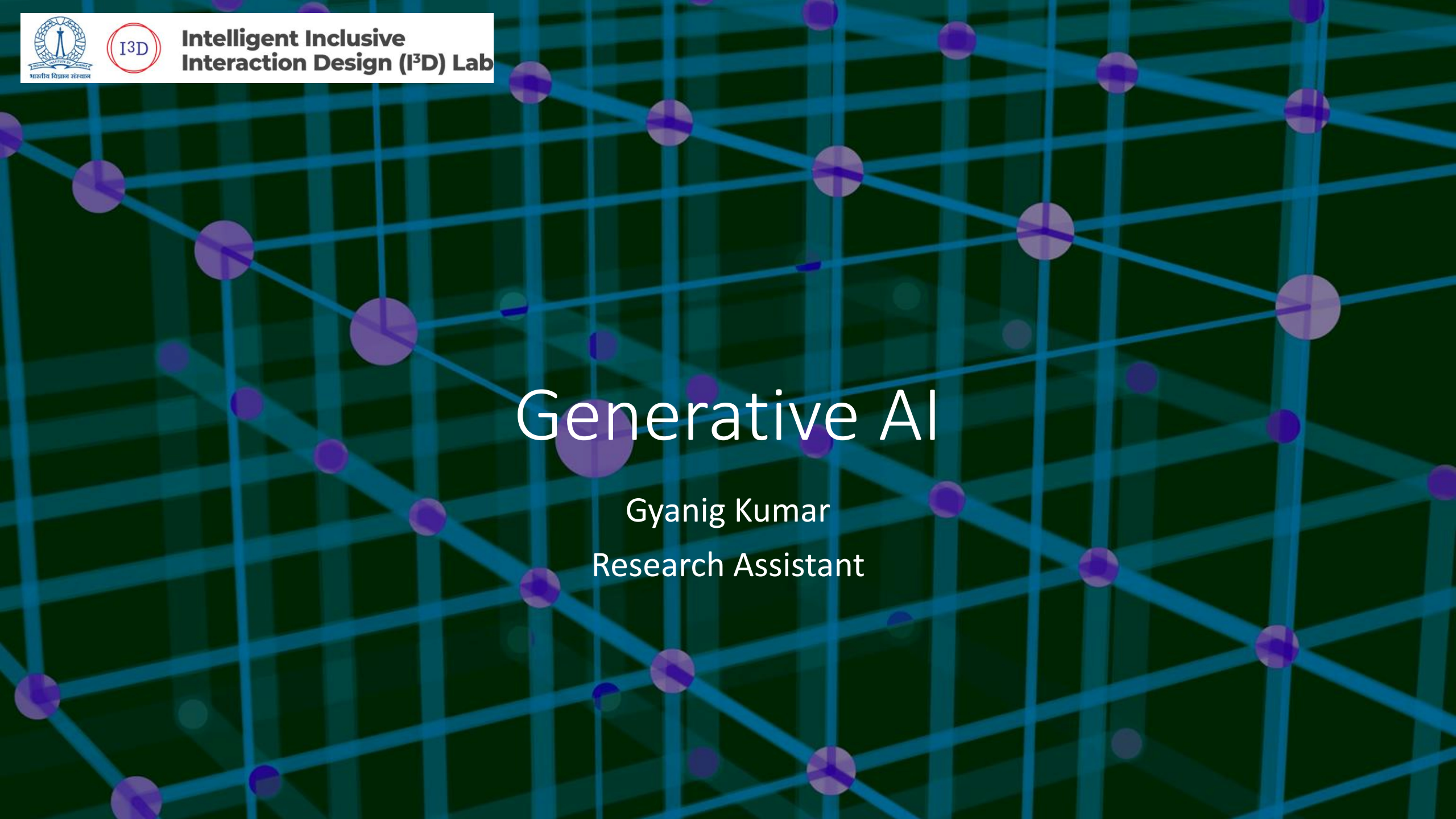


Generative AI

Gyanig Kumar

Research Assistant

Use-Case Example in a Typical Computer Vision Domain – Object Detection in Advanced Manufacturing Setup

Part I
Introduction to GAN

Introduction

- Generative adversarial networks (GANs) or Generative AI are an exciting recent innovation in machine learning.
- GANs are *generative* models:
  - Create new data instances that resemble your training data.
  - Achieve this level of realism by pairing a **generator** and with a **discriminator**

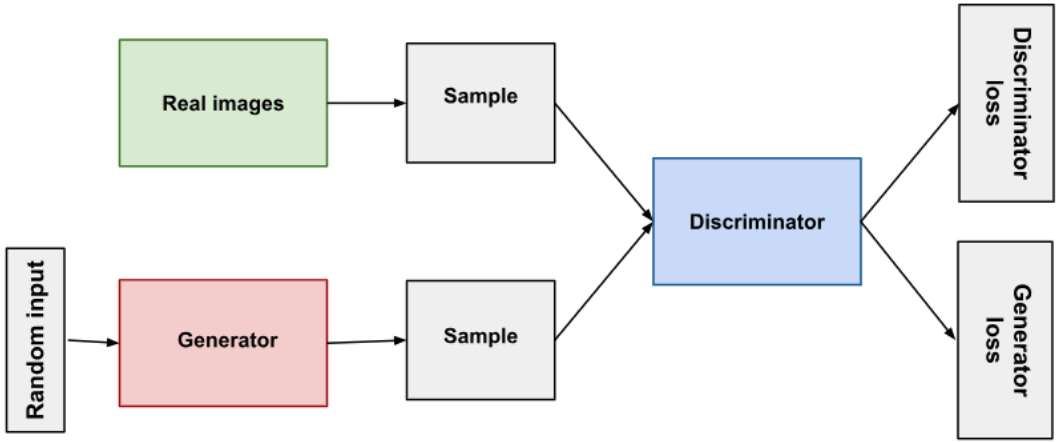
Fig. Images generated by a GAN (Source: NVIDIA)

Background: What is Generative Model?

- Supervised vs. Unsupervised Learning
- Examples of supervised learning problems include classification and regression, and examples of supervised learning algorithms include logistic regression and random forest.
- Examples of unsupervised learning problems include clustering and generative modeling, and examples of unsupervised learning algorithms are K-means and Generative Adversarial Networks.
- More formally, given a set of data instances X and a set of labels Y:
  - **Generative** models capture the joint probability p(X, Y), or just p(X) if there are no labels.
  - **Discriminative** models capture the conditional probability p(Y | X).

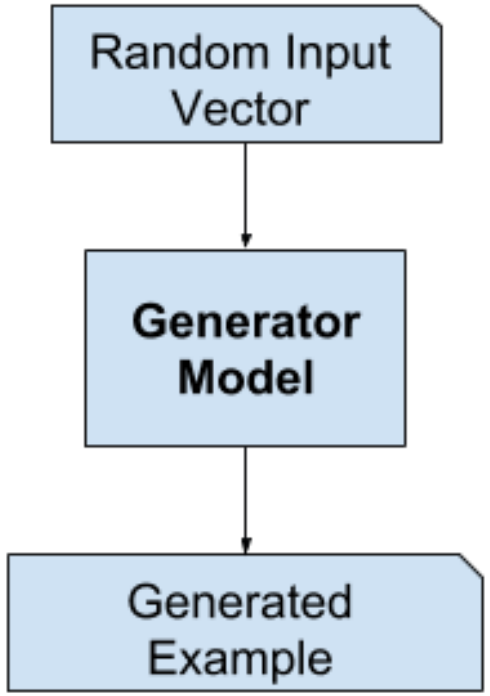
Generative Models Are Hard

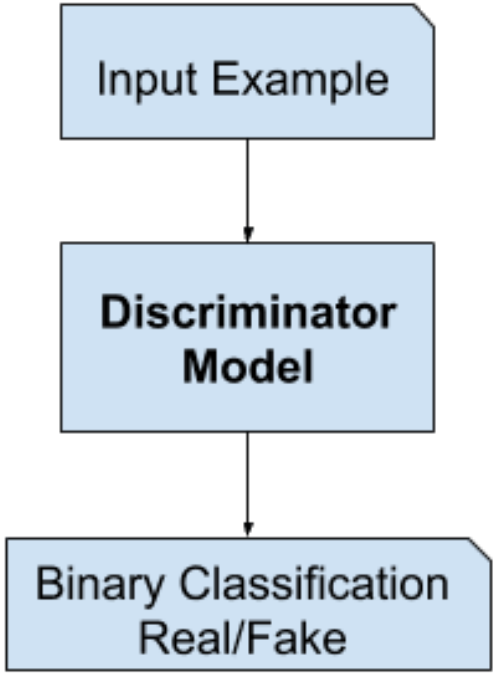
Overview of GAN Structure

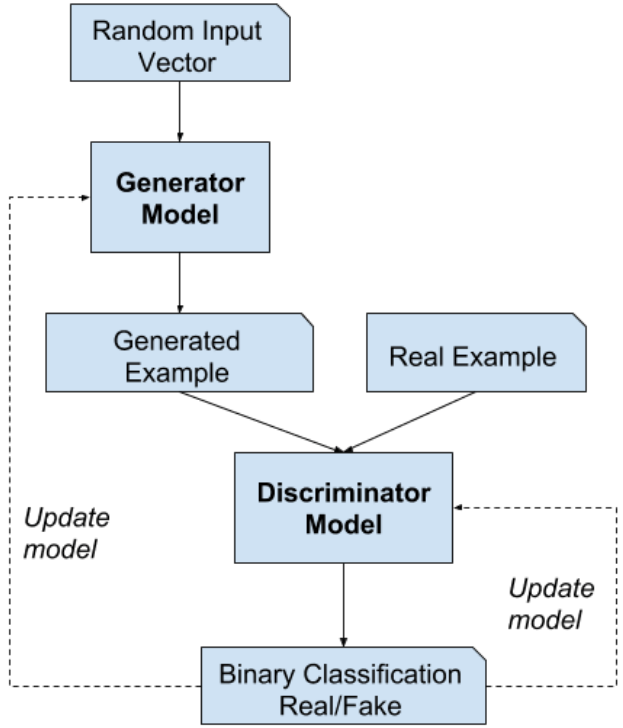


- **Generator** learns to generate plausible data. The generated instances become negative training examples for the discriminator.
- **Discriminator** learns to distinguish the generator's fake data from real data. The discriminator penalizes the generator for producing implausible results.
- Generator output is connected directly to the discriminator input.

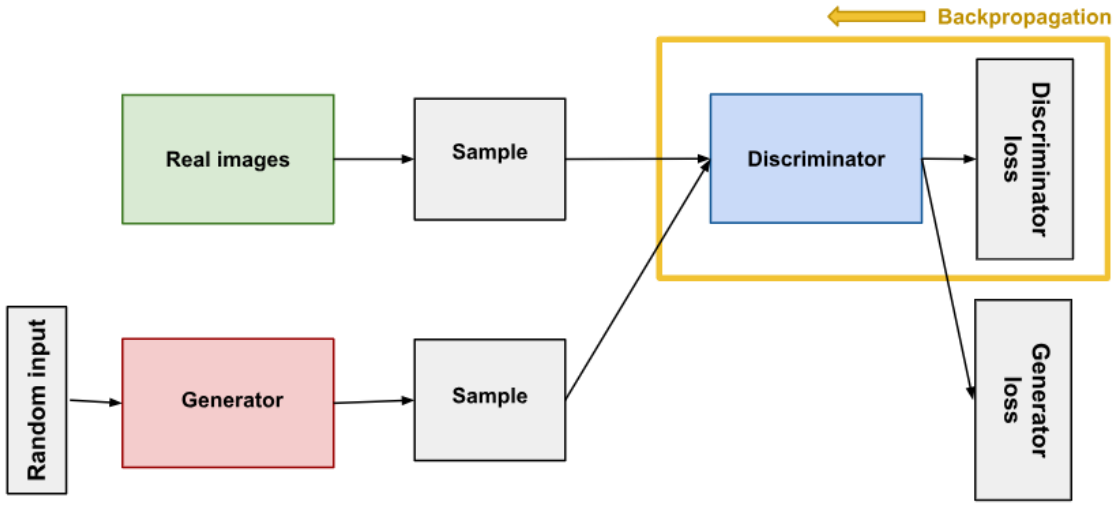
- [training] drives the discriminator to attempt to learn to correctly classify samples as real or fake.
- Simultaneously, the generator attempts to fool the classifier into believing its samples are real.
- At convergence, the generator's samples are indistinguishable from real data, and the discriminator outputs 1/2 everywhere.
- The discriminator may then be discarded.







Discriminator



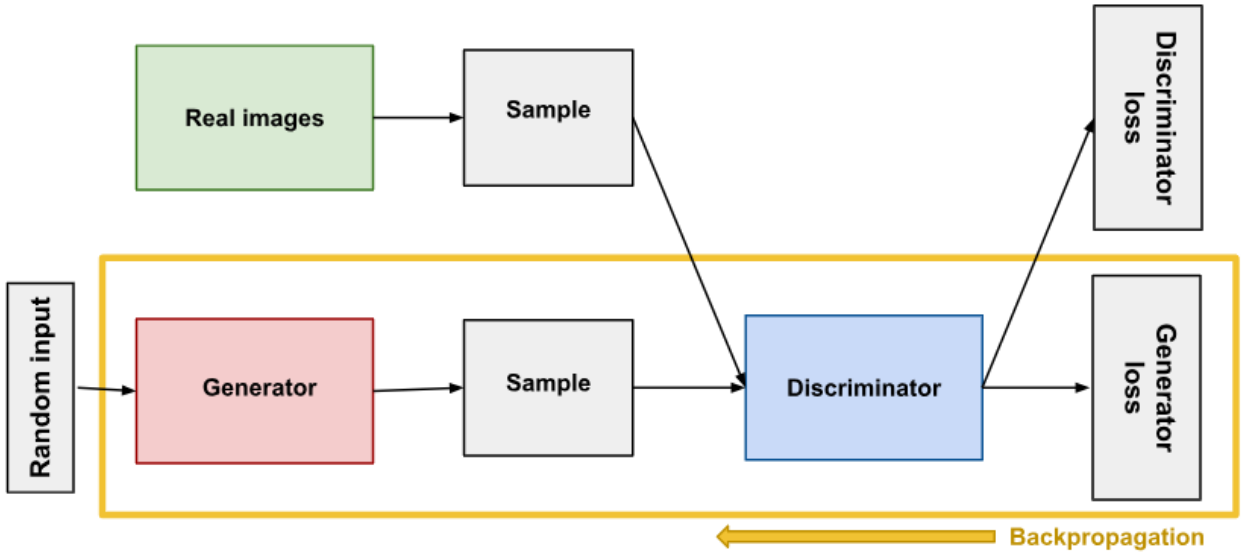
- Discriminator's training data comes from two sources:
  - **Real data** instances, such as real pictures of people.
  - **Fake data** instances created by the generator.
- During discriminator training:
  - The discriminator classifies both real data and fake data from the generator.
  - The discriminator loss penalizes the discriminator for misclassifying a real instance as fake or a fake instance as real.
  - The discriminator updates its weights through backpropagation from the discriminator loss through the discriminator network.

Discriminator can use any network architecture appropriate to the type of data it's classifying

Generator

The portion of the GAN that trains the generator includes:

- 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



Random input is:

- In its most basic form, a GAN takes random noise as its input
- Choose something that's easy to sample from, like a uniform distribution

Generating Random Variables

Computers are fundamentally deterministic

it is possible to define algorithms that generate sequences of numbers whose properties are very close to the properties of theoretical random numbers sequences

Methods: **inverse transform method, rejection sampling, Metropolis-Hasting algorithm** and others

Generating Random Variables

inverse transform method

- way to generate a random variable that follows a given distribution by making a uniform random variable goes through a well designed “transform function” (inverse CDF)

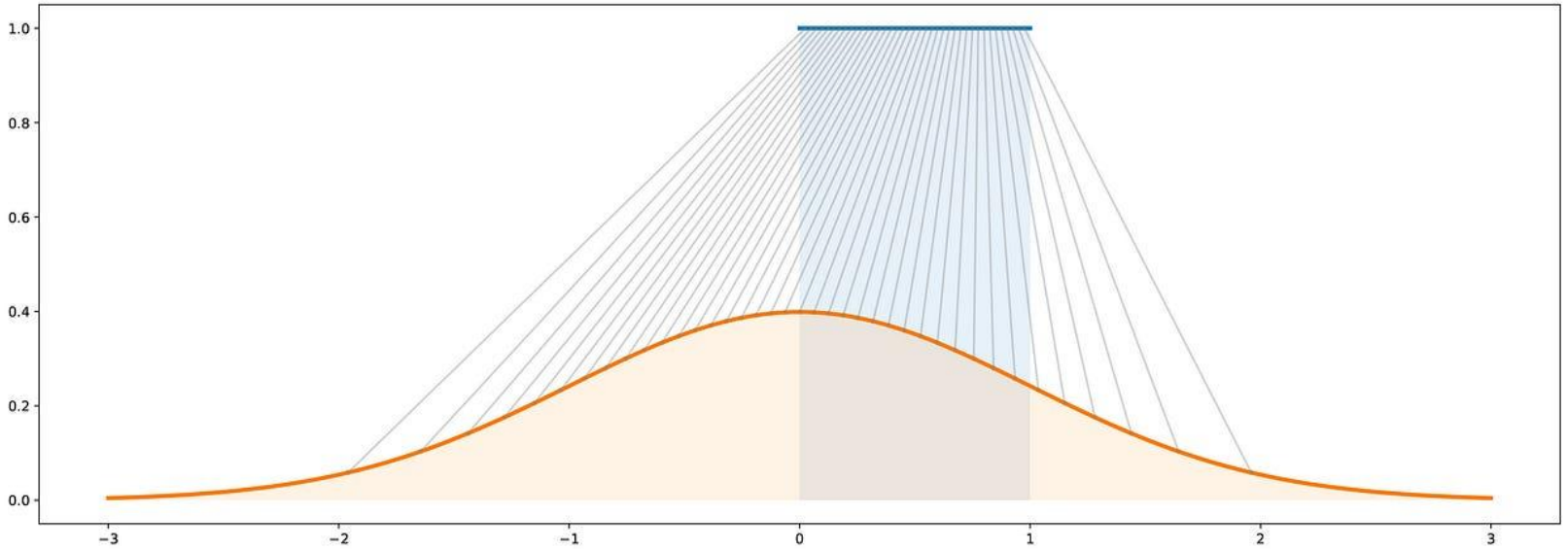
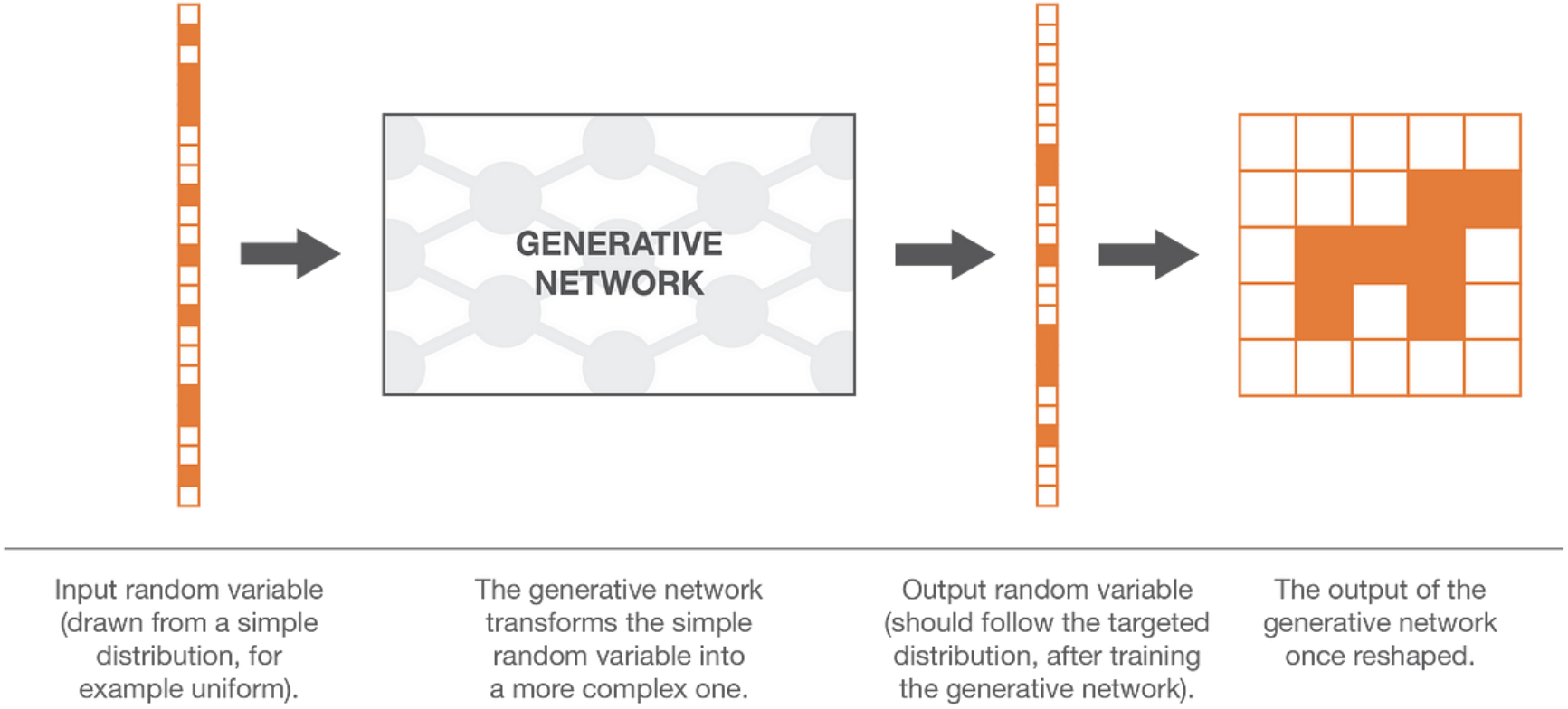


Illustration of the inverse transform method. In blue: the uniform distribution over [0,1]. In orange: the standard gaussian distribution. In grey: the mapping from the uniform to the gaussian distribution (inverse CDF).

Use the Discriminator to train Generator

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



Generative model approximation

- finding the transform function is not as straightforward
- complex function naturally implies neural network modelling
- model the transform function by a neural network
  - input : simple N dimensional uniform random variable
  - output : another N dimensional random variable that should follow complex probability distribution corresponding to desired output

Training GAN

- GANs must juggle two different kinds of training (generator and discriminator)
- GAN convergence is hard to identify

Alternate Training

- The discriminator trains for one or more epochs
- The generator trains for one or more epochs
- Repeat steps 1 and 2 to continue to train the generator and discriminator networks

Convergence

- As the generator improves with training, the discriminator performance gets worse
- This progression poses a problem
  - Discriminator feedback gets less meaningful over time

Loss Functions

minimax loss

$$E_x[\log(D(x))] + E_z[\log(1 - D(G(z))]$$

$\boldsymbol{D(x)}$: discriminator's estimate of the probability that real data instance x is real

$\boldsymbol{E_x}$: expected value over all real data instances

$\boldsymbol{G(z)}$: generator's output when given noise z

$\boldsymbol{D(G(z))}$: discriminator's estimate of the probability that a fake instance is real

$\boldsymbol{E_z}$: expected value over all generated fake instances G(z)

Generator can't directly affect the log(D(x))

- minimax loss function can cause the GAN to get stuck in the early stages
- modify the generator loss so that the generator tries to maximize log D(G(z))

Cont..

Loss Functions

Wasserstein loss (WGAN discriminator is actually called a "critic" instead of a "discriminator"**)**

Critic Loss: $D(x) - D(G(z))$

- discriminator tries to maximize the difference between its output on real instances and its output on fake instances.

Generator Loss: D(G(z))

- generator tries to maximize the discriminator's output for its fake instances.

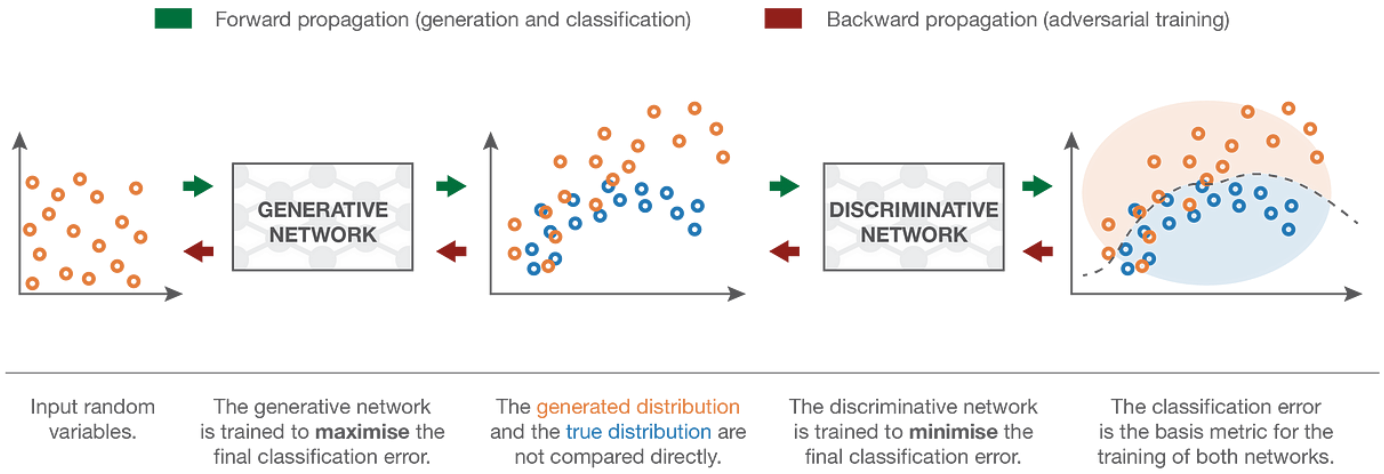
In these functions:

- D(x) is the critic's output for a real instance.
- G(z) is the generator's output when given noise z.
- D(G(z)) is the critic's output for a fake instance.
- The output of critic D does not have to be between 1 and 0.

The approximation: GAN

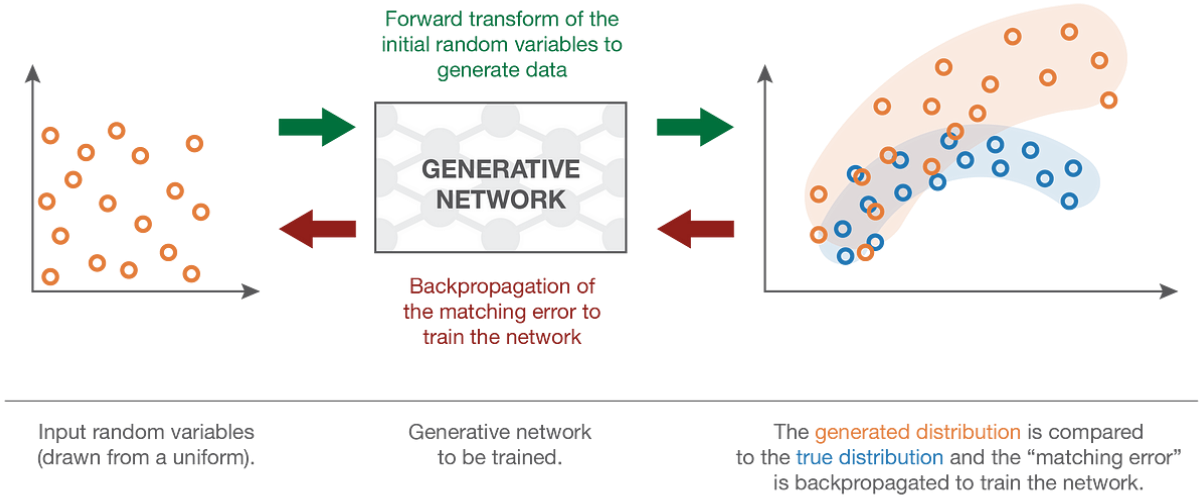
Two networks can then be trained jointly (at the same time) with opposite goals:

- the goal of the generator is to fool the discriminator, so the generative neural network is trained to **maximise the final classification error** (between true and generated data)
- the goal of the discriminator is to detect fake generated data, so the discriminative neural network is trained to **minimise the final classification error**



- **These opposite goals and the implied notion of adversarial training**
- **From a game theory point of view, we can think of this setting as a minimax two-players game**

Direct method



Comparing two probability distributions based on samples

- Maximum Mean Discrepancy (MMD) approach
  - defines a distance between two probability distributions

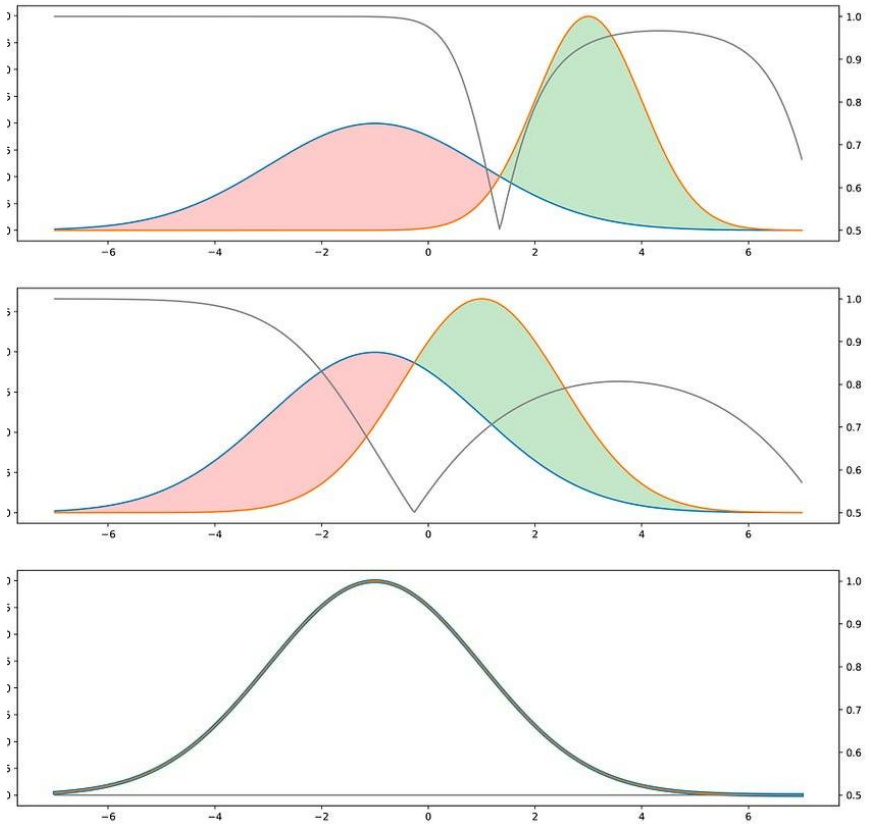
GMN optimize the network by

- generate some uniform inputs
- make these inputs go through the network and collect the generated outputs
- compare the true distribution and the generated one based on the available samples
- use backpropagation to make one step of gradient descent to lower the distance (for example MMD) between true and generated distributions

The “indirect” training method

downstream task of GANs is a discrimination task between true and generated samples [**Discriminator**]

If the two distributions are far apart, the discriminator will be able to classify easily



Intuition for the adversarial method. The blue distribution is the true one, the orange is the generated one. In grey, with corresponding y-axis on the right, we displayed the probability to be true for the discriminator if it chooses the class with the higher density in each point (assuming “true” and “generated” data are in equal proportions). The closer the two distributions are, the more often the discriminator is wrong. When training, the goal is to “move the green area” (generated distribution is too high) towards the red area (generated distribution is too low).

Type of GANs

- **Progressive GAN (Progressive Growing of GANs for Improved Quality, Stability, and Variation)**
  - generator's first layers produce very low-resolution images, and subsequent layers add details
  - Training is quicker
  - Produce higher resolution images

- **Conditional GAN (Conditional Generative Adversarial Nets)**
  - train on a labeled data set and let you specify the label for each generated instance
  - a conditional MNIST GAN would let you specify which digit the GAN should generate
  - Instead of modeling the joint probability P(X, Y), conditional GANs model the conditional probability P(X | Y)

Type of GANs

- **Image-to-Image Translation (Image-to-Image Translation with Conditional Adversarial Networks)**
  - take an image as input and map it to a generated output image with different properties
  - take sketches of objects and turn them into photorealistic images of that object

- **CycleGAN (Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks)**
  - transform images from one set into images that could plausibly belong to another set
  - training data for the CycleGAN is simply two sets of images
  - requires no labels or pairwise correspondences between images

Type of GANs

- **Text-to-Image Synthesis (StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks)**
  - take text as input and produce images that are plausible and described by the text
  - take sketches of objects and turn them into photorealistic images of that object
- **Super-resolution (Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network)**
  - increase the resolution of images, adding detail where necessary to fill in blurry areas

Part II
Synthetic Data

Use-Case Example in a Typical Computer Vision Domain – Object Detection in Advanced Manufacturing Setup

Generation of Synthetic Data

The Use-Cases of GANs

- Virtual Environments
- Style Transfer
- Object Detection
- 3D Object Generation
- Data Imbalance Correction
- Image Augmentation

Intelligent Inclusive Interaction Design (I³D) Lab

CycleGAN

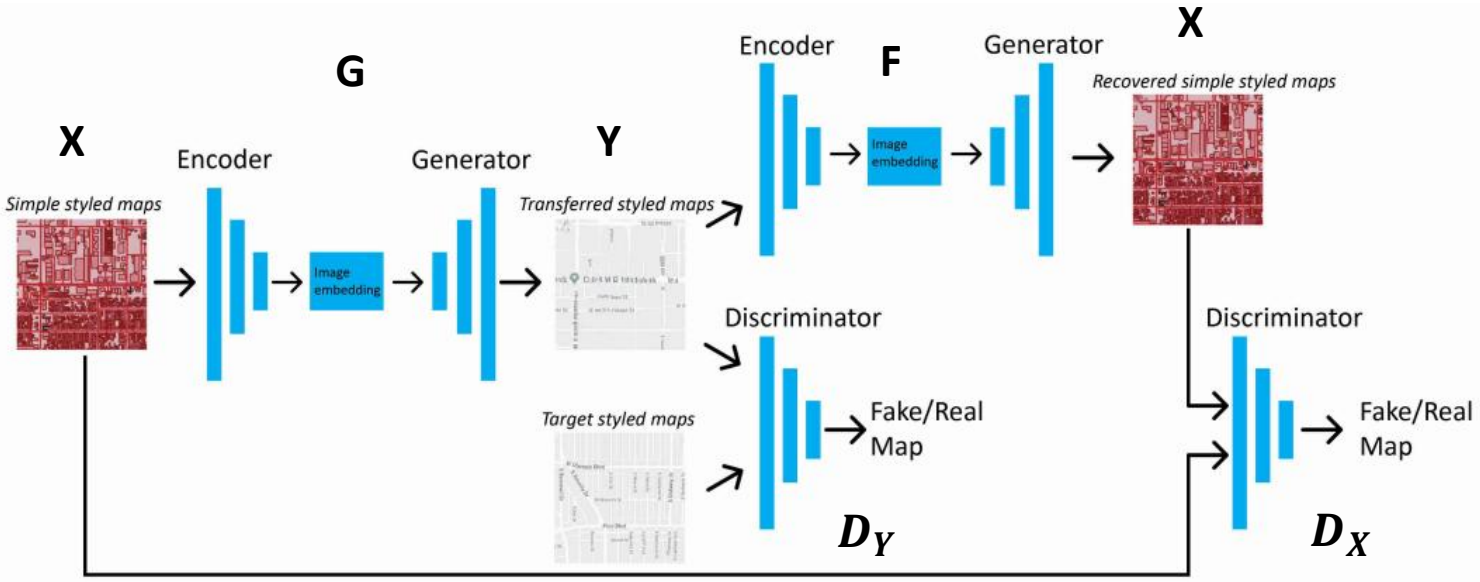
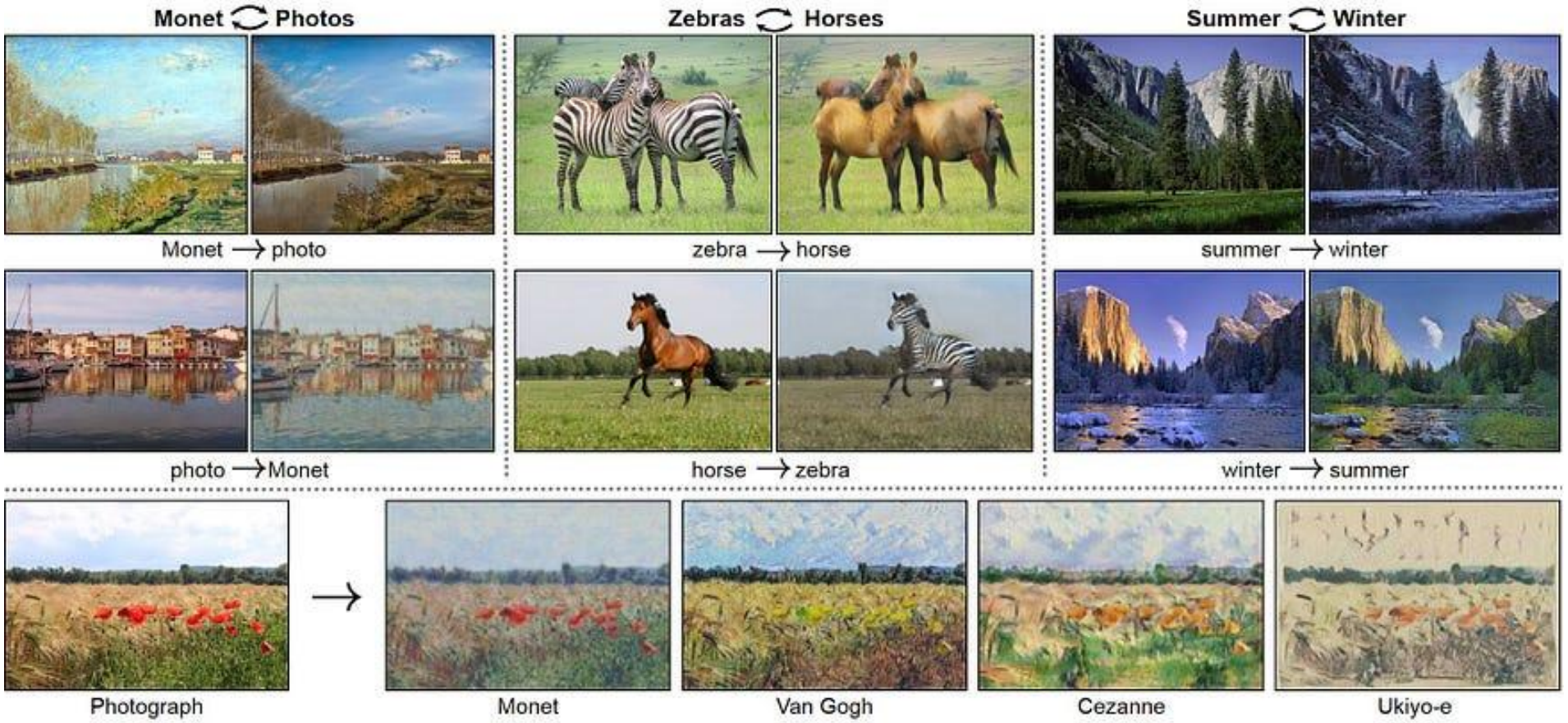


Figure. Overview of CycleGAN architecture: Translating from satellite image to map routes domain

Our goal is to learn mapping functions between two domains X and Y given training samples $\{x_i\}_{i=1}^{N}$ where $x_i \in X$ and $\{y_j\}_{j=1}^{M}$ where $y_j \in Y$[1]. We denote the data distribution as $x \sim p_{data}(x)$ and $y \sim p_{data}(y)$. As illustrated in Figure 3 (a), our model includes two mappings $G : X \rightarrow Y$ and $F : Y \rightarrow X$. In addition, we introduce two adversarial discriminators D_X and D_Y, where D_X aims to distinguish between images $\{x\}$ and translated images $\{F(y)\}$; in the same way, D_Y aims to discriminate between $\{y\}$ and $\{G(x)\}$. Our objective contains two types of terms: *adversarial losses* [16] for matching the distribution of generated images to the data distribution in the target domain; and *cycle consistency losses* to prevent the learned mappings G and F from contradicting each other.

Examples


Monet ⇄ Photos
Zebras ⇄ Horses
Summer ⇄ Winter
Monet → photo
zebra → horse
summer → winter
photo → Monet
horse → zebra
winter → summer
Photograph
Monet
Van Gogh
Cezanne
Ukiyo-e

Cycle Consistency Loss

Objective

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) =& \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ &+ \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ &+ \lambda \mathcal{L}_{\text{cyc}}(G, F), \end{aligned} \quad (3)$$

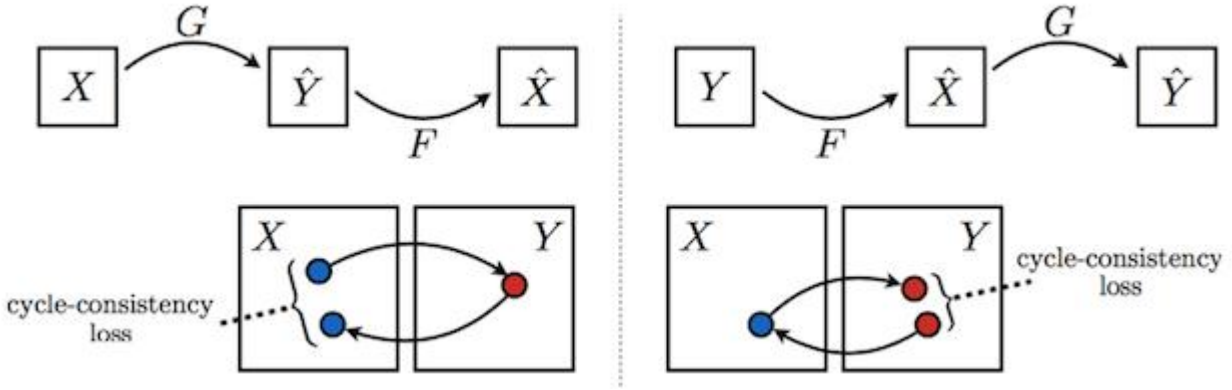
where λ controls the relative importance of the two objectives. We aim to solve:

$$G^*, F^* = \arg\min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y). \quad (4)$$

(b), for each image x from domain X, the image translation cycle should be able to bring x back to the original image, i.e., $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$. We call this *forward cycle consistency*. Similarly, as illustrated in Figure 3 (c), for each image y from domain Y, G and F should also satisfy *backward cycle consistency*: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$. We incentivize this behavior using a *cycle consistency loss*:

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G, F) =& \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] \\ &+ \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1]. \end{aligned} \quad (2)$$

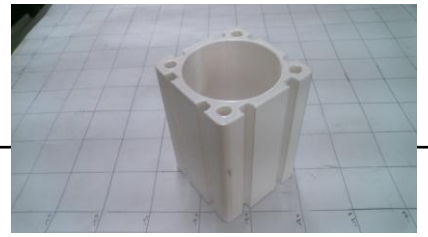
In preliminary experiments, we also tried replacing the L1 norm in this loss with an adversarial loss between $F(G(x))$ and x, and between $G(F(y))$ and y, but did not observe improved performance.

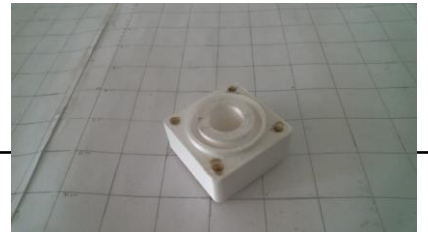


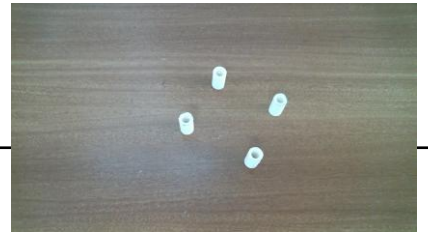
Intelligent Inclusive Interaction Design (I³D) Lab

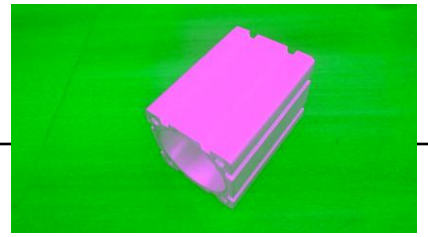
Training Dataset

3D Printed Specimen Replica (Class A)

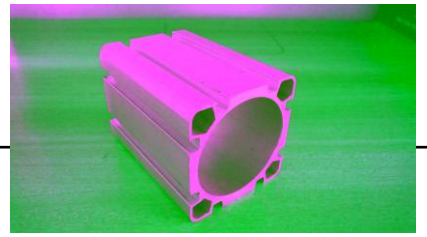
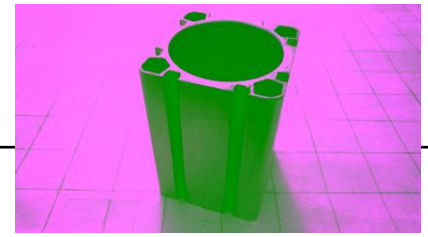






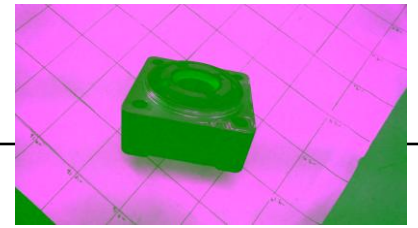



Metal Specimen (Class B)

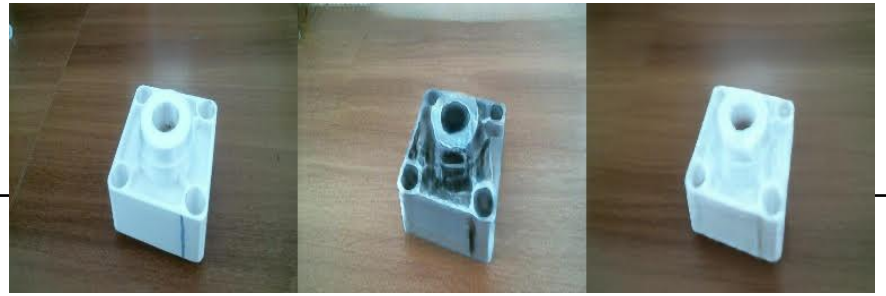






Results of CycleGAN

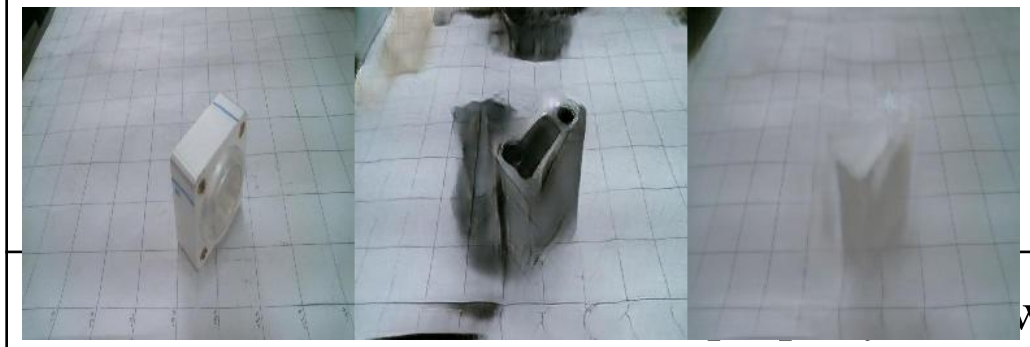
Class A to Class B (Success)



Results of CycleGAN

Class A to Class B (Failures)

reference generated



contrast between the 3D specimen and the background (left) and in environments with high light intensities causing the objects to have a shiny surface while capturing images, due to which the model is not able replicate the expected texture and color for the generated object even though the generated shape matches (right).

Intelligent Inclusive
Interaction Design (I³D) Lab

Thank You.

By Gyanig Kumar
Research Assistant, I3D Lab, IISc