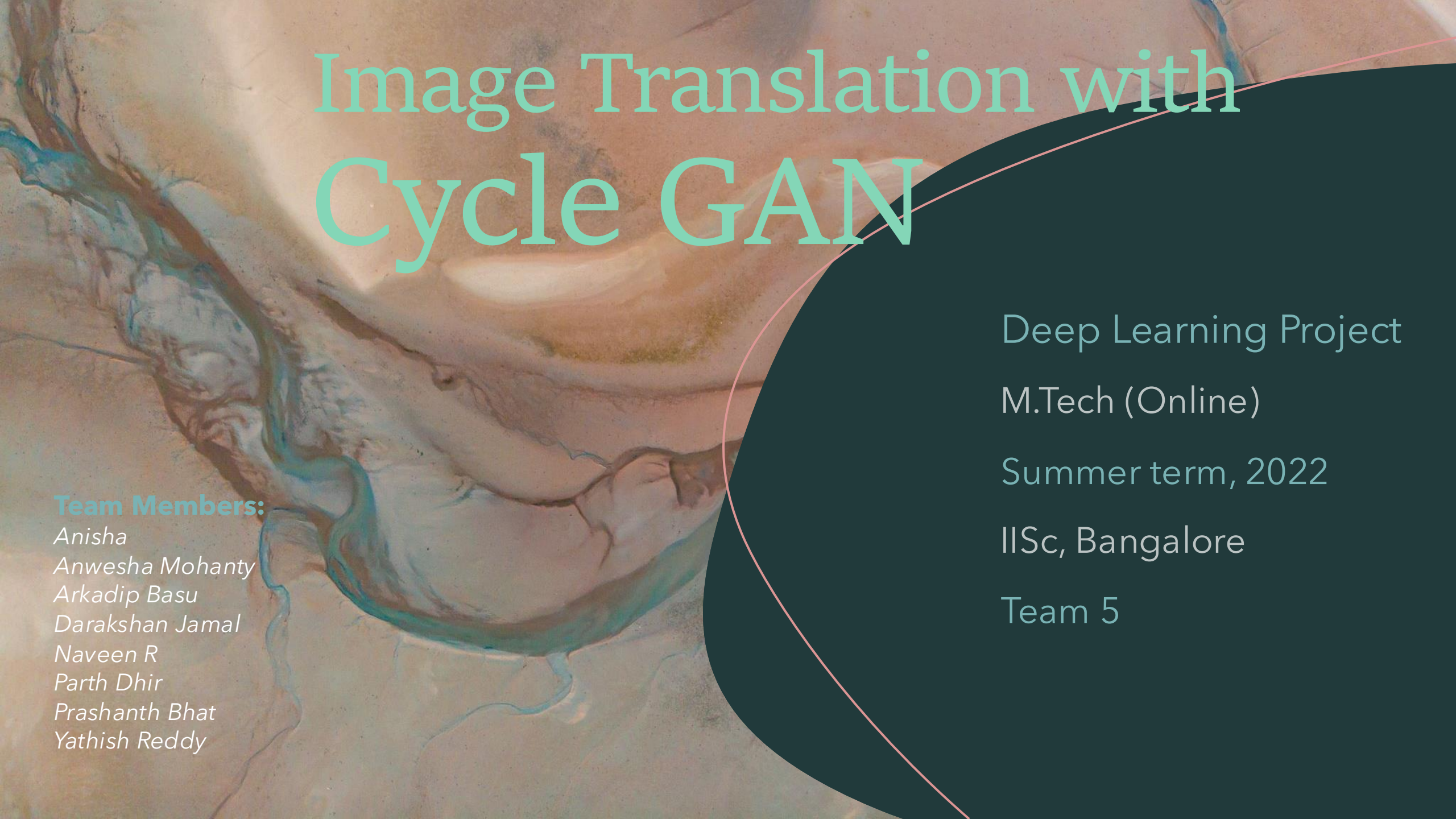


Image Translation with Cycle GAN

Deep Learning Project

M.Tech (Online)

Summer term, 2022

IISc, Bangalore

Team 5

Team Members:

Anisha
Anwesha Mohanty
Arkadip Basu
Darakshan Jamal
Naveen R
Parth Dhir
Prashanth Bhat
Yathish Reddy

What is GAN?

- Generative Adversarial Network (GAN) is a generative modelling approach that learns the underlying distribution of data and generates new set of data similar to the original one.
- GANs can be used to imitate any data distribution like image, text, sound etc.
- Consists of two neural networks – Generators and Discriminators
- Model is trained in an adversarial setting.
- Applications:
  - Generate new data from available data.
  - Text to Image generation.
  - Low resolution to high resolution.
  - Not limited to only images, GANs can generate raw audio from a corporal of speech data.
  - Generate music by using clone voice.

Image to Image Translation: PIX2PIX GAN

- Takes Image as input and maps it to a generated output image with different properties
- The system requires a pairwise correspondence between the images during training
- Examples - Black & White to colour images, Hand drawn purse to actual purse image, etc

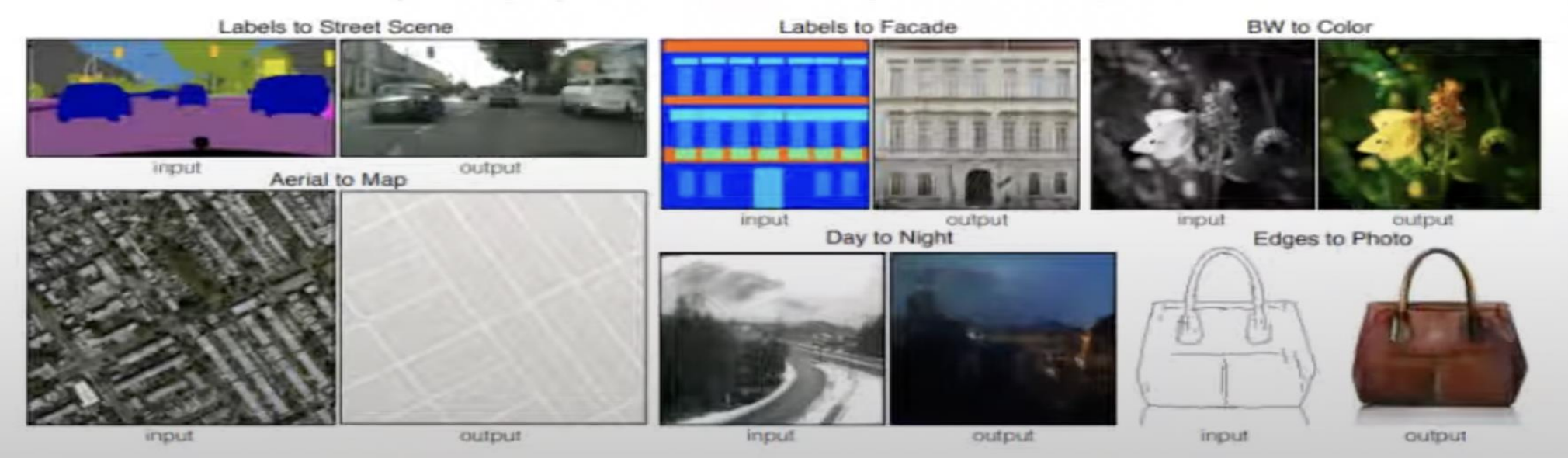
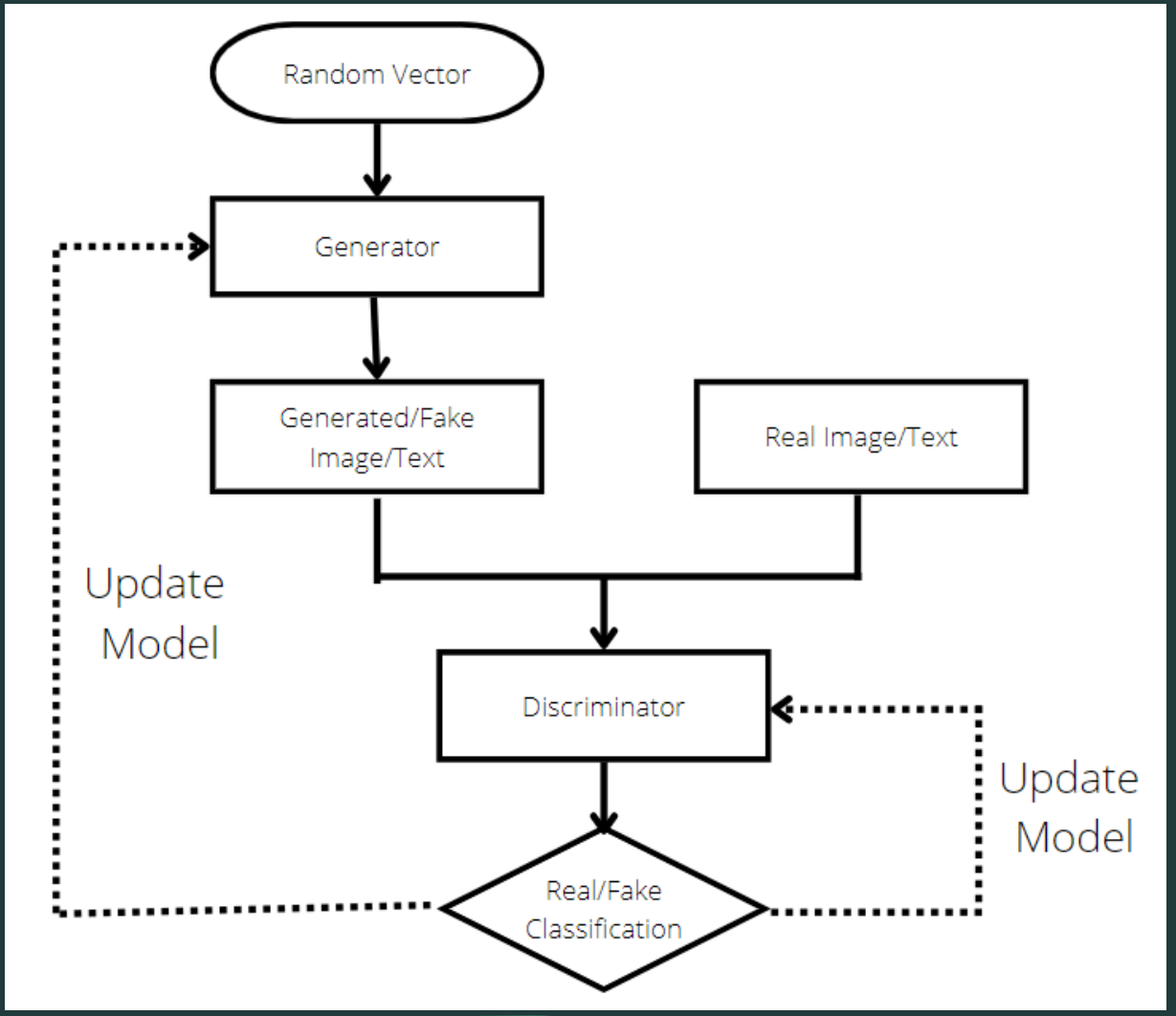


Image from Original paper

How GANs work?

- Generator creates a fake example using a random vector input.
- Fake examples are mixed with Real examples.
- Discriminator classifies the example as real/fake.
- Generator is trained to generate more realistic examples.
- Discriminator is trained to classify real/fake more accurately
- Pix2pix, CycleGAN, StyleGAN, DCGAN, DiscoGAN, TextToImage, IsGAN are some major types of GAN
- Applications: Concept to Reality, Satellite image to MAP, Medical Imaging, Drug discovery Molecular biology



Cycle GAN

- Cycle GAN translates an image from a source domain to a target domain in the absence of paired examples using cycle-consistent adversarial networks.

- It can be used when paired training data does not exist, which includes collection style transfer, object transfiguration, season transfer, photo enhancement.

- Training data for Cycle GAN requires two sets of images. The system requires no labels or pairwise correspondence between images for training

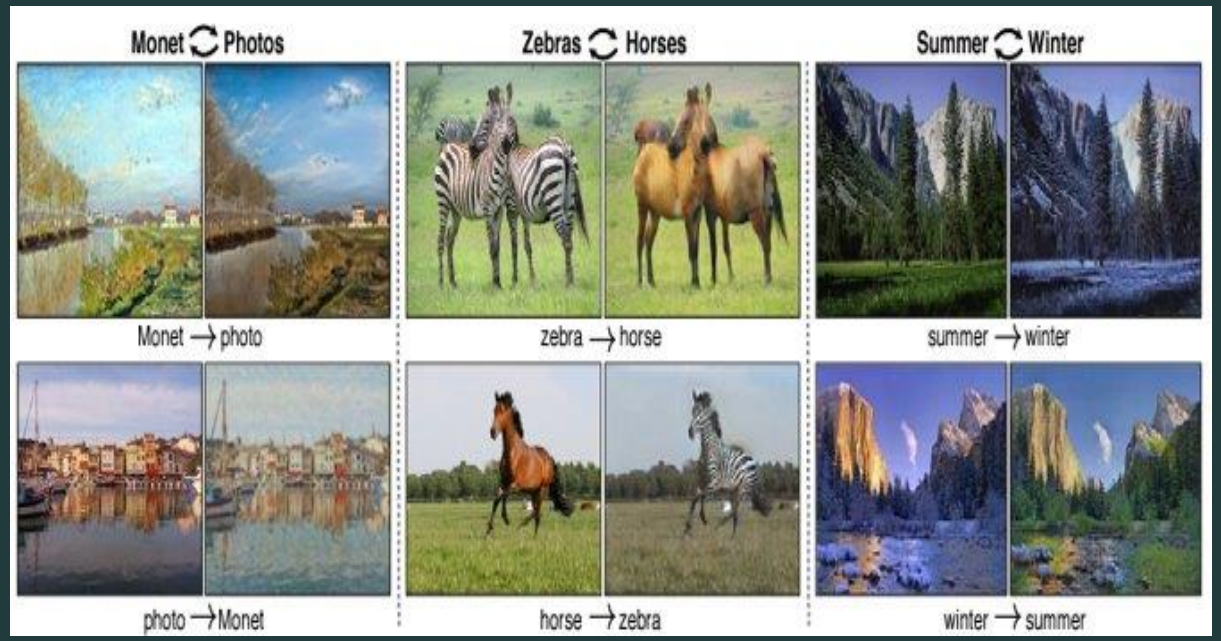


Image from Original paper

Objective

- Through Cycle GAN, we aim to transform Horse image into Zebra and vice versa.
- We will also demonstrate the power of cycle GAN in collection style transfer

End to End Model Overview

Download Data

Image Preprocessing(Rotate ,Flip ,Crop)

Train/Test/Val Split

Image Normalization

Define Classes for Generator ,Discriminator , Resnet ,Losses

EPOCHS = 100+

Generate fake images from generator

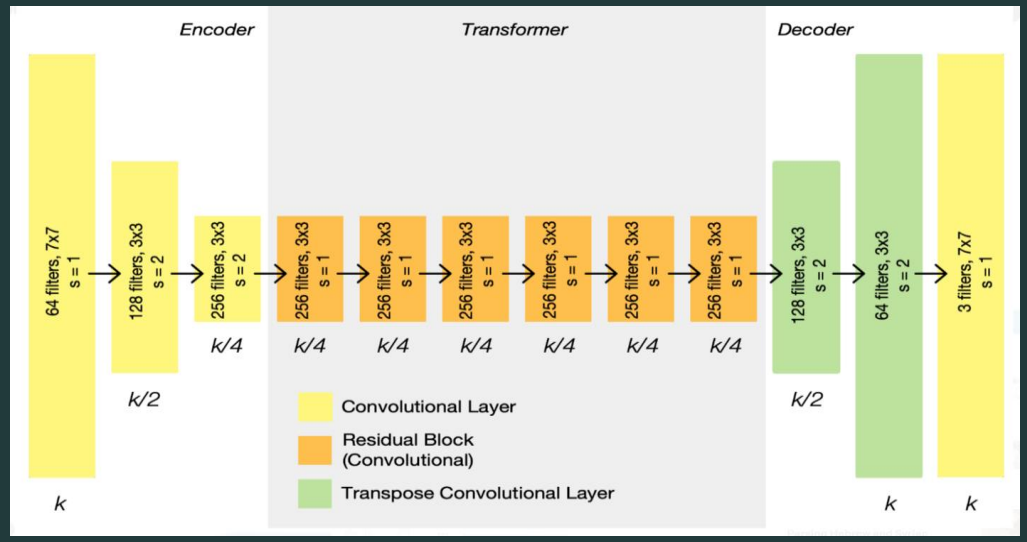
Equal Samples of Real +Fake Images Fed to train only discriminator

Train the GAN with losses keeping Discriminator freeze

Evaluate performance with Test Images

Generator Architecture

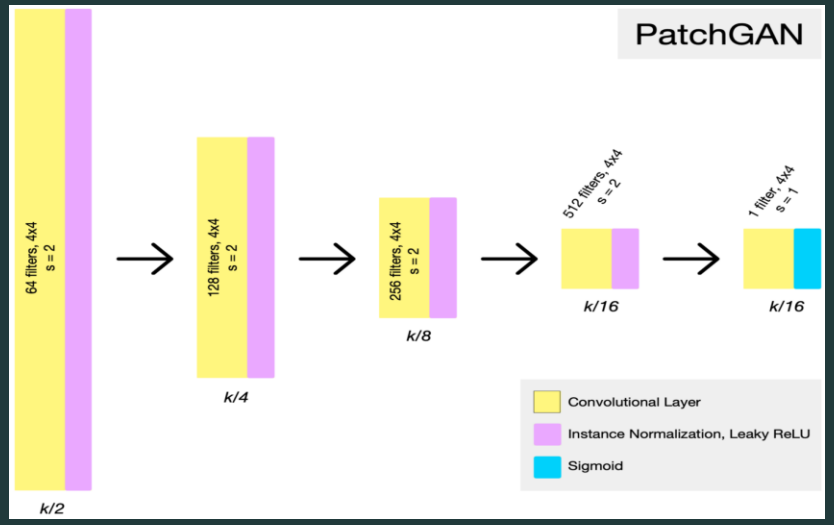
```
Model: "Generator"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
gen_input (InputLayer)       [(None, 128, 128, 3)]     0

Conv2D_7x7_S1_IN (Sequentia  (None, 128, 128, 64)      9664
l)

Conv2D_3x3_S2_1 (Sequential  (None, 64, 64, 128)       74240
)

Conv2D_3x3_S2_2 (Sequential  (None, 32, 32, 256)       295936
)

ResBlock_3x3_S1_1 (Function  (None, None, None, 256)   1181696
al)

ResBlock_3x3_S1_2 (Function  (None, None, None, 256)   1181696
al)

ResBlock_3x3_S1_3 (Function  (None, None, None, 256)   1181696
al)

ResBlock_3x3_S1_4 (Function  (None, None, None, 256)   1181696
al)

ResBlock_3x3_S1_5 (Function  (None, None, None, 256)   1181696
al)

TConv2d_3x3_S2_1 (Sequentia  (None, 64, 64, 128)       295424
l)

TConv2d_3x3_S2_2 (Sequentia  (None, 128, 128, 64)      73984
l)

Conv2D_7x7_S1_OUT (Sequenti  (None, 128, 128, 3)       9420
al)

=================================================================
Total params: 6,667,148
Trainable params: 6,660,742
Non-trainable params: 6,406
_________________________________________________________________
```



- Cycle GAN generator has 3 sections: Encoder, Transformer & Decoder
- Representation size shrinks in the encoder phase, stays constant in the transformer phase, and expands again in the decoder phase.
- Each layer is followed by an instance normalization and ReLU activation (except the last layer which uses tanh and is not normalized)
- Architecture: *c7s1-64, d128, d256, R256, R256, R256,mR256, R256, R256, u128, u64, c7s1-3*

Discriminator Architecture

```
Model: "Discriminator"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dsc_input (InputLayer)      [(None, 128, 128, 3)]     0

 Conv2D_4x4_S2_IN (Sequentia  (None, 64, 64, 64)       3072
 l)

 Conv2D_4x4_S2_1 (Sequential  (None, 32, 32, 128)      131584
 )

 Conv2D_4x4_S2_2 (Sequential  (None, 16, 16, 256)      525312
 )

 Conv2D_4x4_S2_3 (Sequential  (None, 8, 8, 512)        2099200
 )

 Conv2D_4x4_S1_OUT (Conv2D)  (None, 8, 8, 1)           8192

=================================================================
Total params: 2,767,360
Trainable params: 2,765,568
Non-trainable params: 1,792
_________________________________________________________________
```



- Cycle GAN discriminator is a CNN that looks at “patches” of the input image, and outputs the probability of the patch being “real/fake”
- **Each layer is followed by an instance normalization and leaky ReLU activation (except the last layer which uses sigmoid and is not normalized)**
- The discriminator classifies overlapping patches of an image as real/fake instead of one prediction for entire image
- This reduces number of trainable parameters and enables discriminator to work on images arbitrary size
- Architecture: *C64-C128-C256-C512*

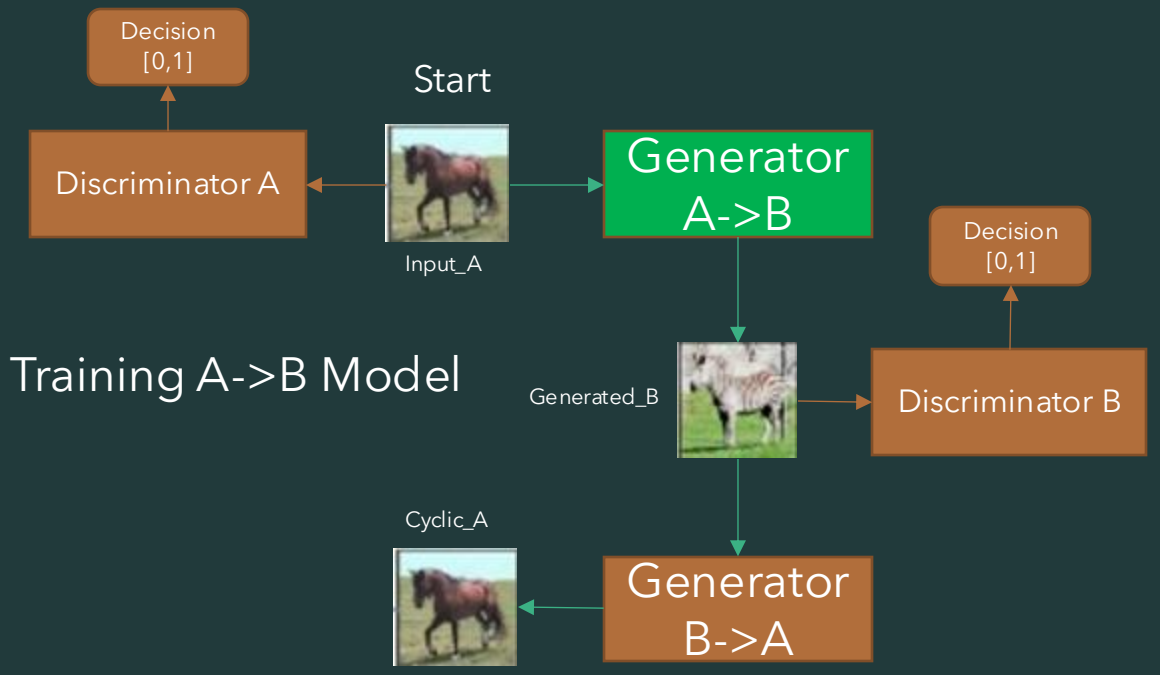
Combined Model

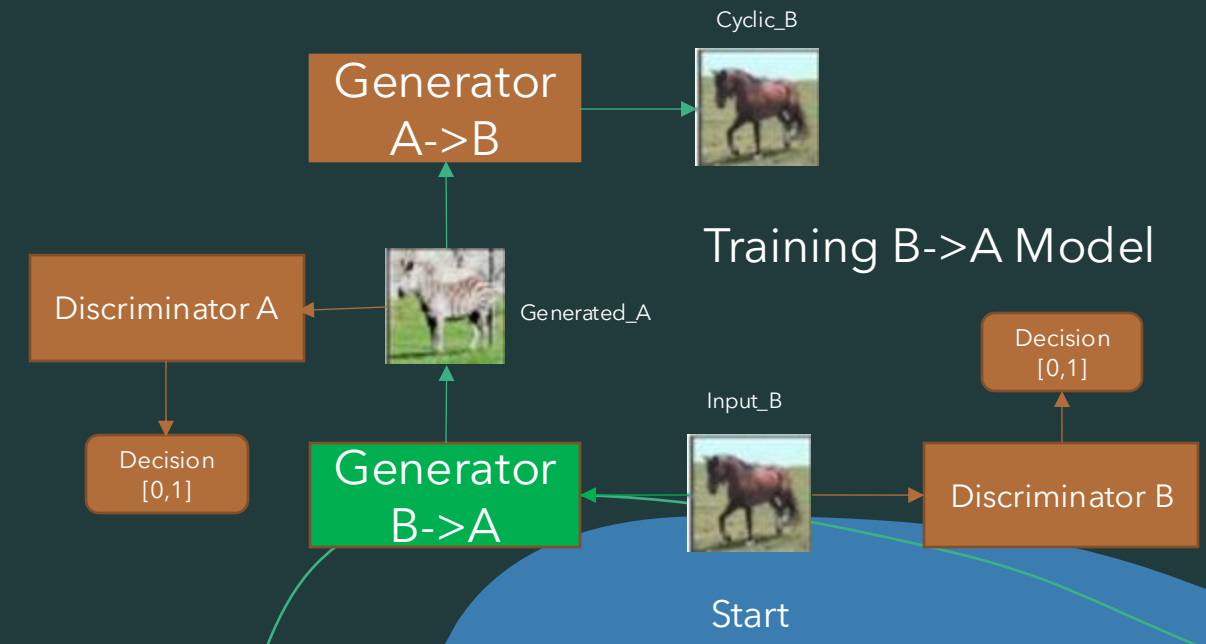
- Define two composite models (Discriminators + Generators), corresponding to A -> B and B -> A, to update generators by adversarial and cycle loss.

- For A -> B model:
  - Set only Generator corresponding to A -> B as trainable.
  - The Discriminators and the other generator will be marked as not trainable.

- Similarly for B -> A model, only generator B ->A will be trainable.

- The generators will be trained by setting all the labels (fake) to generated images as real and aiming to reduce the loss between the discriminator prediction and the fake labels.





Instance Normalization

- Also called contrast normalization
- Instance Normalization performs intensity normalization across the width and height of a single feature map of a single example
- Used in Style Transfer
- Gamma and beta are learnable parameters (per feature map)

$$\textit{Calculate mean and variance}$$

$$mean = \frac{1}{height \times width} \sum_{r=1}^{height} \sum_{c=1}^{width} map[r, c]$$

$$var = \frac{1}{height \times width} \sum_{r=1}^{height} \sum_{c=1}^{width} (map[r, c] - mean)^2$$

$$\textit{Normalize Feature Map}$$

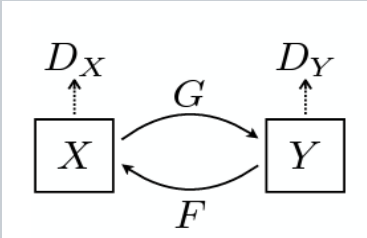
$$\hat{map}[r, c] = \frac{map[r, c] - mean}{\sqrt{var + \epsilon}}$$

$$\textit{Scale and Shift}$$

$$\gamma \times \hat{map}[r, c] + \beta$$

Loss types

Adversarial Loss

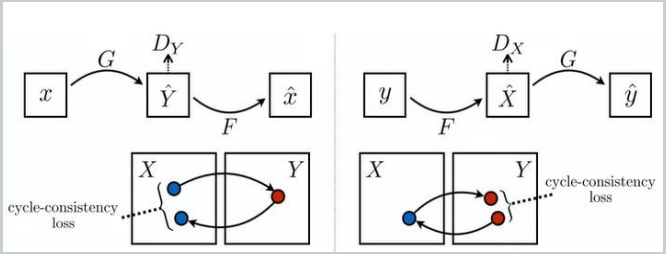


$$Loss_{adv}(G, D_y, X) = \frac{1}{m}\sum_{i=1}^{m}(1 - D_y(G(x_i)))^2$$

$$Loss_{adv}(F, D_x, Y) = \frac{1}{m}\sum_{i=1}^{m}(1 - D_x(F(y_i)))^2$$

- Loss generated when an image is classified incorrectly as fake/real
- Generator tries to minimize this loss and Discriminator tries to maximize it

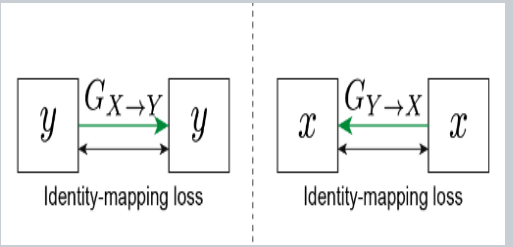
Cycle consistency loss



$$Loss_{cyc}(G, F, X, Y) = \frac{1}{m}\sum_{i=1}^{m}[F(G(x_i)) - x_i] + [G(F(y_i)) - y_i]$$

- Also called reconstruction loss
- It can be forward or backward
- This loss ensures that generators don't change parts of the input image unnecessarily
- F(G(image)) ≈ image

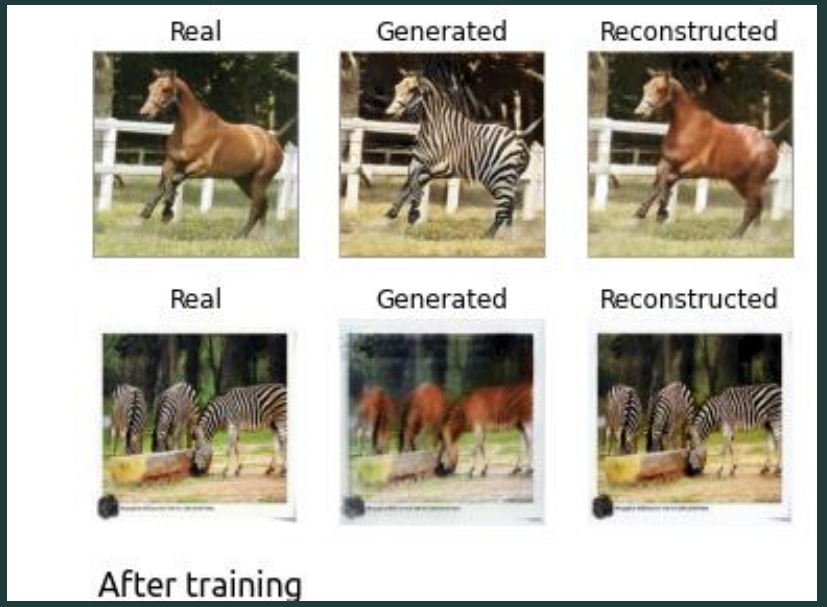
Identity Loss



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

- This makes sure that the generator doesn't change the color of input image too much
- Used with higher weight in tasks such as picture to painting conversion

Results: Horse <-> Zebra



Transformation with epochs

Epoch 20

Epoch 40

Epoch 70

Results: Photo to Monet

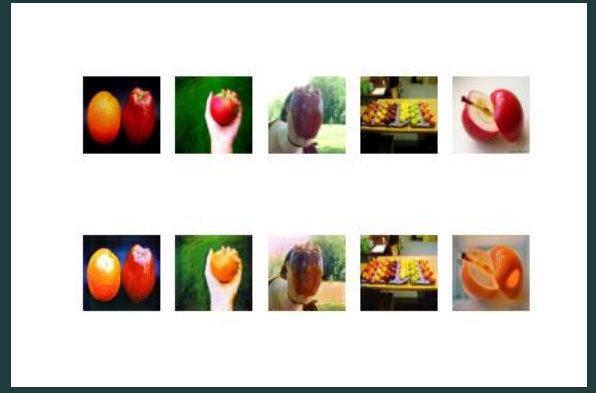
Results: Apple to Orange

Real

Fake

Epoch 5

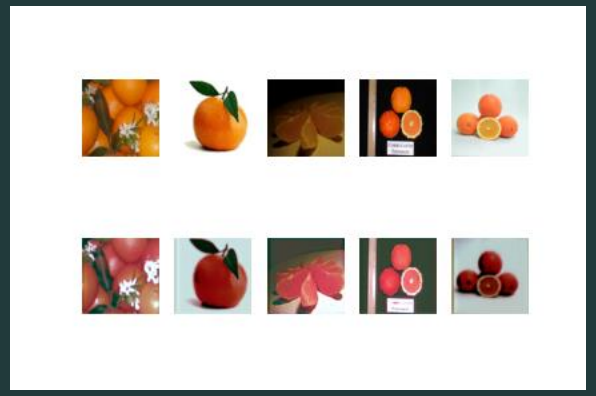
Epoch 15

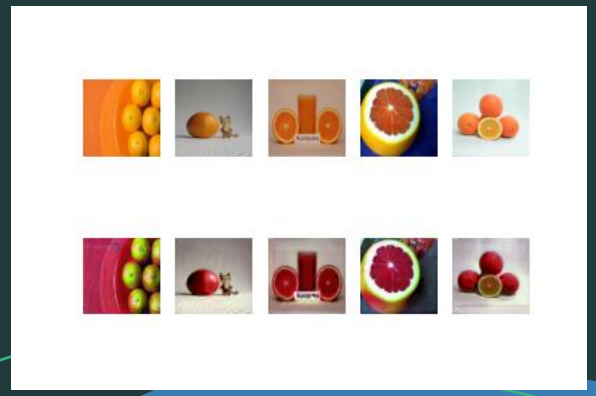


Epoch 25

Real

Fake

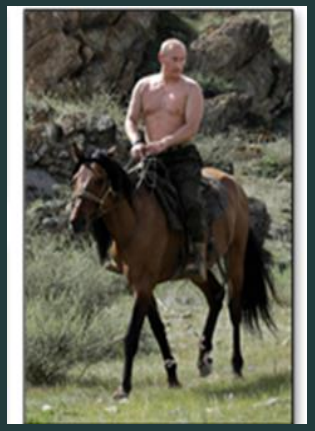


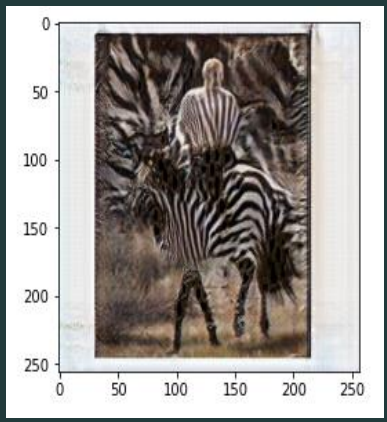


Cycle GAN Limitation

- Cycle GAN works well on tasks that involve colour or texture changes, like day-to-night photo translations, photo-to-painting tasks. However, tasks that require **substantial geometric changes** to the image, such as cat-to-dog translations, usually fail.
- Failure cases are seen with distribution characteristics of the training datasets.

On a lighter note !





Picture to Painting Conversion

- During Training we observed that model didn't interpret clouds & night images that well.
- It showed pretty decent results on images having sunset ,roads & trees.
- Also, if the images have ununiform bright patches , conversion to painting is not that well

GAN Problems – Why Training GAN is difficult

- **Non-convergence**: the model parameters oscillate, destabilize and never converge (NASH Equilibrium, Goodfellow)
- **Mode collapse**: The generator collapses which produces limited varieties of samples
- **Vanishing gradient**: the discriminator gets too successful that the generator gradient vanishes and learns nothing,
- Highly sensitive to the hyperparameter selections.
- GANs are on its highest pace of development .Nonetheless, in many cases completely unpaired data is plentifully available and should be made use of. It pushes the boundaries of what is possible in “unsupervised setting".

Thank you!!!

Colab Link

(This Colab notebook currently implements Photo --> Painting CycleGAN. It can be easily modified and trained to translate images between any two domains (Horse – Zebra, Apple – Orange etc)

Question & Feedback