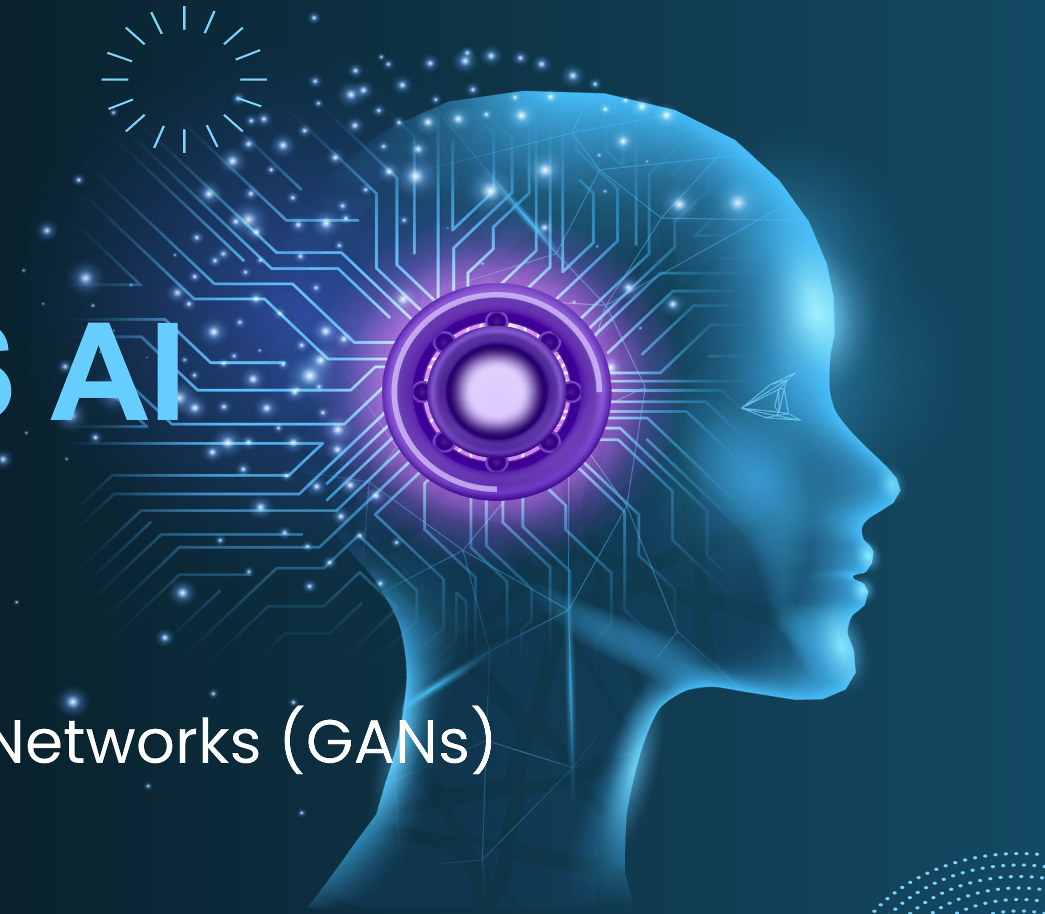
MONASH University

FIT5230 MALICIOUS AI

S2 2025

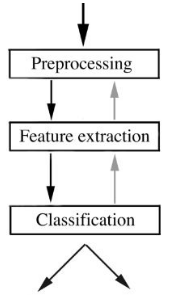
Week 6:

Generative Adversarial Networks (GANs)



Overview

- Recap on Classification



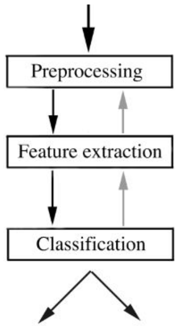
- Generative Models
- Discriminative Models

- Generative Adversarial Networks
    - GAN & Game Theory
    - Game vs Optimisation Problem

Generative Adversarial Networks

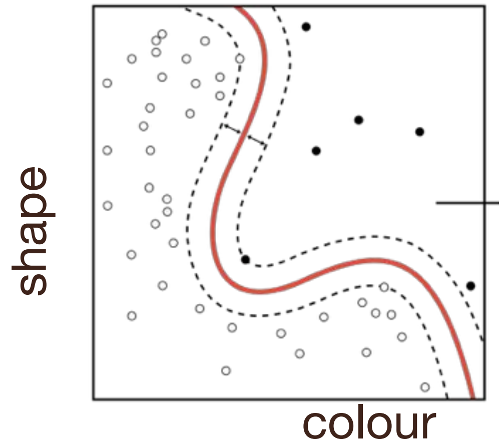
Security Gaming meets AI

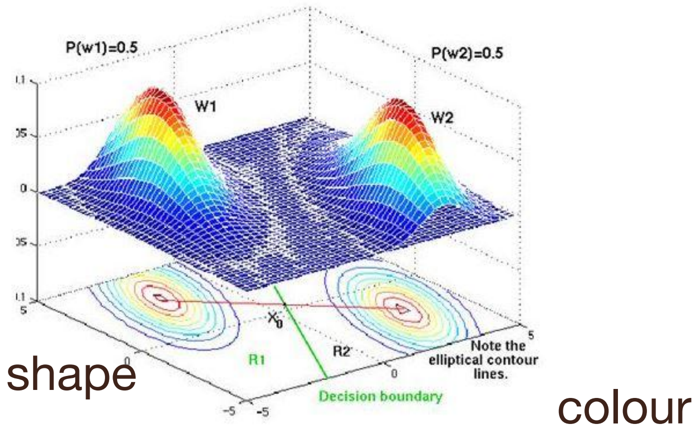
Classification



- Recap: Classification part of Pattern Recognition
  - pre-processing, Feature Extraction, Classification

- Classification: given observed sample x, predict its class / category label y



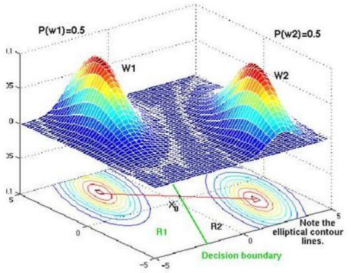


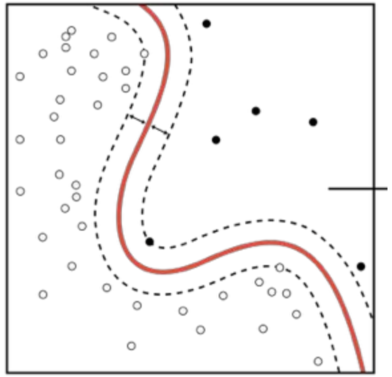
Classification: using statistics

- Classification: given sample x, predict its class label y ∈ {1,2}
- Training:
  - given $(x_1,y_1),\ldots,(x_n,y_n)$
  - compute features of x_i
    - $(f_1^{\,i}, f_2^{\,i}) = FE(x_i)$
  - plot how frequently the features of a class y_i have what values

$\Rightarrow P(Y \mid (f_1^{\,i}, f_2^{\,i}))$

so given an x, can know most likely y





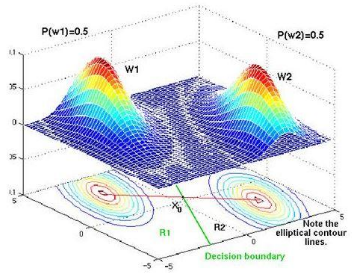
Classification: using statistics

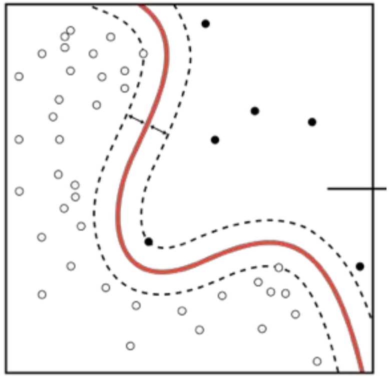
- Classification (check during testing): given unknown sample x, predict its class label y

- Training:

  $\Rightarrow P(Y \mid (f_1, f_2))$

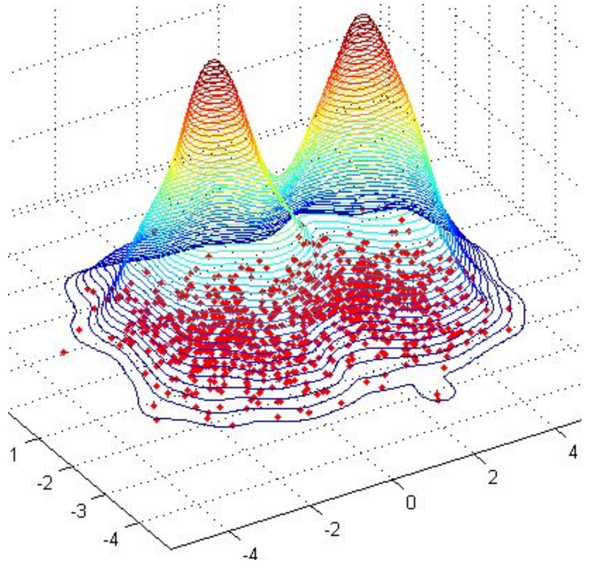
- Testing: given unknown sample x
  - compute $(f_1, f_2) = FE(x)$
  - output y s.t. $\max[P(Y \mid (f_1, f_2))]$





Classification Models

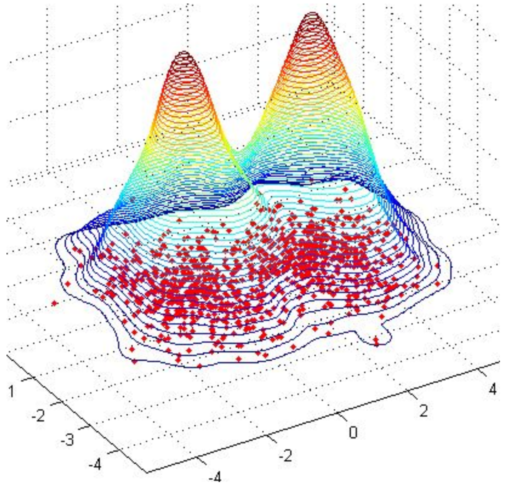
- Classification: given x, predict its class label y
  - Discriminative
    - given x, compute $P(Y \mid F=(f_1,f_2))$
    - output y s.t. max $P(Y|F)$
  - Generative
    - given x, find $P(F, Y)$ or $P(F|Y)$ for each class y
    - then compute $P(Y|F)$ using Bayes' theorem



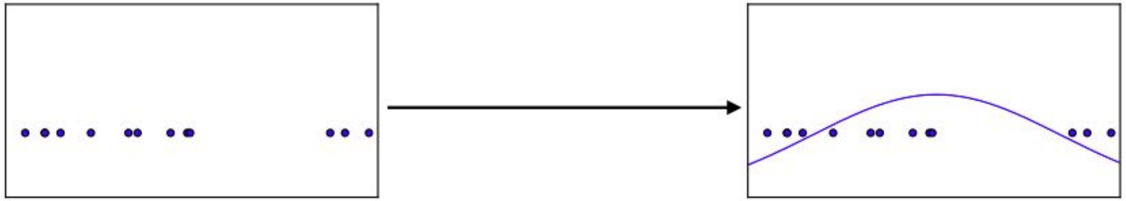
Classification Models*

- Classification: given x, predict its class label y
  - Discriminative
    - given x, compute $P(Y \mid F=(f_1,f_2))$
    - output y s.t. max $P(Y|F)$
  - Generative
    - given x, find $P(F, Y)$ or $P(F|Y)$ for each class y
    - compute $P(Y|F)$

      $= \dfrac{P(F|Y)P(Y)}{P(F)}$ [Bayes']



Generative Models



- Case (I):
  - given samples (the points) x_i from distribution p_{real}, obtain an estimation p_{model} of that distribution
  - use $p_{model}(x)$ to find the probability that x occurs
- Case (II):
  - given samples (the points) x_i from distribution p_{real}, estimate more examples x' from the same distribution
  - e.g. GANs

Generative vs Discriminative Models

- Generative model

P(X,Y) or P(X|Y)

$$\begin{tikzcd}
\boxed{\begin{array}{c}\textbf{samples}\\ \mathbf{x}\end{array}} \arrow[r] & \boxed{G} \arrow[r] & \boxed{\mathbf{p}_{\textbf{model}}} \arrow[r] & \boxed{\begin{array}{c}\textbf{samples}\\ \mathbf{x'}\end{array}}
\end{tikzcd}$$

$\in \mathbf{p}_{\textbf{data}}$

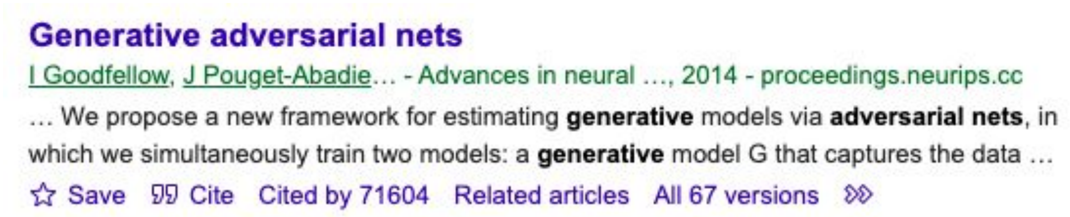
$$\begin{tikzcd}
\boxed{\begin{array}{c}\textbf{samples}\\ \mathbf{x}\end{array}} \arrow[r] & \boxed{G} \arrow[r] & \boxed{\begin{array}{c}\textbf{samples}\\ \mathbf{x'}\end{array}}
\end{tikzcd}$$

- Discriminative model

$$\begin{tikzcd}
\boxed{\begin{array}{c}\textbf{samples}\\ \mathbf{x}\end{array}} \arrow[r] & \boxed{D} \arrow[r] & \boxed{\begin{array}{c}\textbf{class}\\ \mathbf{y}\end{array}}
\end{tikzcd}$$

based on max of P(Y|X)

Generative Adversarial Networks


Generative adversarial nets

I Goodfellow, J Pouget-Abadie... - Advances in neural ..., 2014 - proceedings.neurips.cc

... We propose a new framework for estimating **generative** models via **adversarial nets**, in which we simultaneously train two models: a **generative** model G that captures the data ...

☆ Save 99 Cite Cited by 71604 Related articles All 67 versions »

Ian Goodfellow

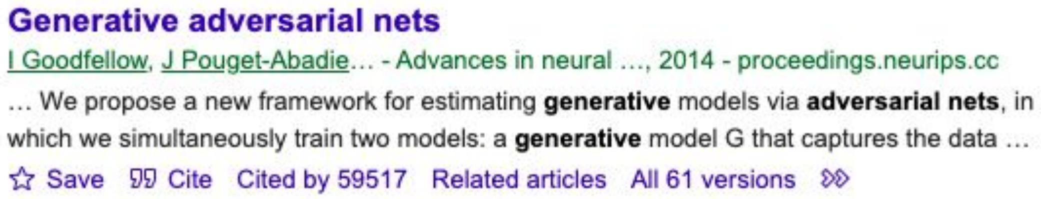
Unknown affiliation

Verified email at cs.stanford.edu

Deep Learning

@ Aug 2024

Generative Adversarial Networks



Generative adversarial nets

I Goodfellow, J Pouget-Abadie... - Advances in neural ..., 2014 - proceedings.neurips.cc

... We propose a new framework for estimating **generative** models via **adversarial nets**, in which we simultaneously train two models: a **generative** model G that captures the data ...

Ian Goodfellow

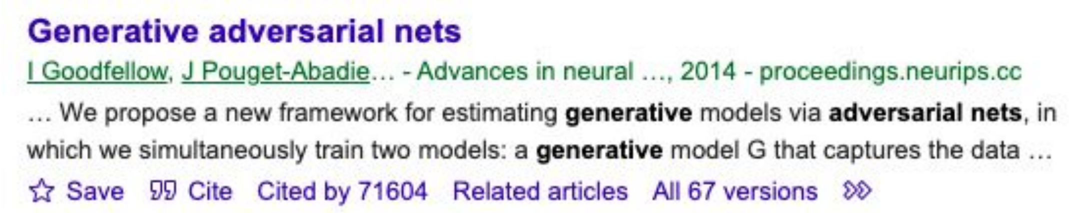
Unknown affiliation

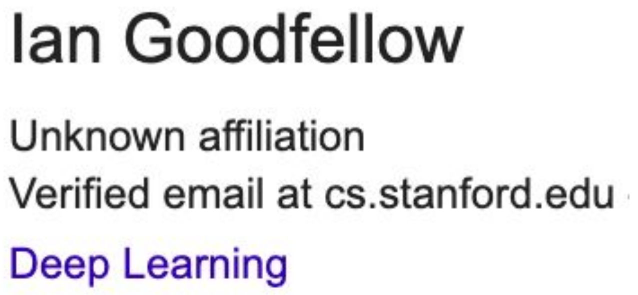
Verified email at cs.stanford.edu

Deep Learning

@ Aug 2023

Generative Adversarial Networks


Generative adversarial nets
I Goodfellow, J Pouget-Abadie… - Advances in neural …, 2014 - proceedings.neurips.cc
… We propose a new framework for estimating **generative** models via **adversarial nets**, in which we simultaneously train two models: a **generative** model G that captures the data …


Ian Goodfellow

Unknown affiliation

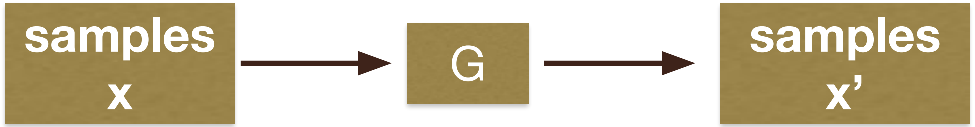
Verified email at cs.stanford.edu

Deep Learning

+ > 12000 citations in a year
- 7100++ citations/year

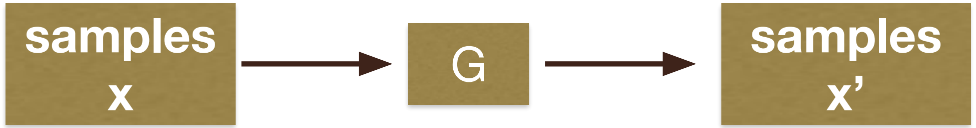
GAN & Generative Models

- GAN is a type of generative model
    - obtain more samples from the same distribution



Generative Adversarial Networks

- a model like security game:
  - 2 opposing players, G vs D



- Generator G
  - creates samples x' from the same distribution as the real samples x s.t. D(x) = D(x') , i.e. D can't tell the difference
  - goal: INDistinguishability of generated samples
    - vs IND for crypto

Generative Adversarial Networks

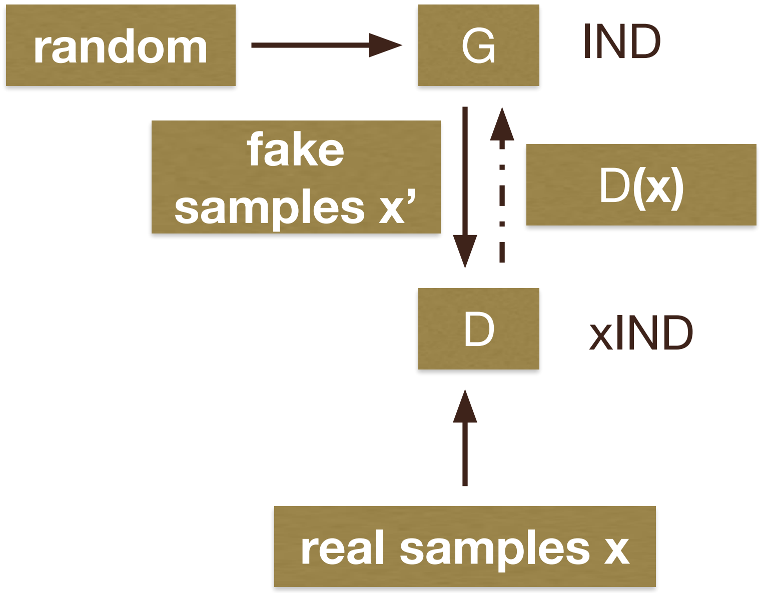
- a model like security game: 2 opposing players, G vs D
- Generator G
    - goal: INDistinguishability of generated samples

$$\begin{tikzcd} \arrow[r, "\text{random}"] & \boxed{G} \arrow[r] & x' \end{tikzcd}$$

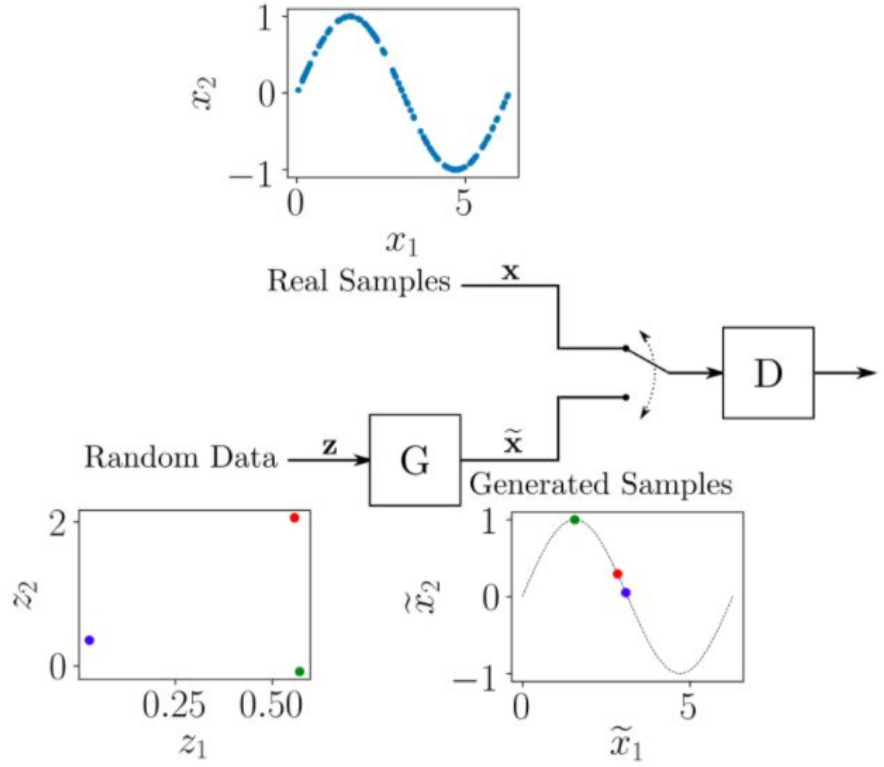
- Discriminator D
    - checks samples to determine if real or fake
    - goal: break IND
    - outputs probability estimation D(·) that i/p is real

$$\begin{tikzcd} \arrow[r, "x", "x'"'] & \boxed{D} \arrow[r] & \begin{array}{c} D(x)= \\ \text{Prob(it's x)} \end{array} \end{tikzcd}$$

Generative Adversarial Networks



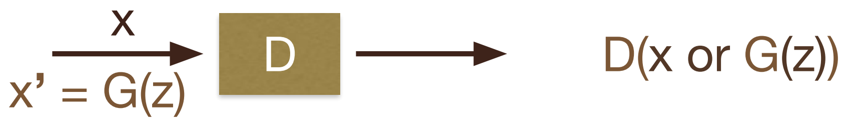
Generative Adversarial Networks



GAN & Game Theory

- GAN Training: 2-player minimax game
  - each player's cost/loss function J_D, J_G = f(both players' params θ_D, θ_G)

- D



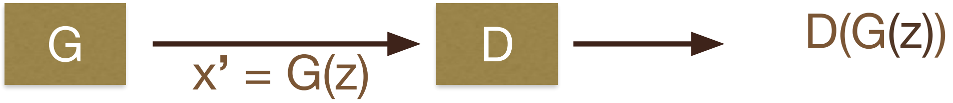
  - maximize the probability D(x) where $x \sim p_{data}$
  - minimize the probability D(G(z)) where $G(z) \sim p_{model}$
  - by controlling only θ_D, taking turns with G

MONASH University

GAN & Game Theory

- GAN Training: 2-player minimax game

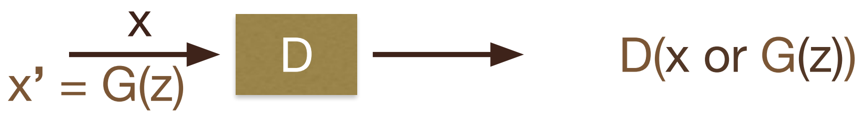
- G



  - maximize the probability D(G(z)) where G(z)~p_{model} , or
  - minimize 1-D(G(z))
  - by controlling only θ_G, taking turns with D

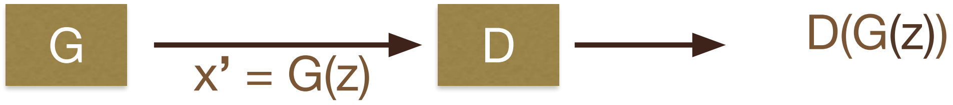
GAN & Game Theory

- D:



  - max the probability D(x),
  - min the probability D(G(z)), or max 1-D(G(z))

- G:



  - max the probability D(G(z)), or min 1-D(G(z))

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

Game vs Optimisation Problem

- Each player's cost J_D (J_G) depends on the other player's parameters θ_G (θ_D), but cannot control that, so is not optimisation but a game

- Solution to optimisation problem is a minimum point

  vs

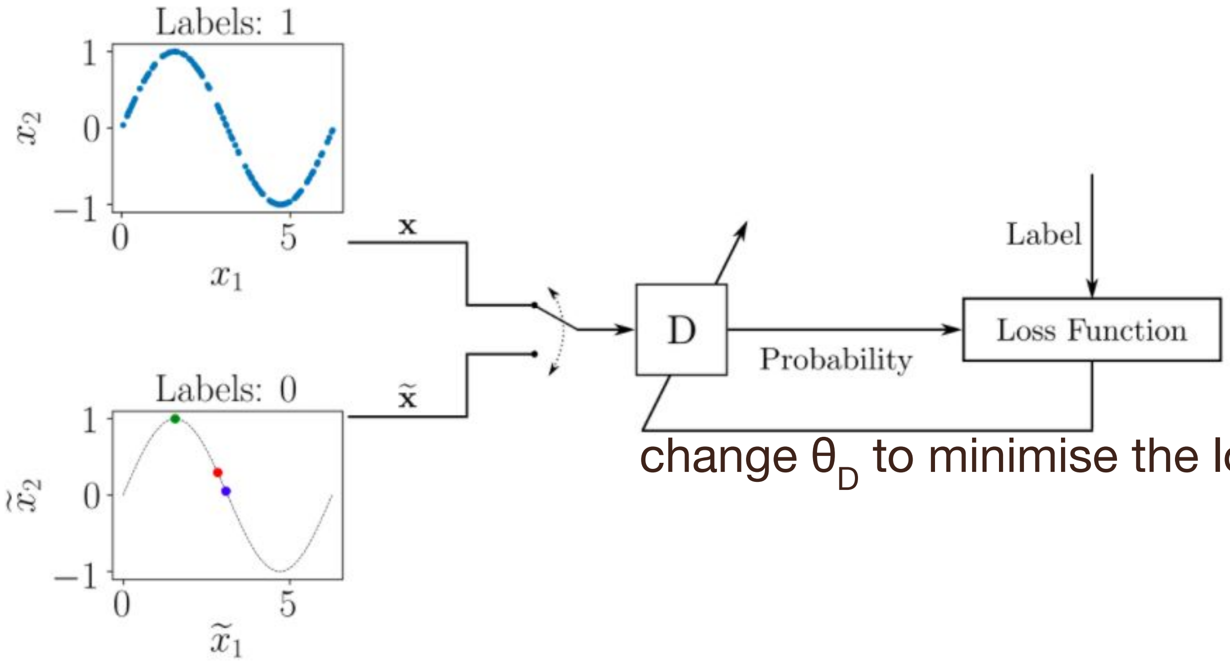
- Solution to a game is a Nash equilibrium

$$\underset{\theta_G}{\arg\min}\ \underset{\theta_D}{\max}\ J(\theta_D, \theta_G)$$

where arg max just means arguments of the maxima i.e. values of θ

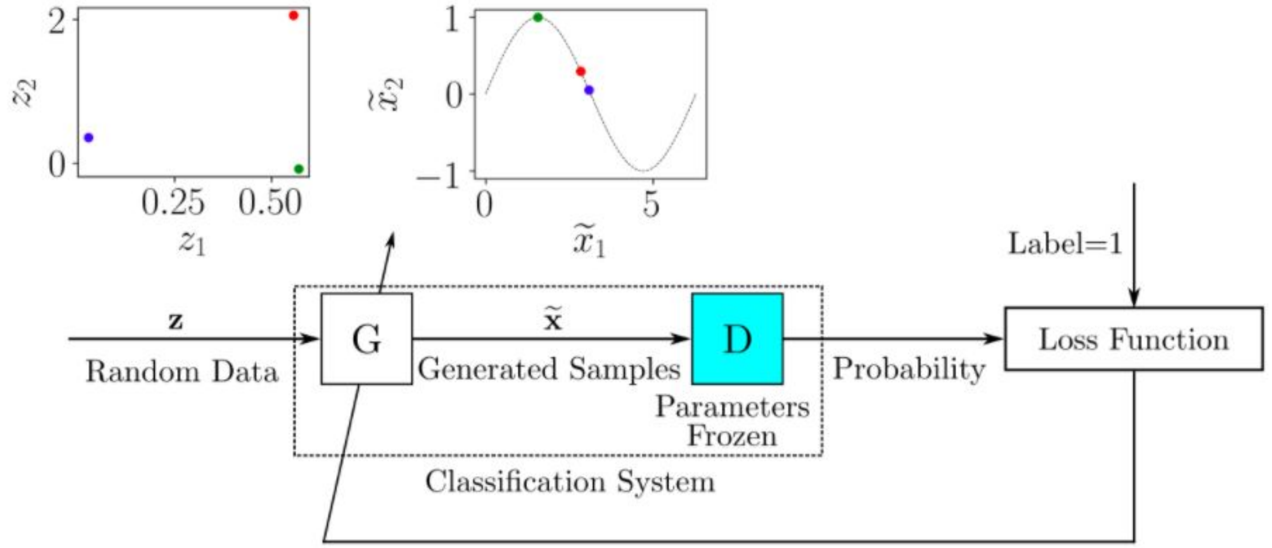
s.t. the expression has a max value

GAN Discriminator D



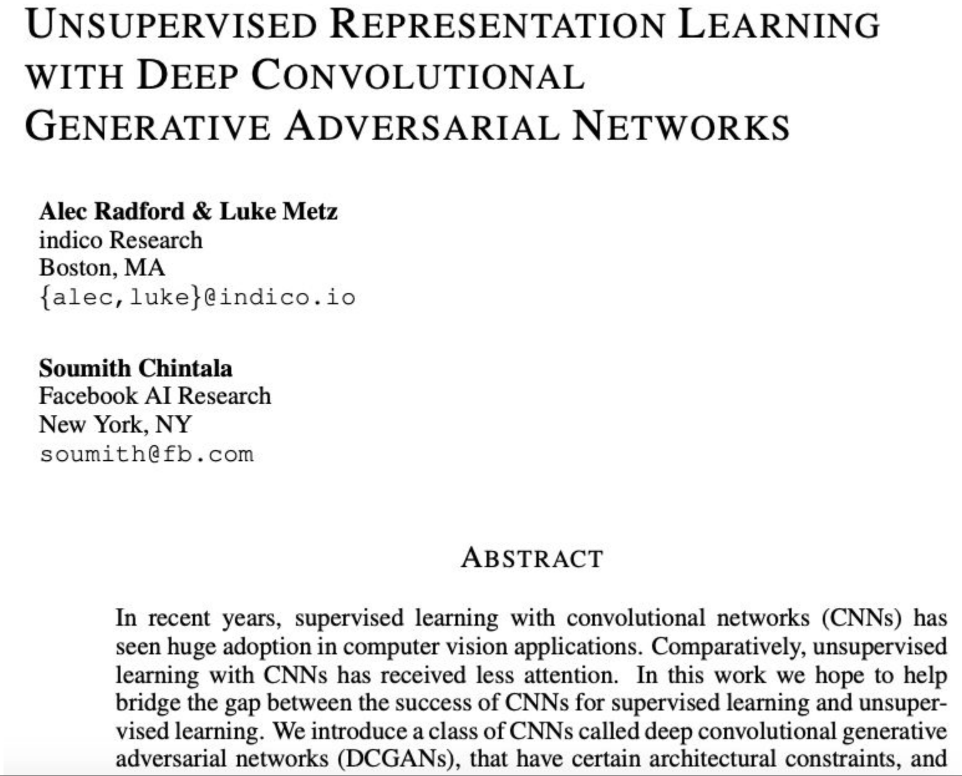
change θ_D to minimise the loss function

GAN Generator G



G benefits from D's outputs

DCGAN @ICLR 2016



UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

Alec Radford & Luke Metz
indico Research
Boston, MA
{alec,luke}@indico.io

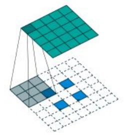
Soumith Chintala
Facebook AI Research
New York, NY
soumith@fb.com

ABSTRACT

In recent years, supervised learning with convolutional networks (CNNs) has seen huge adoption in computer vision applications. Comparatively, unsupervised learning with CNNs has received less attention. In this work we hope to help bridge the gap between the success of CNNs for supervised learning and unsupervised learning. We introduce a class of CNNs called deep convolutional generative adversarial networks (DCGANs), that have certain architectural constraints, and

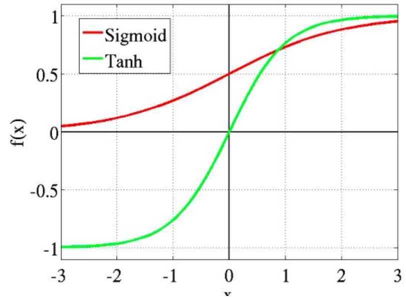
cited > 18,000 times

DCGAN

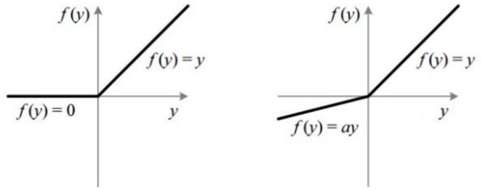


- overall network structure based on all convolutional net
  - conventional: conv, pooling, fully connected
  - AllConvNet: replace pooling with conv, no fully connected layer
- batch normalisation layers in most layers of G and D
- ReLU for G except Tanh for last layer, LeakyReLU for D
- does not have pooling nor unpooling layers
  - upsampling done via transposed convolution with stride > 1
- uses Adam optimiser instead of SGD to update the weights

DCGAN



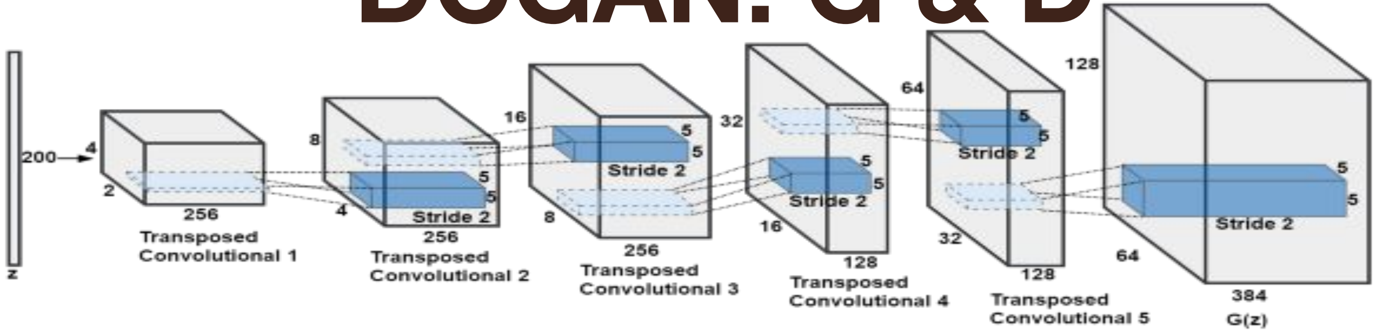
- Tanh activation function: range –1 to 1



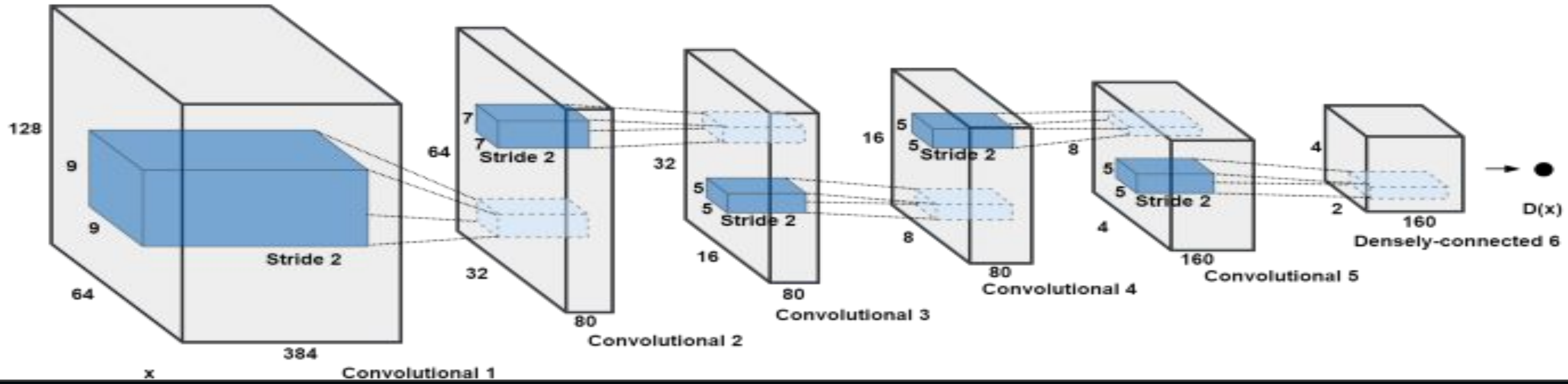
- LeakyReLU:
  - non-zero for –ve input
- Adam optimiser:
  - alternative optimization to stochastic/random gradient descent, to update the neural network weights

DCGAN: G & D

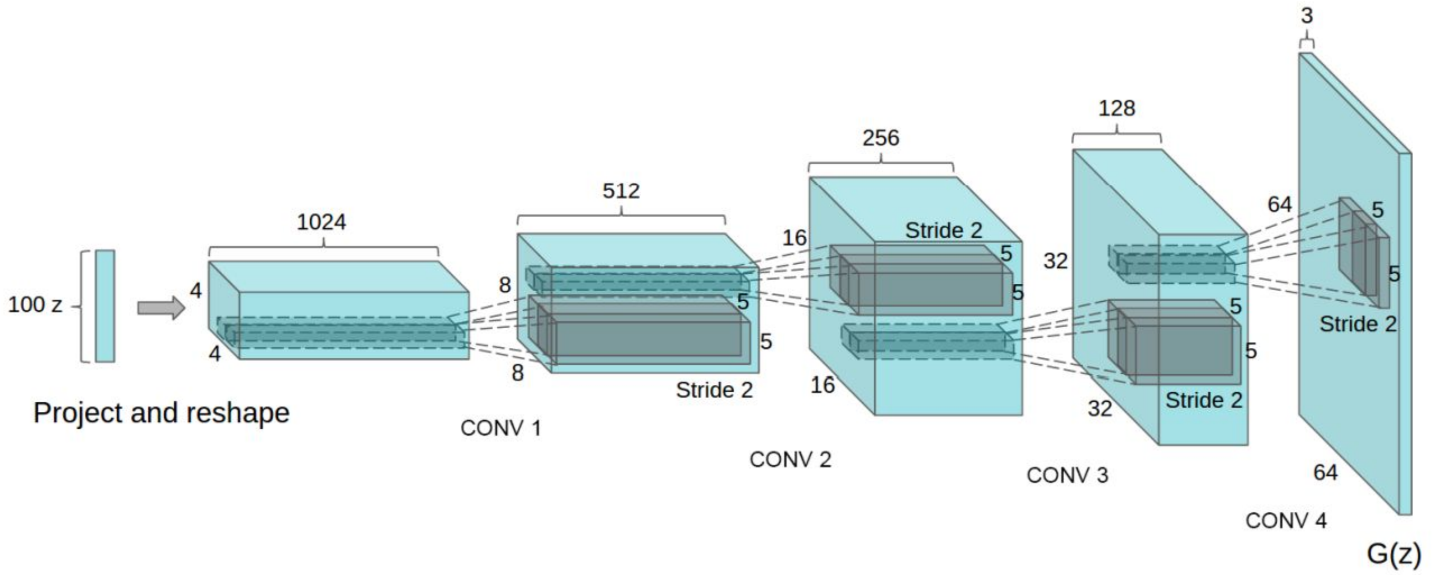
(a)


z
200
4
2
256
Transposed Convolutional 1
8
4
5
5
Stride 2
256
Transposed Convolutional 2
16
8
5
5
Stride 2
256
Transposed Convolutional 3
32
16
5
5
Stride 2
128
Transposed Convolutional 4
64
32
5
5
Stride 2
128
Transposed Convolutional 5
128
64
5
5
Stride 2
384
G(z)

(b)

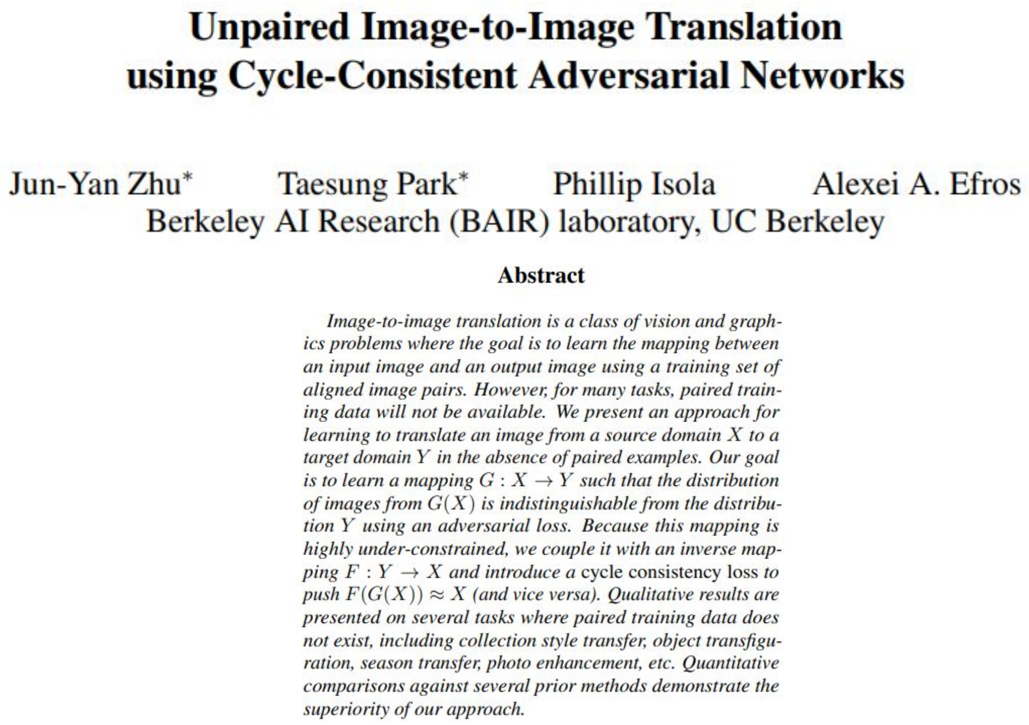

128
9
9
Stride 2
64
384
x
Convolutional 1
64
7
7
Stride 2
32
80
Convolutional 2
32
5
5
Stride 2
16
80
Convolutional 3
16
5
5
Stride 2
8
80
Convolutional 4
8
5
5
Stride 2
4
160
Convolutional 5
4
2
160
Densely-connected 6
D(x)

DCGAN: G



100 z
Project and reshape
1024
4
4
CONV 1
512
8
8
5
5
Stride 2
CONV 2
256
16
16
Stride 2
5
5
CONV 3
128
32
32
5
5
Stride 2
CONV 4
3
64
64
5
5
Stride 2
G(z)

CycleGAN@ ICCV 2017



Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

Jun-Yan Zhu[*] Taesung Park[*] Phillip Isola Alexei A. Efros
Berkeley AI Research (BAIR) laboratory, UC Berkeley

Abstract

Image-to-image translation is a class of vision and graphics problems where the goal is to learn the mapping between an input image and an output image using a training set of aligned image pairs. However, for many tasks, paired training data will not be available. We present an approach for learning to translate an image from a source domain X to a target domain Y in the absence of paired examples. Our goal is to learn a mapping $G : X \rightarrow Y$ such that the distribution of images from $G(X)$ is indistinguishable from the distribution Y using an adversarial loss. Because this mapping is highly under-constrained, we couple it with an inverse mapping $F : Y \rightarrow X$ and introduce a cycle consistency loss *to push $F(G(X)) \approx X$ (and vice versa). Qualitative results are presented on several tasks where paired training data does not exist, including collection style transfer, object transfiguration, season transfer, photo enhancement, etc. Quantitative comparisons against several prior methods demonstrate the superiority of our approach.*

cited > 28,000 times

CycleGAN

- Learns to translate between domains using **unpaired datasets**.
- CycleGAN consists of:
  - Two Generators:
    - G: X → Y (e.g., horse → zebra)
    - F: Y → X (e.g., zebra → horse)

  - Two Discriminators:
    - D_Y: distinguishes real vs fake images in domain Y
    - D_X: distinguishes real vs fake images in domain X

CycleGAN

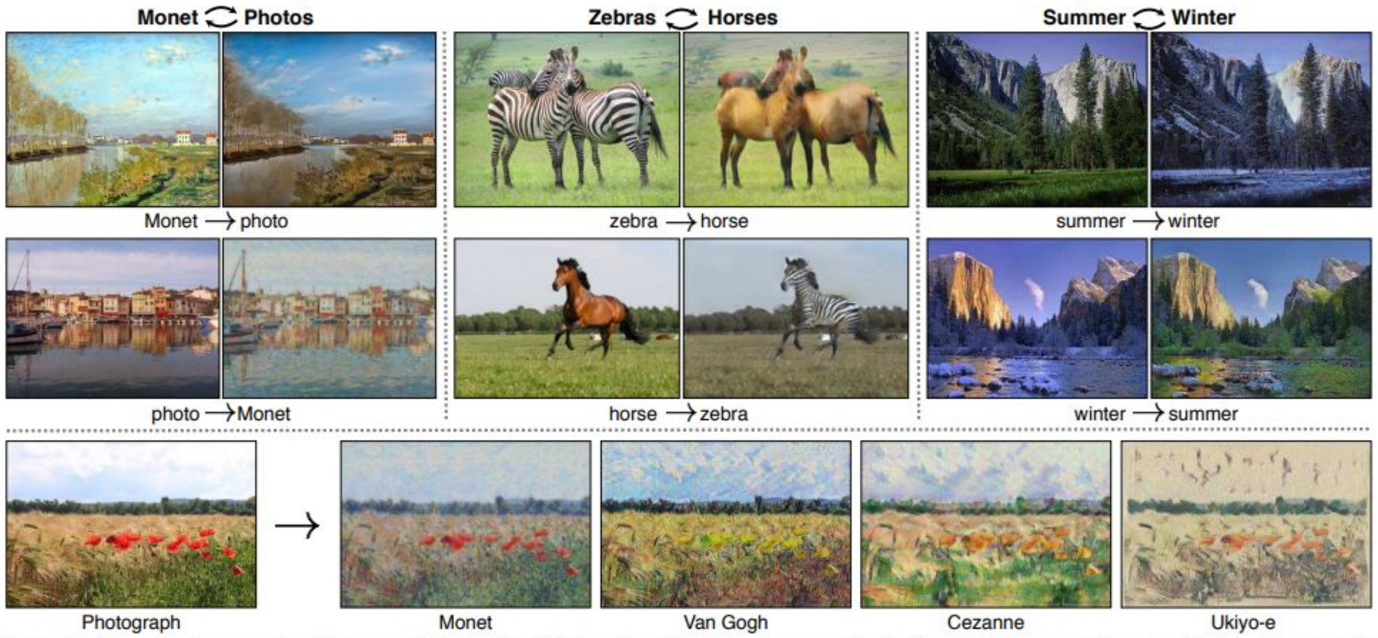
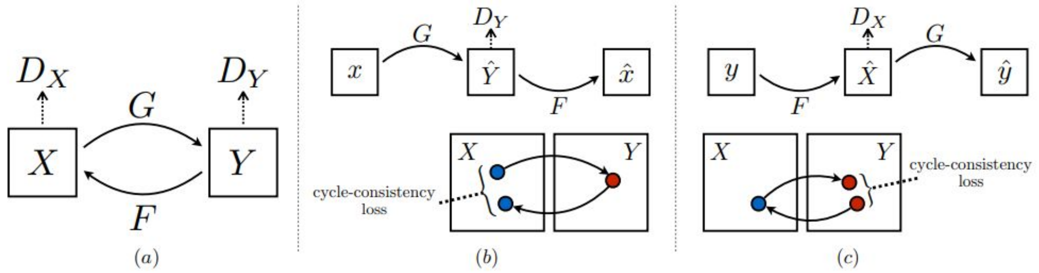


Figure 1: Given any two unordered image collections X and Y, our algorithm learns to automatically "translate" an image from one into the other and vice versa. Example application (*bottom*): using a collection of paintings of a famous artist, learn to render a user's photograph into their style.

CycleGAN

- Cycle consistency loss:
    - If you translate an image from domain *X* to *Y* using *G*, and then back to *X* using *F*, you should get the original image: $F(G(x)) \approx x$
    - Similarly: $G(F(y)) \approx y$
- Prevents the generators from producing arbitrary outputs and enforces semantic preservation.



CycleGAN

- Adversarial Loss encourages generators to produce realistic images that fool the discriminators:

$$
\begin{aligned}
\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) =& \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] \\
&+ \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))],
\end{aligned}
$$

- Cycle Consistency Loss ensures that translating back and forth yields the original image:

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

Evaluation Metrics for GANs

- **Inception Score (IS)**
  - Measures image quality and diversity.
    - Image Quality: A good image should be confidently classified into a specific class.
    - Image Diversity: The generated images should span across many different classes.
  - Uses a pretrained classifier (e.g., Inception v3).
  - High score = sharp, diverse images.

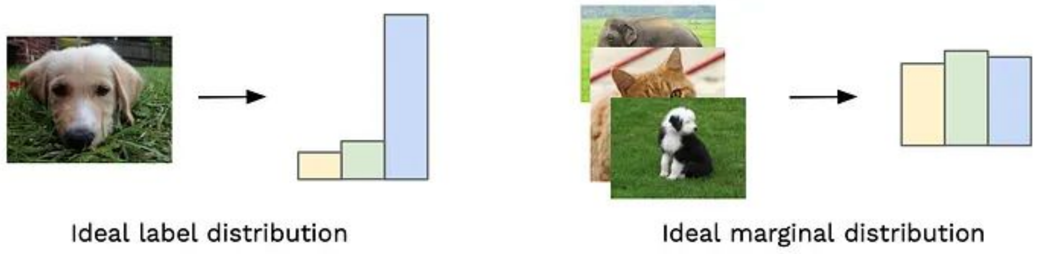


Image source

Inception Score (IS)*

- The Inception Score is defined as:

$$\exp(\mathbb{E}_{\boldsymbol{x}} \mathrm{KL}(p(y|\boldsymbol{x}) || p(y)))$$

where:

- x be a generated image.
- p(y|x) be the conditional label distribution predicted by the Inception model.
- p(y) be the marginal distribution over all generated images.
- D_{KL} is the Kullback-Leibler divergence.
- E_x is the expectation over all generated images.

Image source

Evaluation Metrics for GANs

- **Fréchet Inception Distance (FID)**
  - Compares statistics (e.g., mean and covariance) of real vs generated images.
  - Lower FID = closer to real distribution.
  - More robust than IS for detecting mode collapse.

Fréchet Inception Distance (FID)*

- FID is computed as:

$$\|\boldsymbol{m} - \boldsymbol{m}_w\|_2^2 + \operatorname{Tr}\left(\boldsymbol{C} + \boldsymbol{C}_w - 2\left(\boldsymbol{C}\boldsymbol{C}_w\right)^{1/2}\right)$$

Where:

- $\|\boldsymbol{m} - \boldsymbol{m}_w\|_2^2$ measures the difference in means
- Tr is the trace of a matrix (sum of diagonal elements)
- $\left(\boldsymbol{C}\boldsymbol{C}_w\right)^{1/2}\Big)$ is the matrix square root of the product of covariances.

Evaluation Metrics for GANs

- **Perceptual Path Length (PPL)**
  - Measures smoothness of latent space interpolation
  - Reflects how well the generator maps small changes in latent vectors to perceptually smooth changes in the generated images.
  - Lower PPL = more consistent transitions

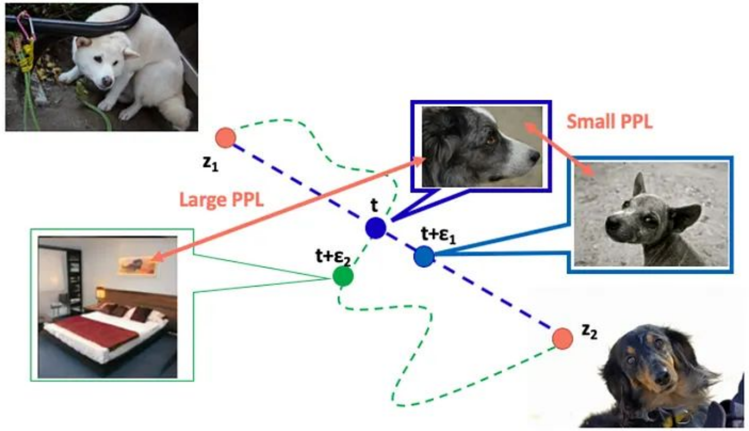


Image source

Perceptual Path Length (PPL)*

- PPL is computed as:
  - Sample two latent vectors z_1 and z_2
  - Interpolate between them using a small step ϵ: $z(t) = (1 - t)z_1 + tz_2$
  - Generate images from interpolated latent vectors: $I(t) = G(z(t))$
  - Compute PPL:

$$\mathbb{E}_{z_1, z_2, t}\left[\frac{d(I(t), I(t+\epsilon))}{\epsilon^2}\right]$$

Where:

- d(x) is a perceptual distance (not pixel-wise)
- The expectation is taken over many random pairs and interpolation points

Evaluation Metrics for GANs

- **Precision** & **Recall**
  - Precision: How realistic are the generated samples?
  - Recall: How diverse are the generated samples?

- Embed images (real and generated) into a feature space using a pretrained network.
- Model the distributions of real and generated images in this space.
- For each generated image:
  - Check if it lies within the manifold of real images → contributes to precision.
- For each real image:
  - Check if it lies within the manifold of generated images → contributes to recall.

Precision & Recall

- Let:
  - G: set of generated samples
  - R: set of real samples
  - $\phi(x)$: feature embedding of image x
- Precision: Fraction of $\phi(g) \in G$ that are close to some $\phi(r) \in R$.
- Recall: Fraction of $\phi(r) \in R$ that are close to some $\phi(g) \in G$.
- “Closeness” is typically defined using a k-nearest neighbor or radius-based threshold in feature space.

MONASH University

Appendix

Cross Entropy Loss*

- Loss? Common in machine learning

- Cross-entropy loss / log loss used in GAN
  - for classification where output = probability

- Information vs Entropy

Cross Entropy Loss*

- Cross-entropy loss / log loss

- Information h(x) = –log(p(x)) , –ve is to ensure the result is +ve

- Measure of Information:
  - to quantify how much info is available
  - how much surprise when something happens?
    - low probability event: high info / surprising / uncertain
      - if $p(x) \approx 0$, info $h(x) = -\log(p(x)) \rightarrow \infty$, very surprising
    - high probability event: low info / ho-hum

Cross Entropy Loss*

- Information h(x) = –log(p(x)) , –ve is to ensure the result is +ve

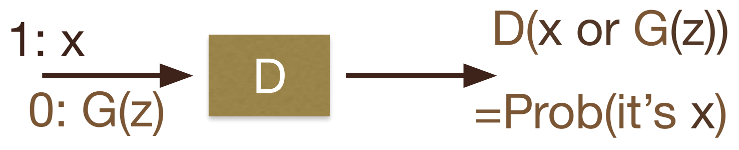
- Entropy H(x):
  - average number of bits/info/surprise/uncertainty to transmit a random event from a probability distribution

$$\mathrm{H}(X) := -\sum_{x\in\mathcal{X}} p(x)\log p(x) = \mathbb{E}[-\log p(X)].$$

  - expected value of self-information of a random variable

Cross Entropy Loss*

- Loss? Common in machine learning
- GAN loss? cross-entropy loss



- Cross-entropy loss:
  - calculate total entropy between two distributions
    - in the case of GAN: distribution p(x) of real label and distribution q(x) of predicted label

$$H(p, q) = \mathbb{E}_{x \sim p(x)}[-\log q(x)]$$

- where label is predicted by D