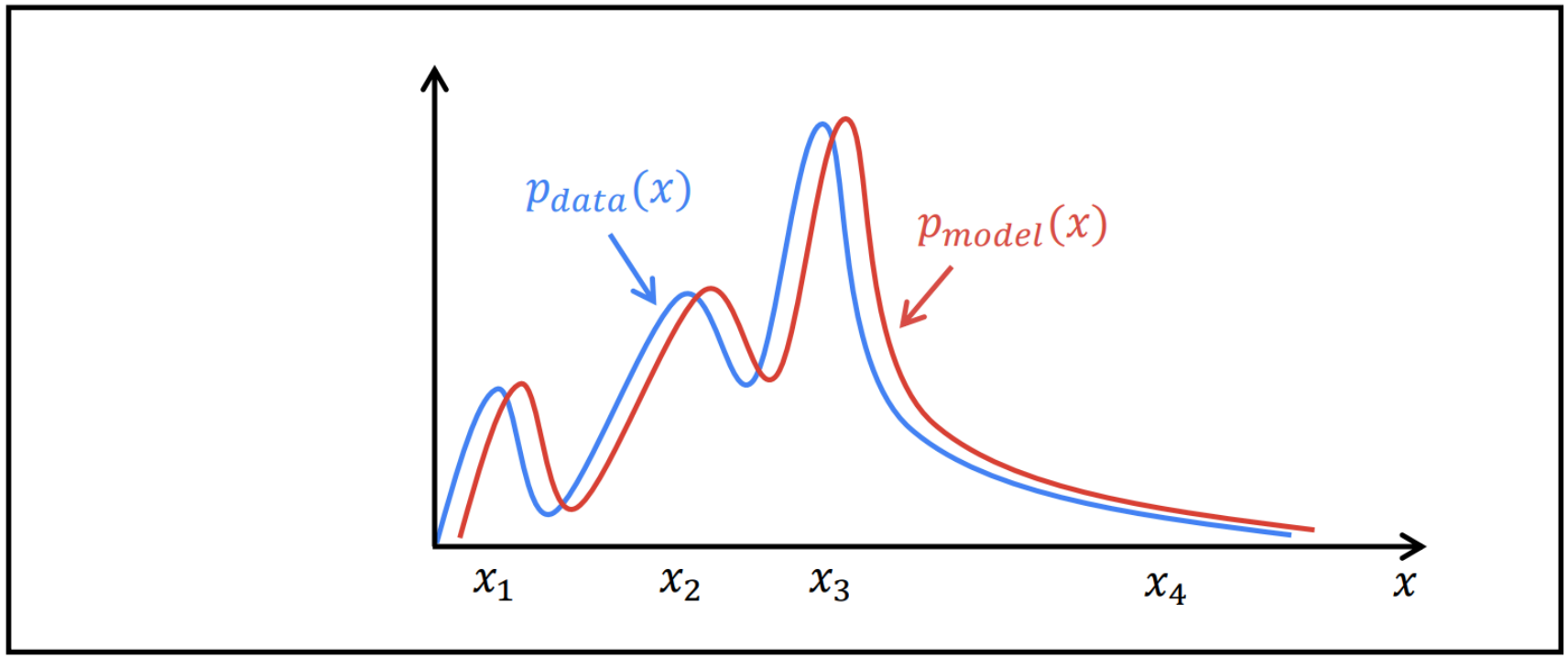
Source: Yunjey Choi, Korea University

Generative Modeling

$p_{model}(x)$ — *Distribution of images generated by the model*

The goal of the generative model is to find a $p_{model}(x)$ that approximates $p_{data}(x)$ well.

$p_{data}(x)$ — *Distribution of actual images*



Generative Modeling (2)

1. We have training samples from an unknown distribution p_{data}
2. We want a model that can generate (or draw) samples from some distribution p_{model}
3. p_{model} should be an estimate of p_{data}
4. A model that can sample from this p_{model} is termed as a ***generative model***

GANs are an example of such a model

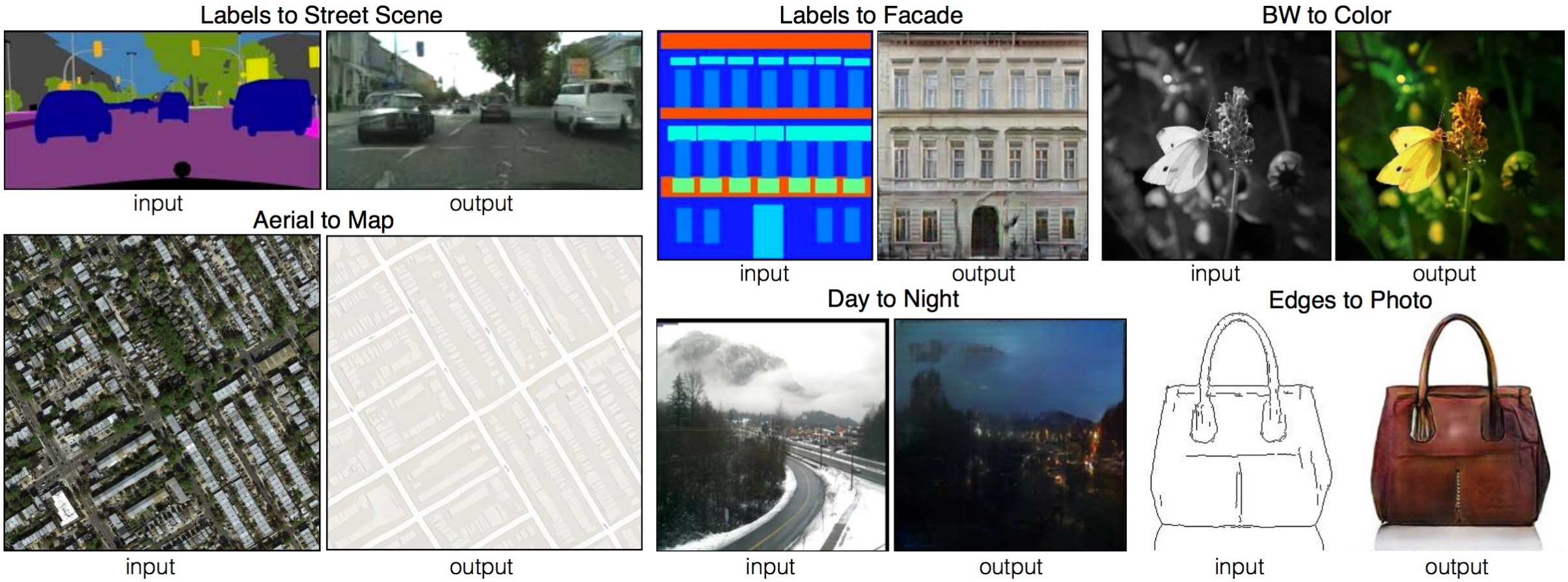
Why Generative Models?

“What I cannot create, I do not understand.”

—Richard Feynman

Applications of Generative Models

Image-to-Image Translation



Source: https://arxiv.org/abs/1611.07004

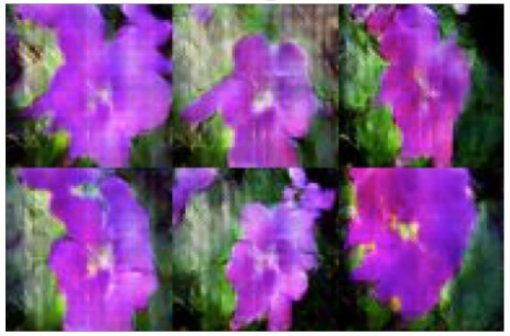
Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. “Image-to-image translation with conditional adversarial networks”. arXiv preprint arXiv:1611.07004. (2016).

Text-to-Image

this small bird has a pink breast and crown, and black primaries and secondaries.

this magnificent fellow is almost all black with a red crest, and white cheek patch.

the flower has petals that are bright pinkish purple with white stigma

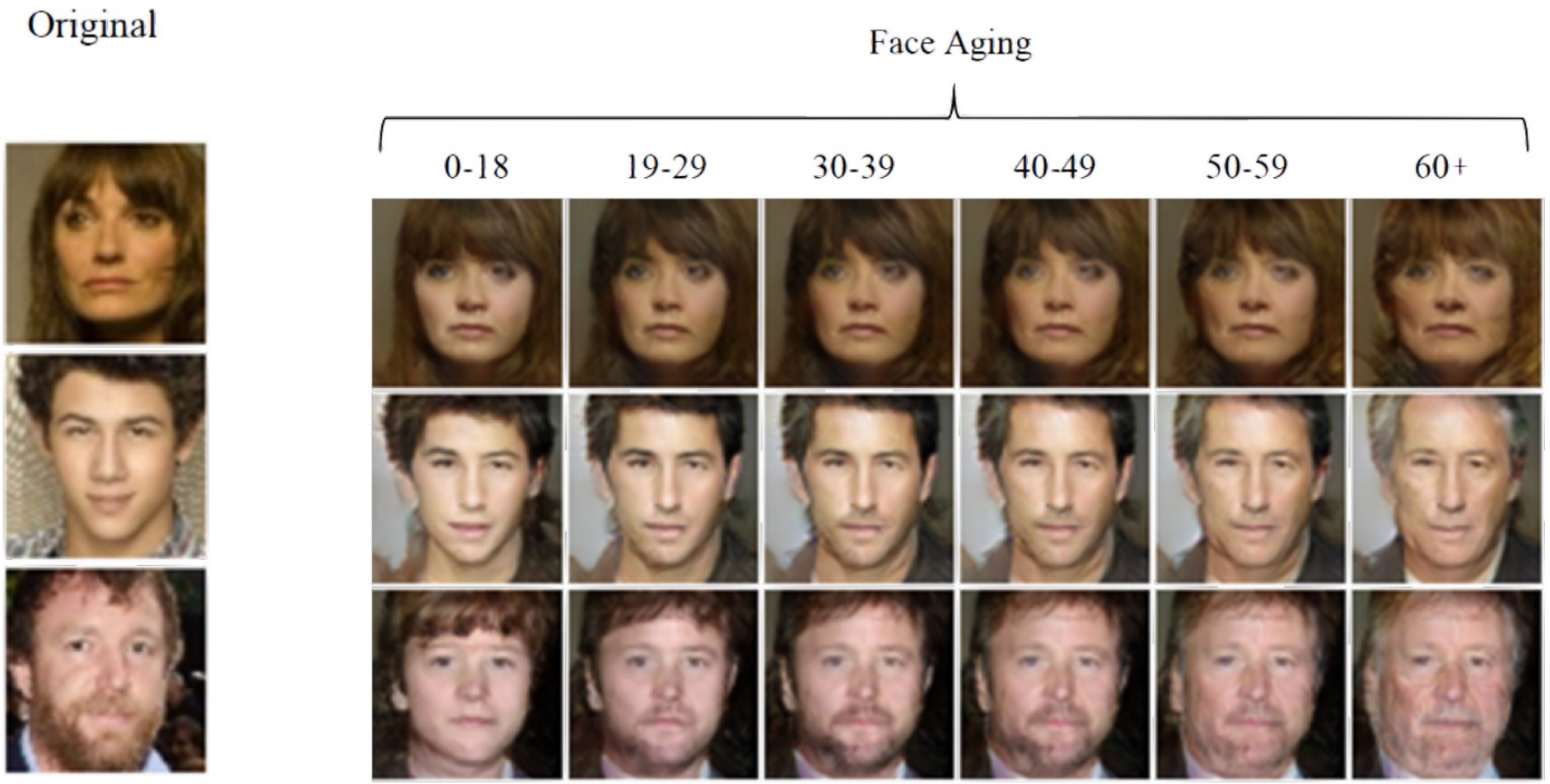


this white and yellow flower have thin white petals and a round yellow stamen

Source: https://arxiv.org/abs/1605.05396

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. “Generative adversarial text to image synthesis”. ICML (2016).

Face-Aging



Antipov, G., Baccouche, M., & Dugelay, J. L. (2017). “Face Aging With Conditional Generative Adversarial Networks”. arXiv preprint arXiv:1702.01983.

Image Inpainting

Ugur Demir and Gozde Unal. Patch-Based Image Inpainting with Generative Adversarial Networks. *arXiv preprint*, 2018. URL http://arxiv.org/abs/1803.07422.

Art Creation

Ahmed Elgammal, Bingchen Liu, Mohamed Elhoseiny, and Marian Mazzone. CAN: Creative Adversarial Networks, Generating ”Art” by Learning About Styles and Deviating from Style Norms. *arXiv preprint*, (Iccc):1–22, 2017. doi: 10.1089/cyber.2017.29084.csi. URL http://arxiv.org/abs/1706.07068.

Reparing Software Vulnerabilities

Table 2: Successful Repairs: (Top) This function calls sprintf to print out two strings, but only provides the first string to print. Our GAN repairs it by providing a second string. (Bottom) This function uses a variable again after freeing it. Our GAN repairs it by removing the first free.

With Vulnerability	Repaired

void CWE685_Function_Call_With_Incorrect_ Number_Of_Arguments() {
  char dst[DST_SZ];
  sprintf(dst, "%s %s", SRC_STR);
  printLine(dst);
}
	void CWE685_Function_Call_With_Incorrect_ Number_Of_Arguments() {
	  char dst[DST_SZ];
	  sprintf(dst, "%s %s", SRC_STR, SRC_STR);
	  printLine(dst);
	}

void CWE415_Double_Free___malloc_free_ struct_31() {
  twoints *data;
  data = NULL;
  data = (twoints *)malloc(100 * sizeof(twoints));
  free(data);
  {
  twoints *data_copy = data;
  twoints *data = data_copy;
  free(data);
  }
}
	void CWE415_Double_Free___malloc_free_ struct_31() {
	  twoints *data;
	  data = NULL;
	  data = (twoints *)malloc(100 * sizeof(twoints));
	  {
	  twoints *data_copy = data;
	  twoints *data = data_copy;
	  free(data);
	  }
	}

Harer J, Ozdemir O, Lazovich T, Reale CP, Russell RL, Kim LY, Chin P. Learning to repair software vulnerabilities with generative adversarial networks. arXiv preprint arXiv:1805.07475. 2018 May 18.

Generative Adversarial Networks (GANs)

Generative adversarial networks are deep neural net architectures comprised of two nets, putting one against the other (thus the "adversarial").

Ian Goodfellow
GAN – 2014

Facebook's AI research director Yann LeCun called adversarial training "the most interesting idea in the last 10 years in ML."

Generative Adversarial Networks (GANs)

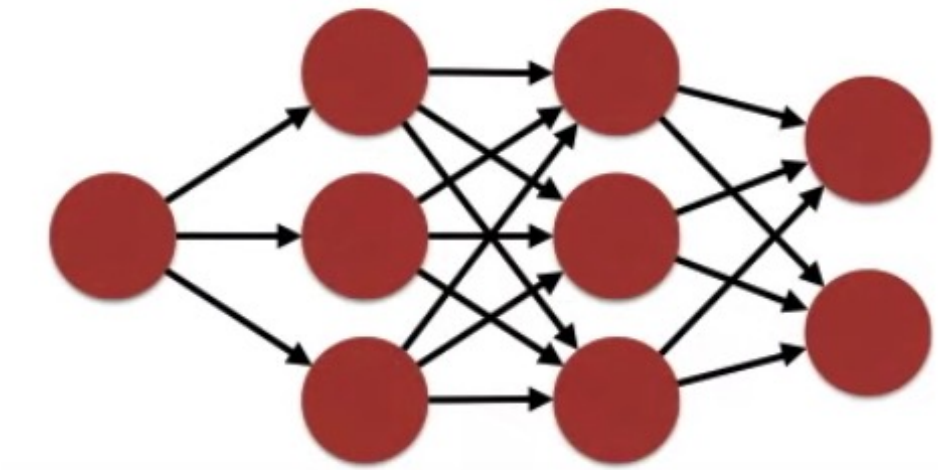
1. Generative
   - Learn a generative model
2. Adversarial
   - Train in an adversarial setting
3. Networks
   - Use deep nerual networks

Components of GANs

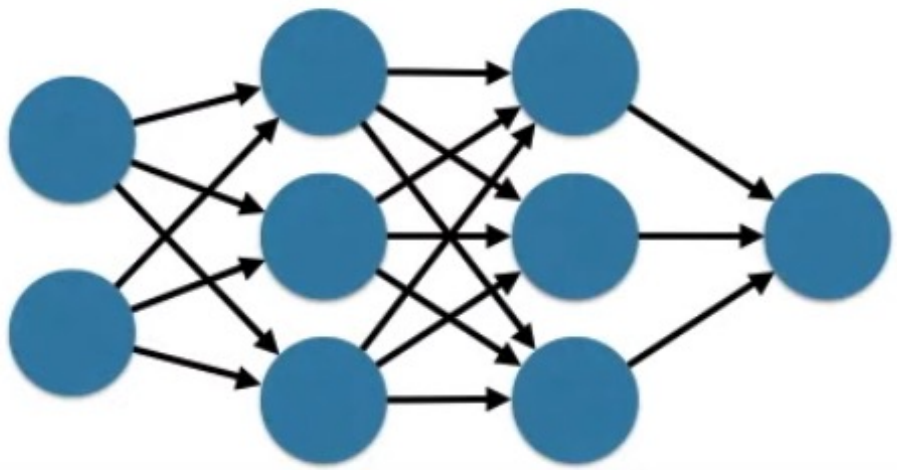
1. A generator function (G) that tries to create real-looking examples
2. A discriminator function (D) that tries to distinguish real from fake examples
3. Functions are updated in a feedback loop, making each better at its task

Like two players in a game: a generator and a discriminator

G and D are Deep Networks

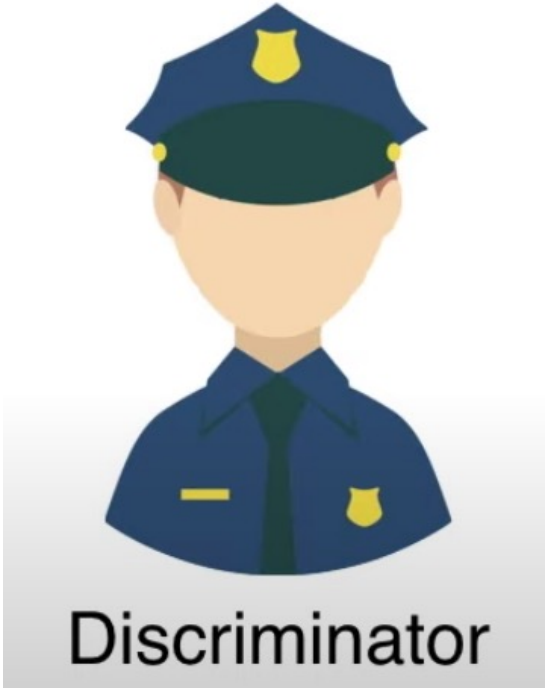


Generator



Discriminator

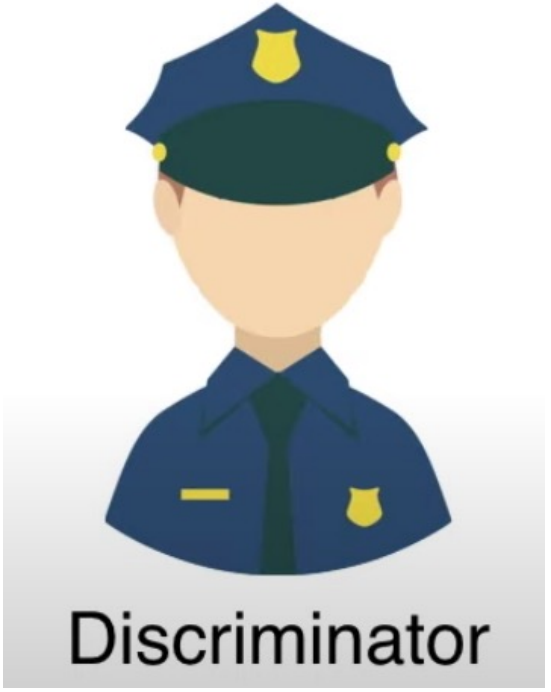
Behave as Counterfiter and a Cop (*Learning from Each Other*)



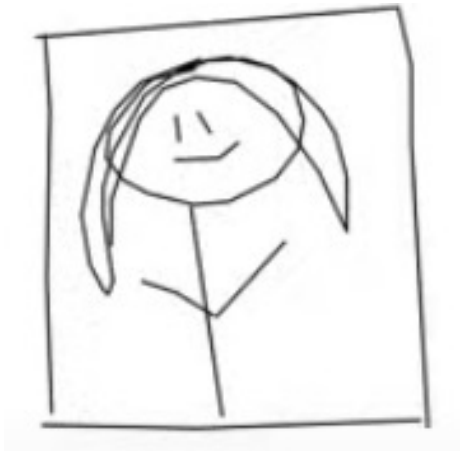
Behave as Counterfiter and a Cop (2)

Ok! I should do better!

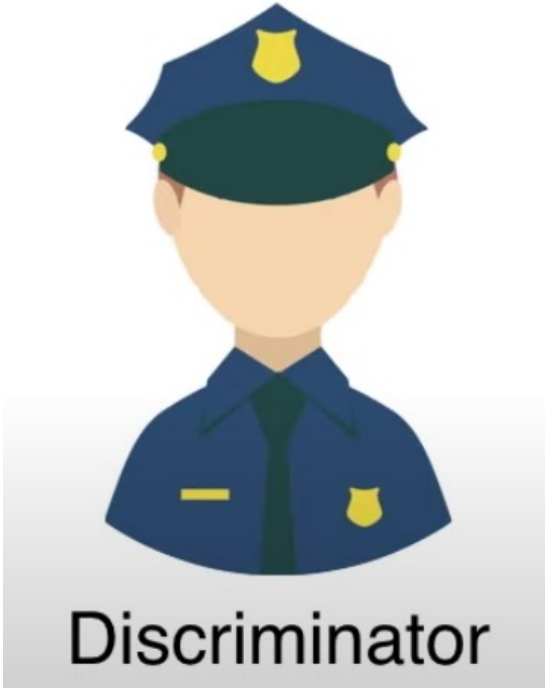
This is Mona Lisa.



Behave as Counterfiter and a Cop (3)



This is Mona Lisa



Ok! I should do even better!!

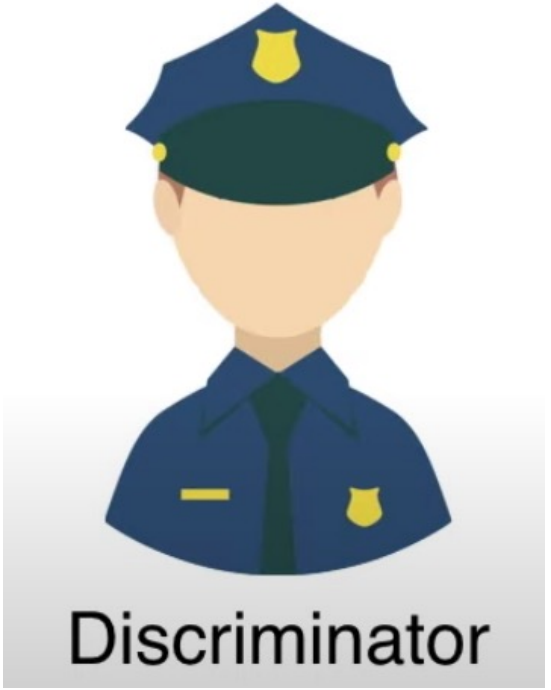
Behave as Counterfiter and a Cop (4)

This is Mona Lisa

Ok! Keep Learning!!

Behave as Counterfiter and a Cop (5)

This is Mona Lisa



Ok! Almost there!!

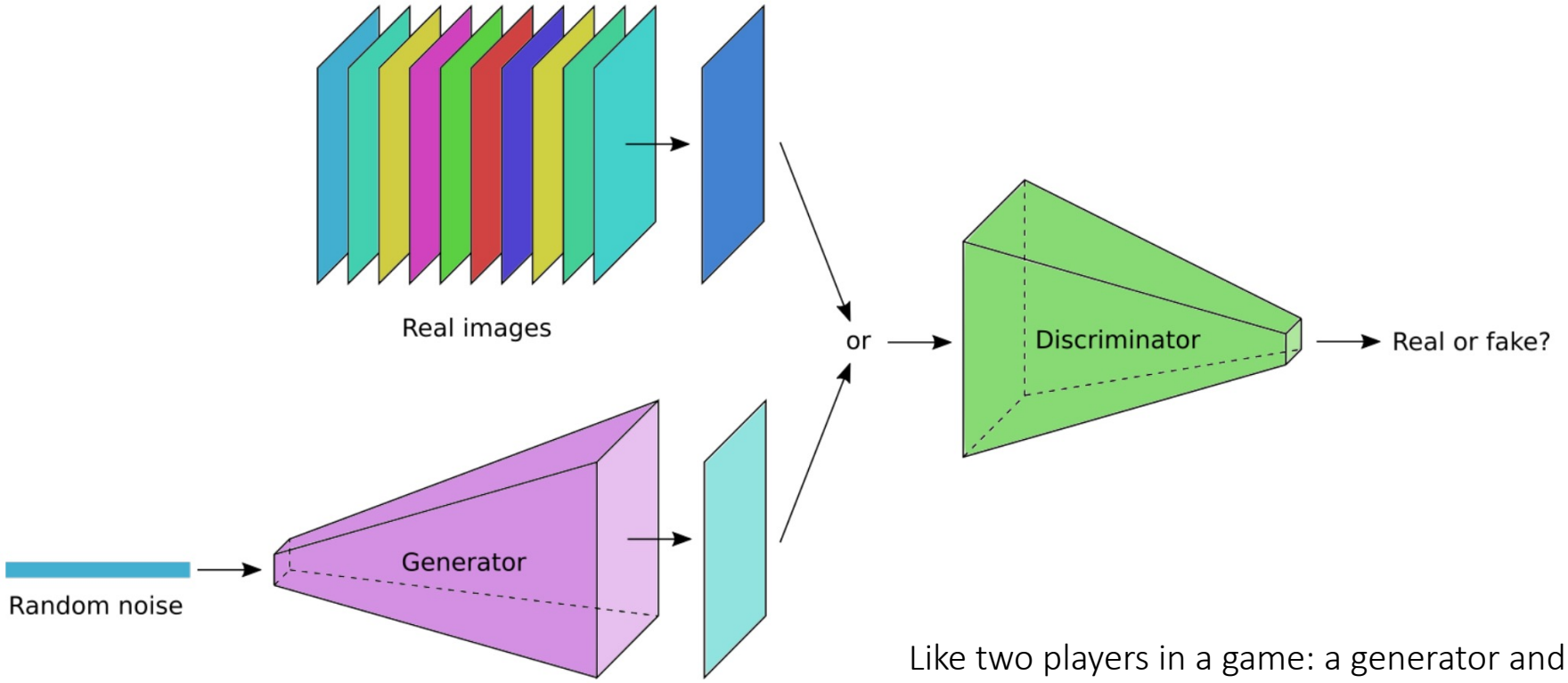
Behave as Counterfiter and a Cop (4)

This is Mona Lisa

Aha!!

Components of GANs

1. A generator function (G) that tries to create real-looking examples
2. A discriminator function (D) that tries to distinguish real from fake examples
3. Functions are updated in a feedback loop, making each better at its task



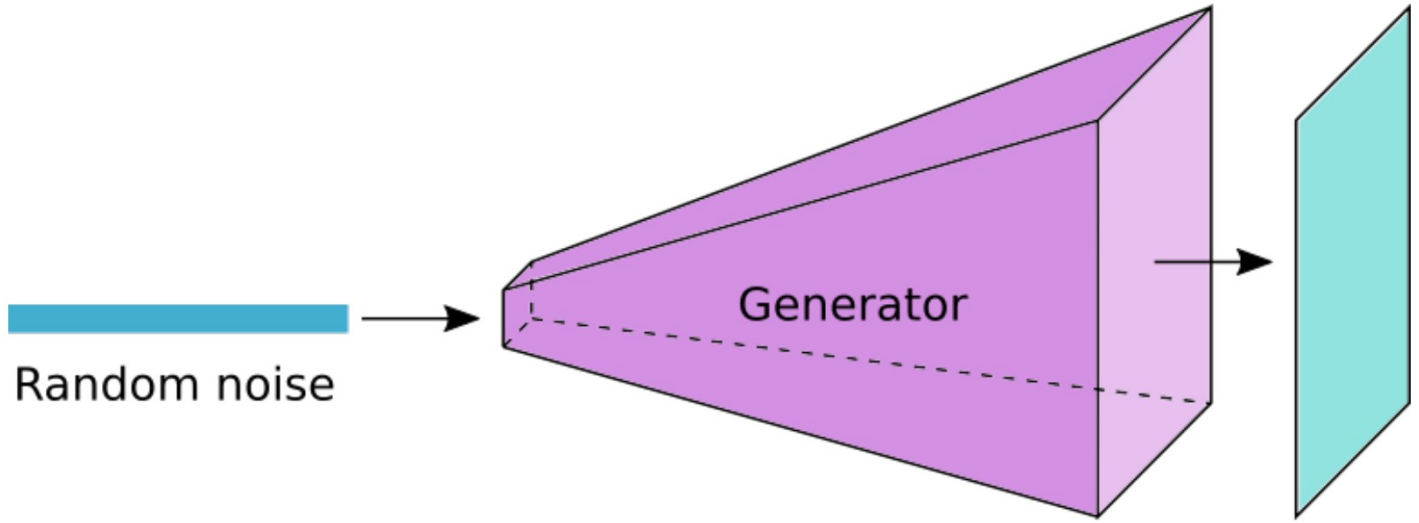
Like two players in a game: a generator and a discriminator

Generator

The generator is a function

$$G\left(z; \boldsymbol{\theta}^{(G)}\right): z \mapsto x$$

- Let G be the generator which is a generative neural network
- The input z is a random vector sampled from some simple prior distribution
- The generator takes as input $z \sim N(0, 1)$ and produces $G(z) = x$

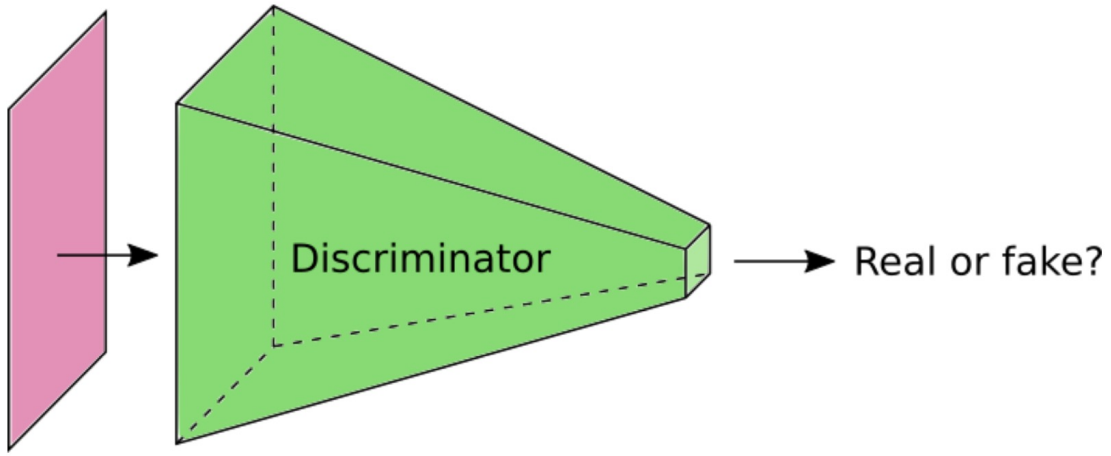


Discriminator

- The discriminator is a standard classification network
- Trained to differentiate between real (x) and fake $(x = G(z))$ images
- Outputs a single number in [0, 1]
  - $D(x) \sim 0 \rightarrow D$ believes x is fake
  - $D(x) \sim 1 \rightarrow D$ believes x is real

The discriminator is a function

$D(x; \boldsymbol{\theta}^{(D)}): x \mapsto [0,1]$



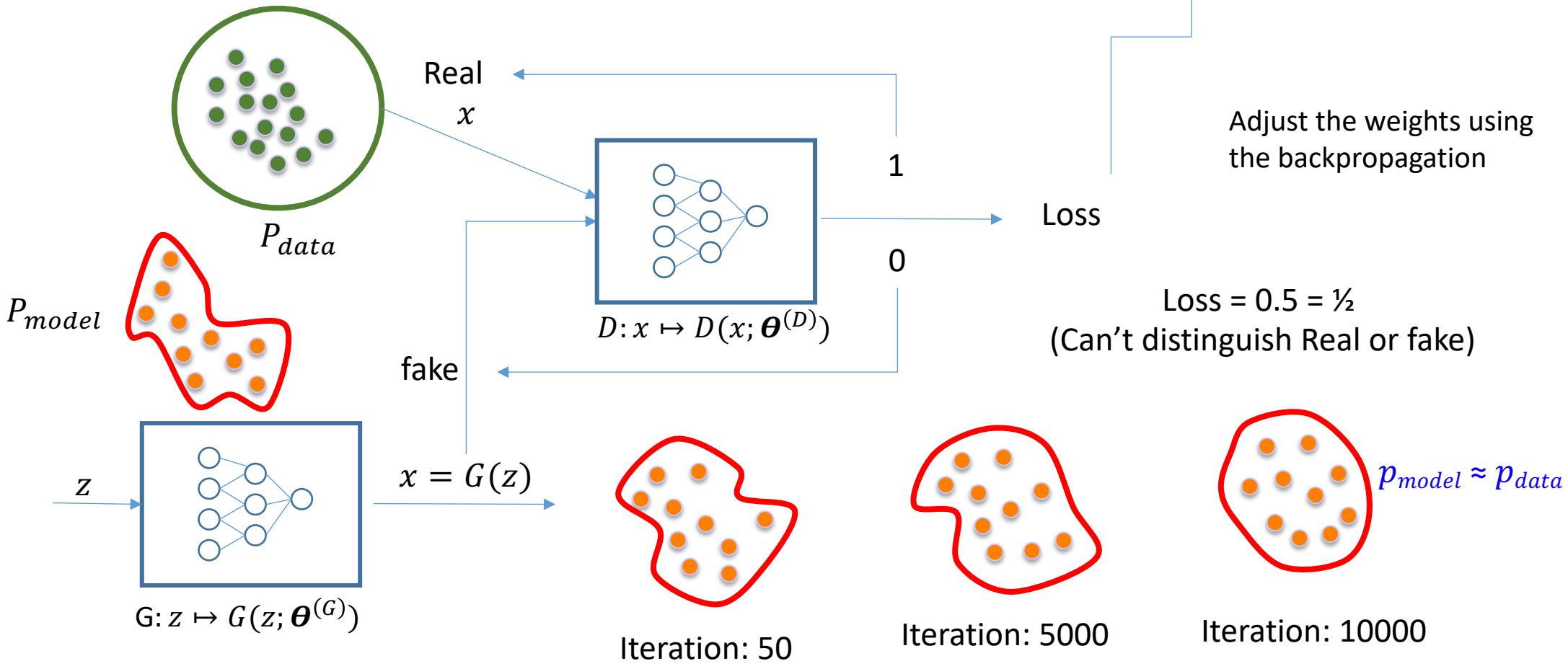
Loss Function of GANs

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

Loss Function of GANs

- Binary Classification
- Binary Cross Entropy

Discriminator Role: distinguish between the real and fake
Generator Role: Create data in such a way that it can fool the discriminator



Source: https://towardsdatascience.com/demystifying-cross-entropy-e80e3ad54a8

Cross-Entropy as a Loss Function

Animal	Label
Dog	[1 0 0 0 0]
Fox	[0 1 0 0 0]
Horse	[0 0 1 0 0]
Eagle	[0 0 0 1 0]
Squirrel	[0 0 0 0 1]

We can treat one hot encoding as a probability distribution for each image. Let's see an examples.

$$
\begin{aligned}
P_1(dog) &= 1 \\
P_1(fox) &= 0 \\
P_1(horse) &= 0 \\
P_1(eagle) &= 0 \\
P_1(squirrel) &= 0
\end{aligned}
$$

Machine Learning Model

$$Q_1 = [0.4 \quad 0.3 \quad 0.05 \quad 0.05 \quad 0.2]$$

$$P_1 = [1 \quad 0 \quad 0 \quad 0 \quad 0]$$

$$H(P_1, Q_1) = -\sum_i P_1(i) \log Q_1(i)$$

$$
\begin{aligned}
&= -(1 \log 0.4 + 0 \log 0.3 + 0 \log 0.05 + 0 \log 0.05 + 0 \log 0.2) \\
&= -\log 0.4 \\
&\approx 0.916
\end{aligned}
$$

The cross-entropy goes down as the prediction gets more and more accurate. It becomes zero if the prediction is perfect.

$$Q_1 = [0.98 \quad 0.01 \quad 0 \quad 0 \quad 0.01]$$

$$H(P_1, Q_1) = -\sum_i P_1(i) \log Q_1(i)$$

$$
\begin{aligned}
&= -(1 \log 0.98 + 0 \log 0.01 + 0 \log 0 + 0 \log 0 + 0 \log 0.01) \\
&= -\log 0.98 \\
&\approx 0.02
\end{aligned}
$$

Binary Cross-Entropy

- In binary classification: we have two classes only.
  - For example: there are only dogs or cats in images.

- Thus cross-entropy formula contains only two probabilities:

$$H(P,Q) = -\sum_{i=(cat,dog)} P(i)\log Q(i)$$
$$= -P(cat)\log Q(cat) - P(dog)\log Q(dog)$$

  - Using the following relationship

$$P(dog) = (1 - P(cat))$$

$$H(P,Q) = -P(cat)\log Q(cat) - (1 - P(cat))\log(1 - Q(cat))$$

$$BinaryCrossEntropy = -P\log\hat{P} - (1 - P)\log(1 - \hat{P})$$

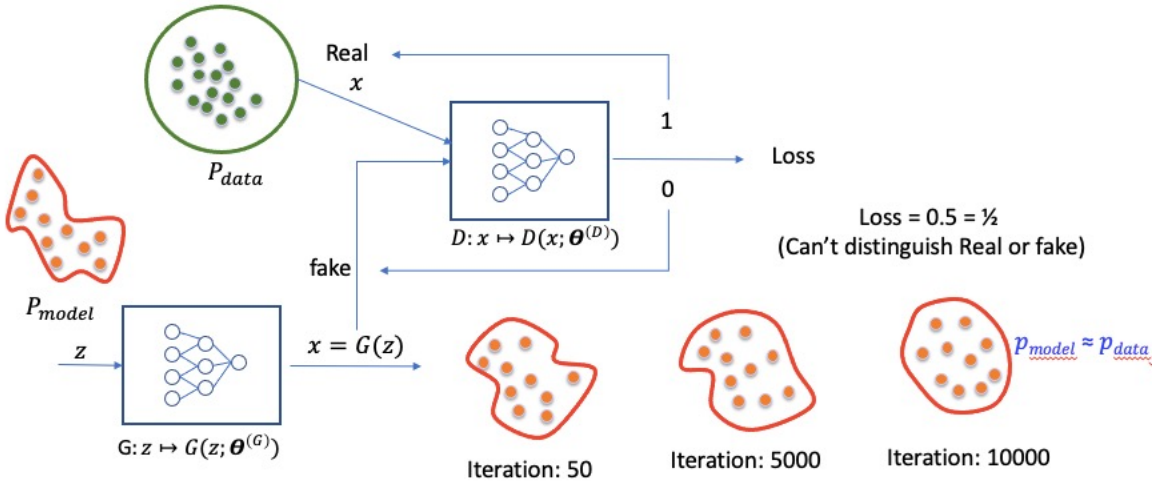
Further simplify

$$P = P(cat)$$
$$\hat{P} = Q(cat)$$

Back to Loss Function of GANs

$$L(Y, \hat{Y}) = [Y \log \hat{Y} + (1 - Y) \log(1 - \hat{Y})]$$

$Y = original$
$\hat{Y} = predicted$



The label for the data coming from $\boldsymbol{P_{data}(x)}$ is $\boldsymbol{y = 1}$ and $\boldsymbol{\hat{y} = D(x)}$, so we can get

$$L(1, D(x)) = [\log D(x)]$$

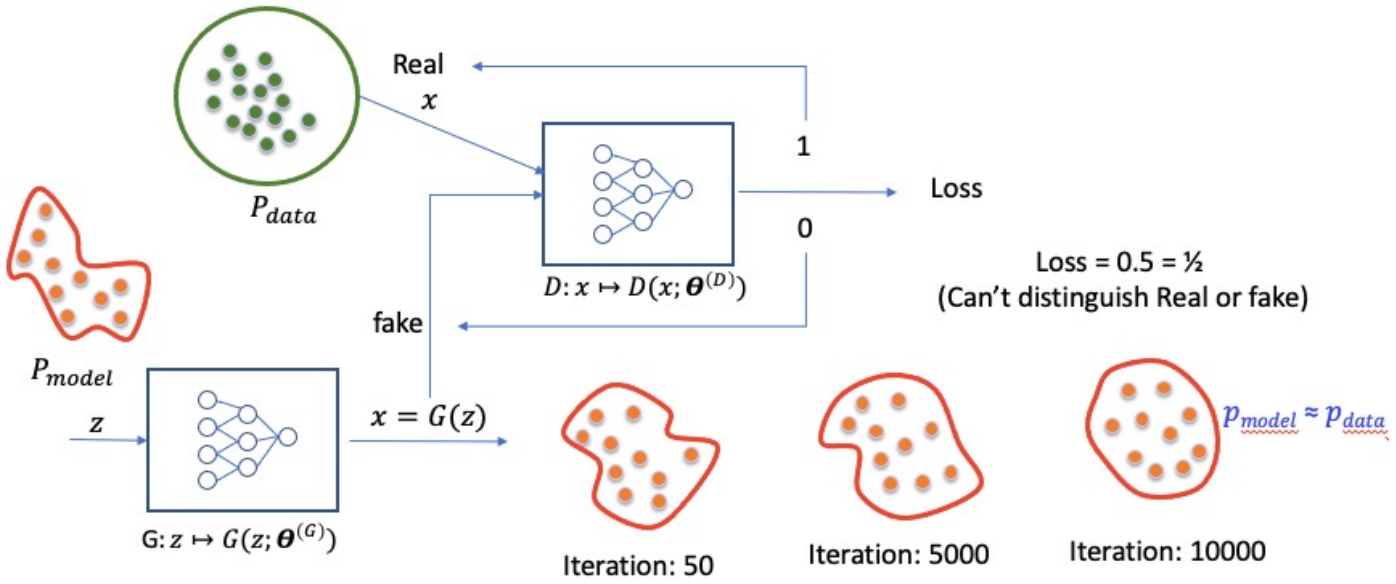
The label for the data coming from $\boldsymbol{P_{model}(z)}$ is $\boldsymbol{y = 0}$ and $\boldsymbol{\hat{y} = D(G(z))}$, so we can get

$$L(0, D(G(z)) = [\log(1 - D(G(z)))]$$

Loss Function of GAN

- Objective of **discriminator** is to correctly classify the fake vs real. So, we need to **maximize** the previous both cases

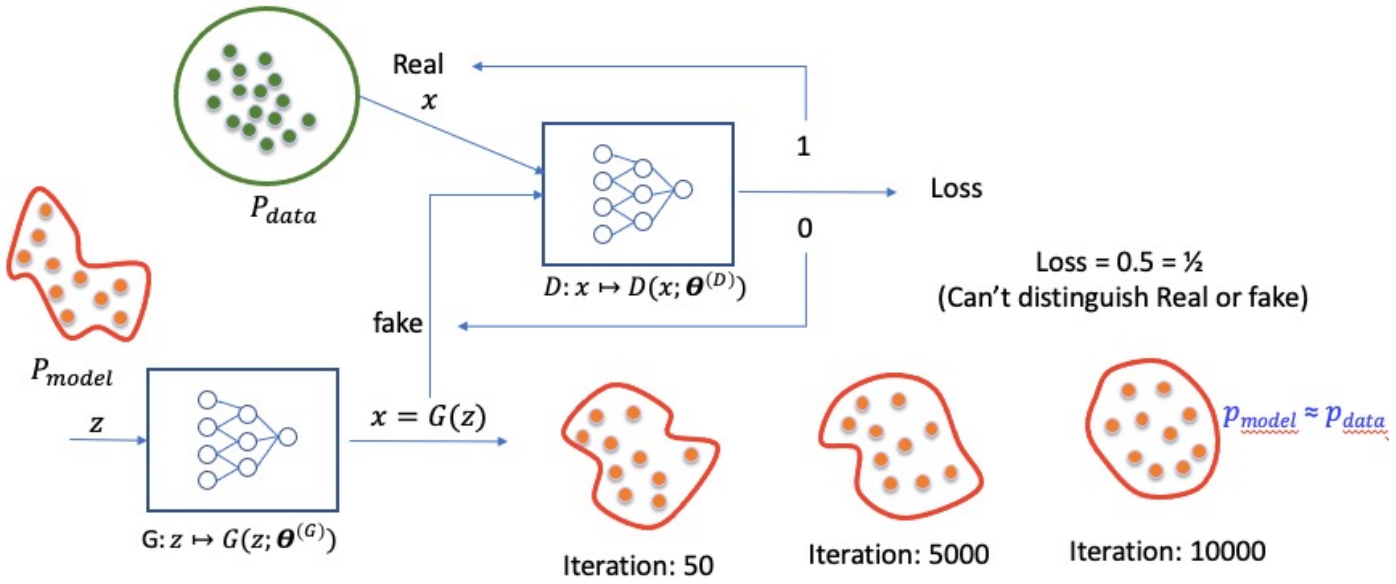
$$max \qquad [\log D(x)] \quad + \quad [\log(1 - D(G(z)))] \quad = D$$



Loss Function of GAN

- In case of generator, we want to maximize D on G(z)

$$max\ D(G(z)) = 1 \quad \Rightarrow \quad min \log(1 - D(G(z))) \quad = G$$



Loss Function of GAN

$$\max\left[\log D(x) + \log(1 - D(G(z)))\right] \qquad = D$$

$$min[\log(1 - D(G(z)))] \qquad = G$$

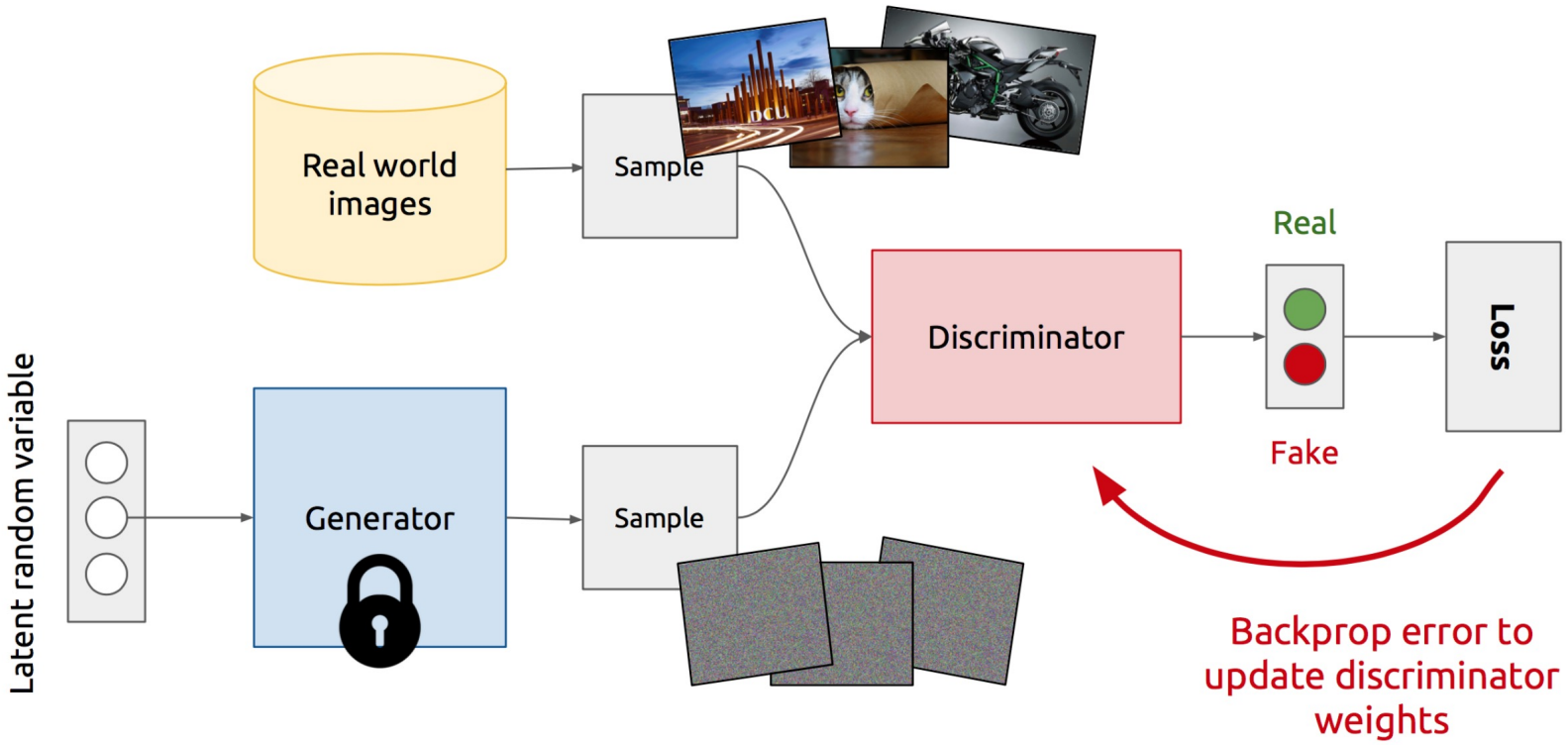
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Training GANs

- Training of GANs proceeds in alternating steps:
  1. The discriminator trains for one or more epochs.
  2. The generator trains for one or more epochs.
  3. Repeat steps 1 and 2 to continue to train the generator and discriminator networks.

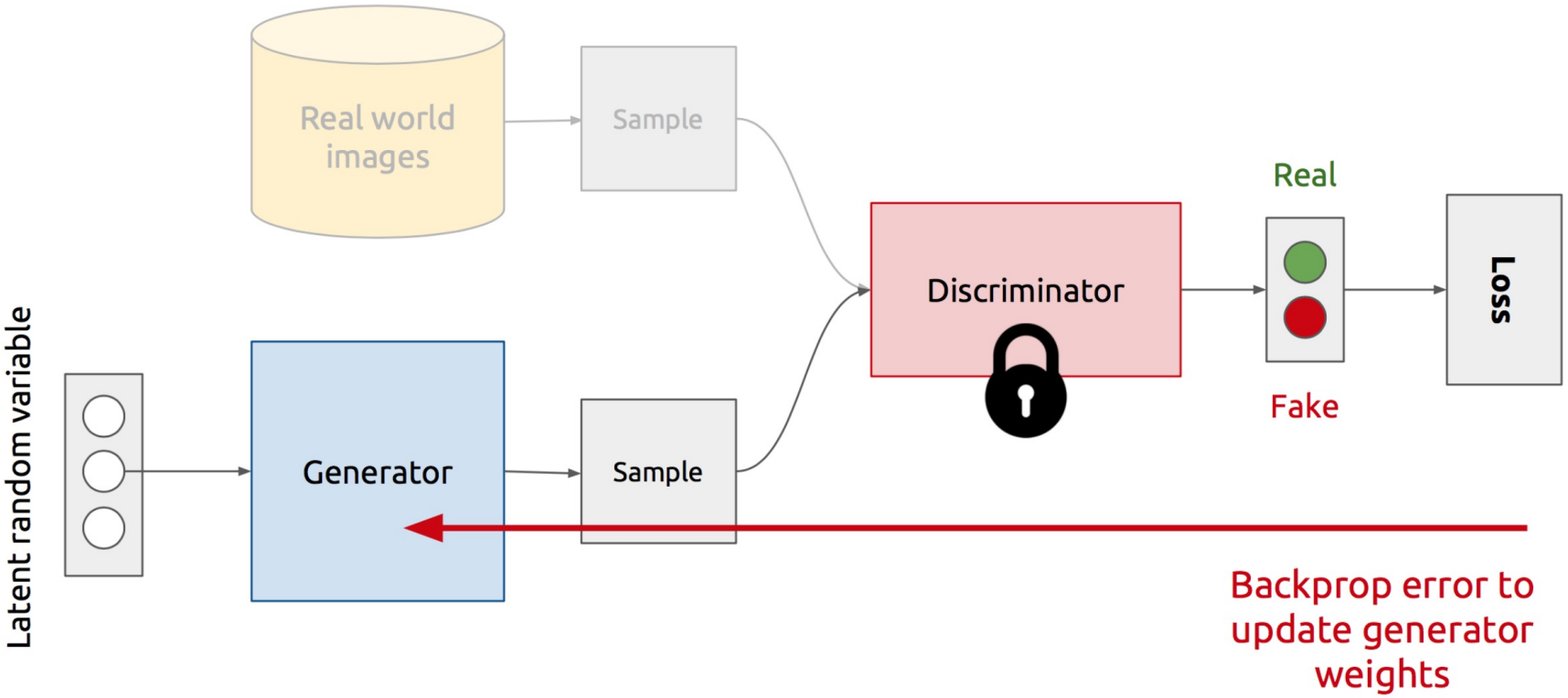
Training Discriminator

1. Fix generator weights, draw samples from both real world and generated images
2. Train discriminator to distinguish between real world and generated images



Training Generator

1. Fix discriminator weights
2. Sample from generator
3. Backprop error through discriminator to update generator weights



Training GANs

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**

Discriminator Updates

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

Generator Updates

- 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:

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

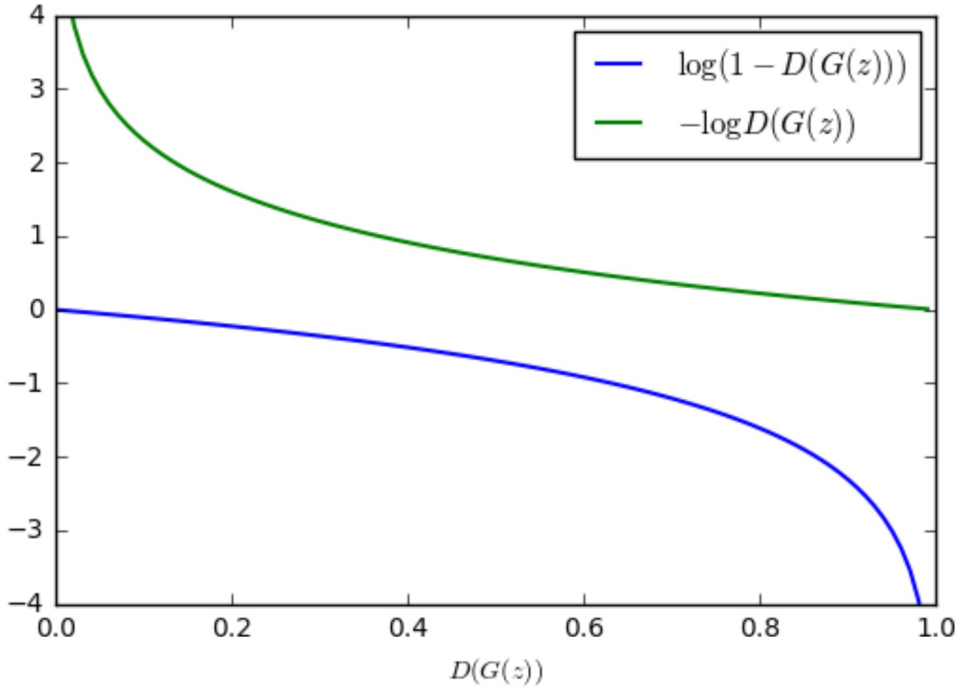
end for

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

In practice, it does not work well, and we use a slightly modified objective

Modification in Generator Updates

- When the sample is likely fake, we want to give a feedback to the generator (using gradients).
- However, in this region where D(G(z)) is close to 0, the curve of the loss function is very flat, and the gradient would be close to 0.
- **Trick:** Instead of minimizing the likelihood of the discriminator being correct, maximize the likelihood of the discriminator being wrong



Training GANs (Modified)

procedure GAN TRAINING

for number of training iterations **do**

for k steps **do**

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

$$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} \left[\log D_\theta \left(x^{(i)} \right) + \log \left(1 - D_\theta \left(G_\phi \left(z^{(i)} \right) \right) \right) \right]$$

end for

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

$$\nabla_\phi \frac{1}{m} \sum_{i=1}^{m} \left[\log \left(D_\theta \left(G_\phi \left(z^{(i)} \right) \right) \right) \right]$$

end for

end procedure

Further Investigation: http://www.offconvex.org/2020/07/06/GAN-min-max/

Thispersondoesnotexist.com

To show you how powerful are GANs

Tips and Tricks to Train GANs

- GANs are difficult to train
  - Non-Convergence
  - Mode-Collapse

- More Information: https://github.com/soumith/ganhacks

Active Area of Research

- Better loss functions, more stable training (Wasserstein GAN, LSGAN, many others)
- Conditional GANs, GANs for all kinds of applications

Self-study

1. Between G and D, whose job is relatively easier? Why is that?

2. Is it important that both G and D are equally good? Why is that?

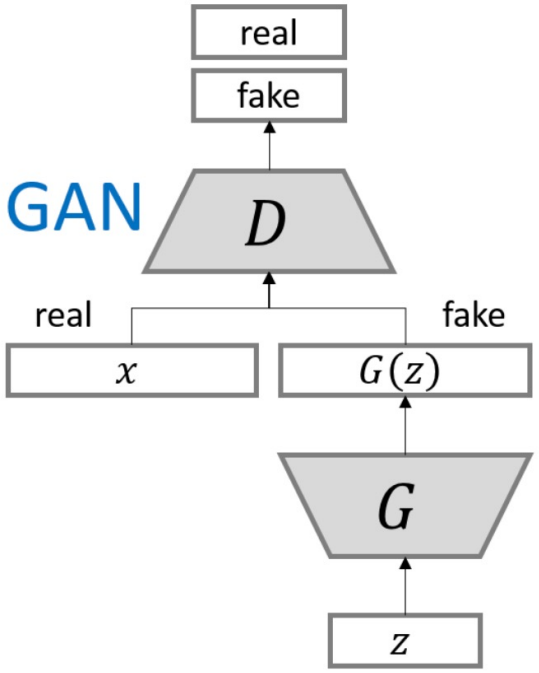
3. What is "Truncation" in regards to GANs? What are fidelity and diversity?

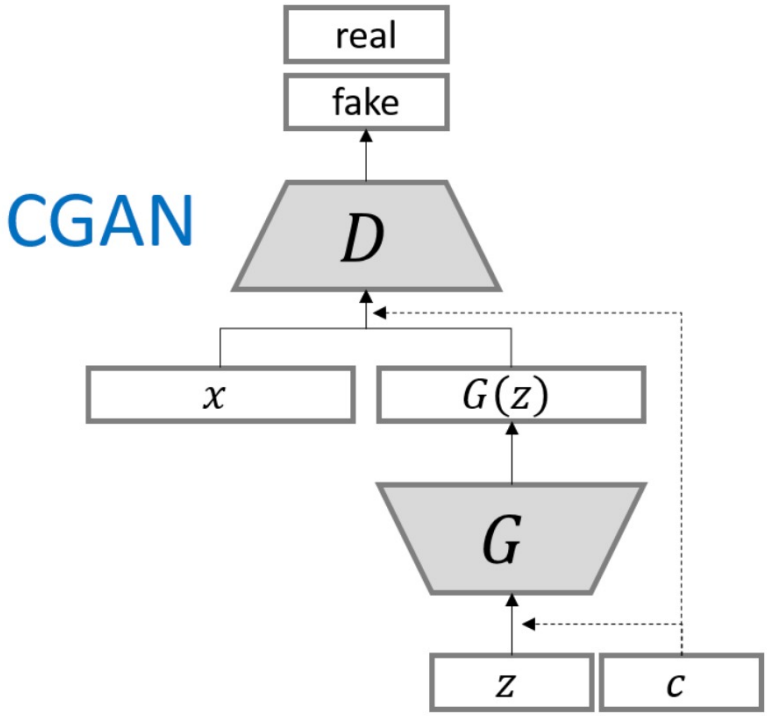
Summary

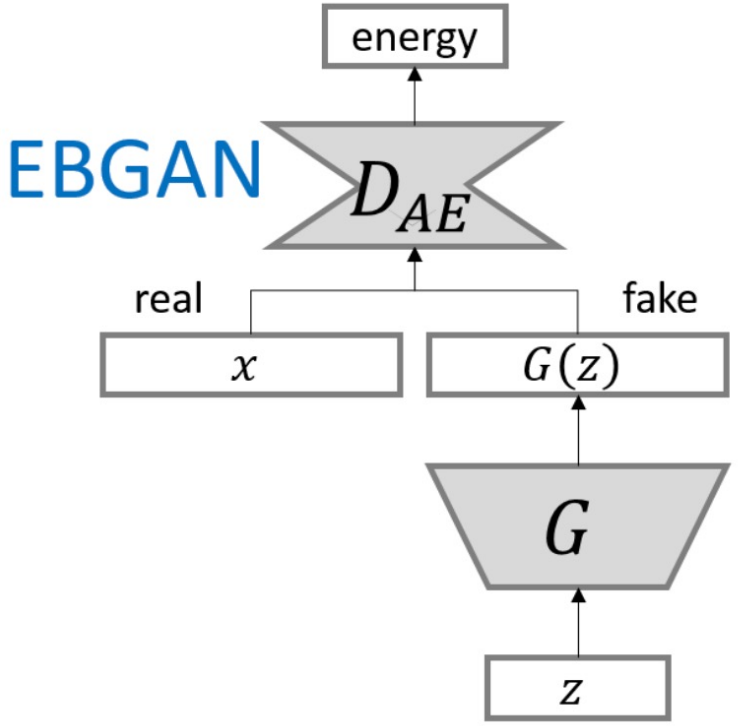
1. Generative vs Discriminative Models
2. Modeling data distribution
3. Generative Adversarial Networks
   - GAN Components
   - GAN Objective
   - GAN Training
4. Issues with GANs
   - Convergence
   - Mode Collapse
5. Related Concepts: Truncation, Fidelity, Diversity

Appendix

Variants of GAN Structure

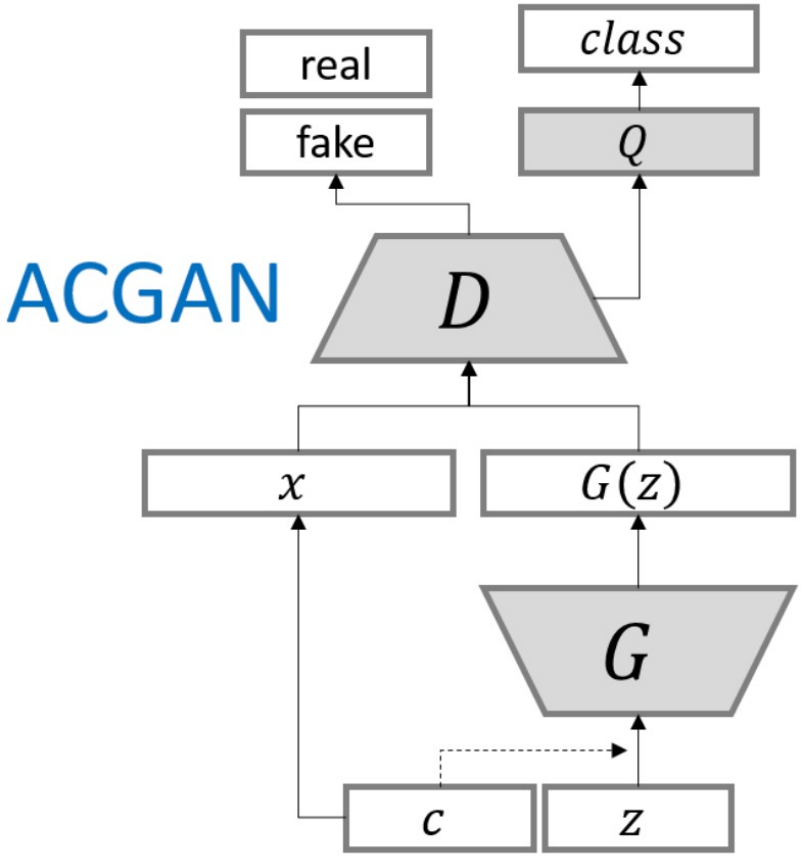


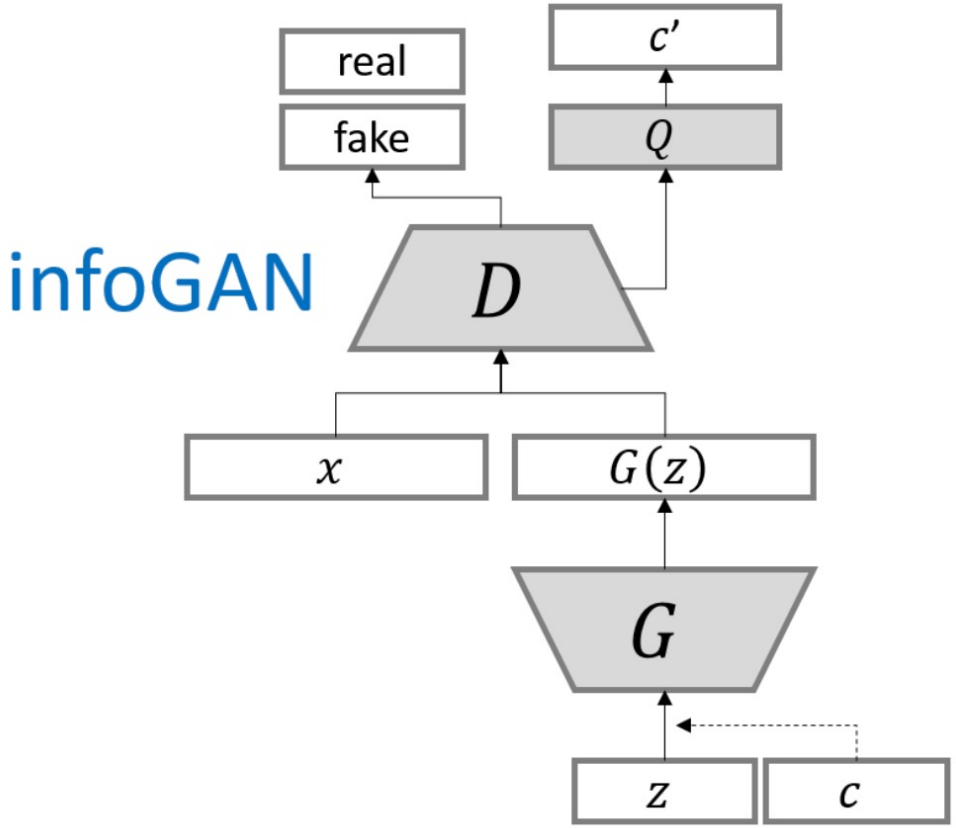




Source: https://github.com/hwalsuklee/tensorflow-generative-model-collections

Variants of GAN Structure





Source: https://github.com/hwalsuklee/tensorflow-generative-model-collections

Conditional GANs (CGAN)

- In an unconditioned generative model, there is no control on modes of the data being generated.
- GANs can be extended to a conditional model if both the generator and discriminator are conditioned on some extra information y.
- y could be any kind of auxiliary information, such as class labels or data from other modalities.
- We can perform the conditioning by feeding y into the both the discriminator and generator as additional input layer.

Conditional GAN (cGAN)

Conditional Loss Function

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

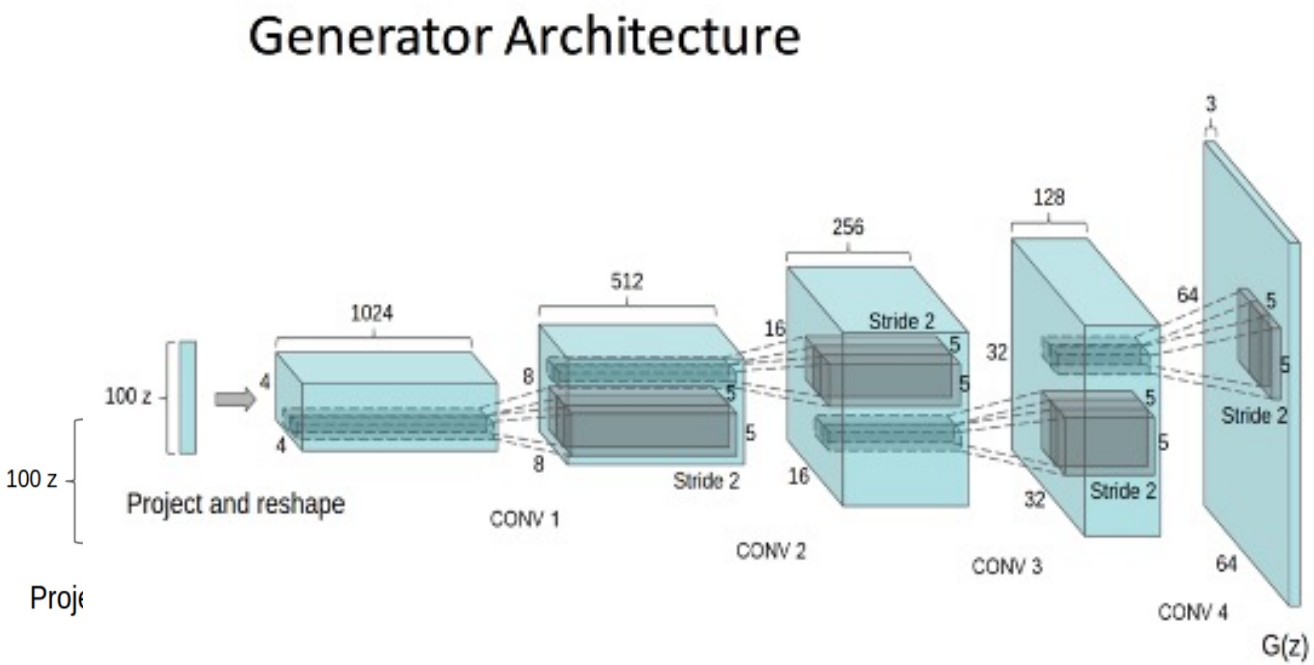
- MNIST Example
  - To be able to tell our generator to generate any digit we want, only a very small change in training is needed.
  - For each iteration, the generator takes as input not only z but also a one-hot encoded vector indicating the digit.
  - The discriminator input consists then of not only the real or fake sample but also the same label vector.

Conditionally generated digit samples

Conditional GAN (Text to Image)

Caption	Image
a pitcher is about to throw the ball to the batter	
a group of people on skis stand in the snow	
a man in a wet suit riding a surfboard on a wave	

Deep Convolutional GANs (DCGANs)



Key ideas:

- Replace FC hidden layers with Convolutions
  - **Generator:** Fractional-Strided convolutions
- Use Batch Normalization after each layer
- **Inside Generator**
  - Use ReLU for hidden layers
  - Use Tanh for the output layer

Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv:1511.06434 (2015).

Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv:1511.06434 (2015).