**Presented by:**

**Oluwatoyin Sanni - 1871835**

**Mohammed Sukhsarwala - 1848668**


SAPIENZA
UNIVERSITÀ DI ROMA

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

**ELECTIVE IN AI-1**

# INTRODUCTION

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.

Applications:

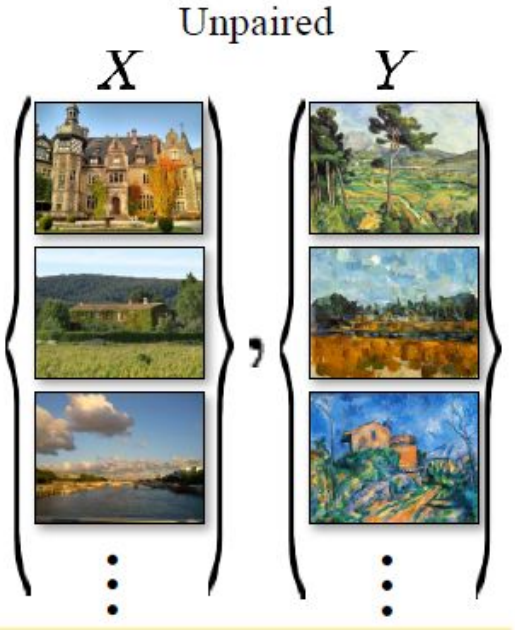
- Collection Style Transfer
- Object Transfiguration
- Season Transfer
- Photo Enhancement

# REASONS FOR CHOOSING THIS PAPER

Image To Image Translations

Approach of learning to translate an image from a X domain to Y domain **without paired examples.**

Learning to capture special characteristics of one image COLLECTION and then translating it to another image COLLECTION, in absence of PAIRED EXAMPLES




# UNPAIRED IMAGES

Obtaining Paired Training data can be difficult and expensive.

Perform mapping of ‘X’ Image Collection to ‘Y’ Image Collection, such that output image is indistinguishable from the images of ‘Y’.

# CYCLE CONSISTENCY

If we translate a sentence from English to French, then again translating that sentence back to English, should give me back the original sentence.

# NETWORK ARCHITECTURE

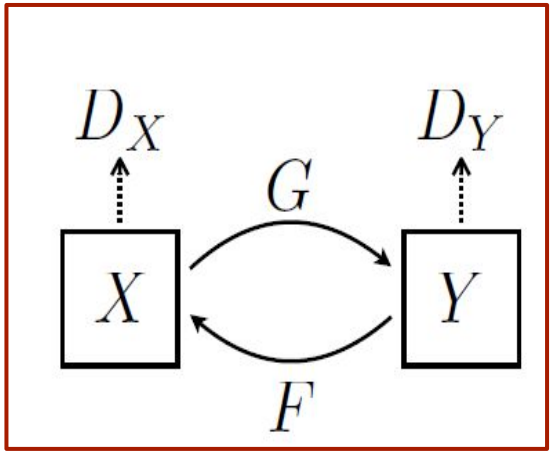
According to the paper:

1. Since it is a cyclic system, there are 2 Generators with same architecture and 2 Discriminators with same discriminator architectures.
2. We are also calculating various types of Losses, one being the adversarial loss on individual Generators and secondly, the cyclic consistency loss.
3. The learning rate is set to **0.0002**
4. We optimize the Discriminator, for only **even** number of EPOCHs to slow down the rate at which the Discriminator learns with respect to the Generator.

# NETWORK ARCHITECTURE

As the fig. Illustrates:

- The Generator G, generates an image y, using images from Domain(X).
- The Generator F, generates an image x, using images from Domain(Y).
- Therefore we require 2 Discriminators. Each D takes two inputs(the image generated by G and the corresponding original image) and the output is the likelihood the D image is from the original data.
- In other words, it calculates the individual adversarial losses and the cyclic consistency loss, whether the image is fake or real.



# GENERATOR ARCHITECTURE

1. This network contains two stride-2 convolutions, several residual blocks, and two fractionally stride convolutions with stride ½.

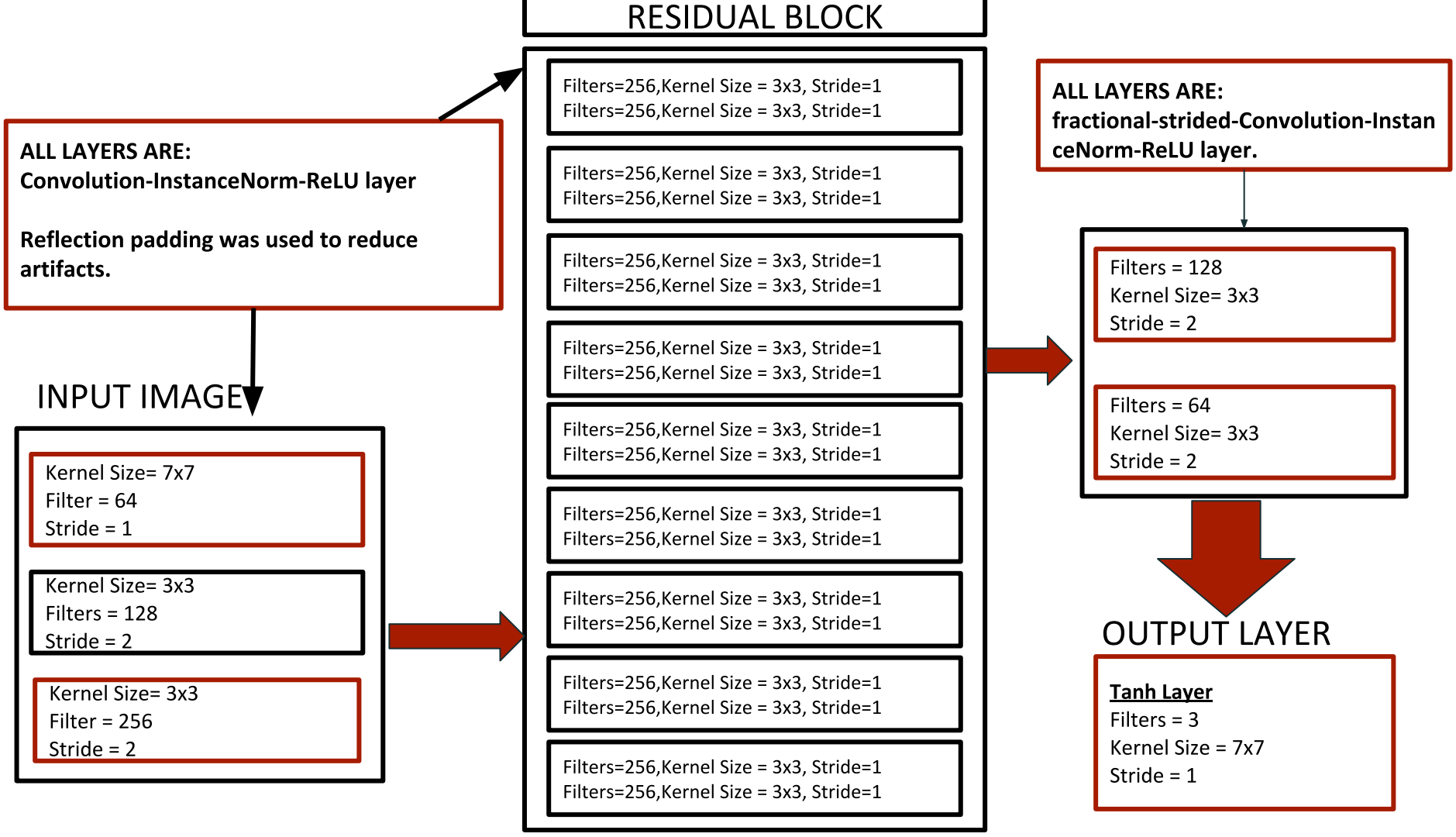
2. We used 9 residual blocks according to the paper for our 256x256 resolution training images.

3. Reflection padding was used to reduce artifacts.

4. All layers contain Instance Normalization and Relu as Activation Layer.

5. We are using Fractional Strided Convolution, which is the Transpose Convolution as we have stride = ½ , as described in the paper.

6. We used tanh for our last layer for the generator just like the original paper did, and this is to ensure that the output has pixels in the range of [0,255].


RESIDUAL BLOCK
ALL LAYERS ARE:
Convolution-InstanceNorm-ReLU layer
Reflection padding was used to reduce artifacts.
INPUT IMAGE
Kernel Size= 7x7
Filter = 64
Stride = 1
Kernel Size= 3x3
Filters = 128
Stride = 2
Kernel Size= 3x3
Filter = 256
Stride = 2
Filters=256,Kernel Size = 3x3, Stride=1
Filters=256,Kernel Size = 3x3, Stride=1
Filters=256,Kernel Size = 3x3, Stride=1
Filters=256,Kernel Size = 3x3, Stride=1
Filters=256,Kernel Size = 3x3, Stride=1
Filters=256,Kernel Size = 3x3, Stride=1
Filters=256,Kernel Size = 3x3, Stride=1
Filters=256,Kernel Size = 3x3, Stride=1
Filters=256,Kernel Size = 3x3, Stride=1
Filters=256,Kernel Size = 3x3, Stride=1
Filters=256,Kernel Size = 3x3, Stride=1
Filters=256,Kernel Size = 3x3, Stride=1
ALL LAYERS ARE:
fractional-strided-Convolution-InstanceNorm-ReLU layer.
Filters = 128
Kernel Size= 3x3
Stride = 2
Filters = 64
Kernel Size= 3x3
Stride = 2
OUTPUT LAYER
Tanh Layer
Filters = 3
Kernel Size = 7x7
Stride = 1

# DISCRIMINATOR ARCHITECTURE

1. Discriminator uses a 70x70 PATCHGANs.
2. The task of these PATCH GANs are to classify whether 70x70 overlapping image patches are real of fake. Such patch level discriminator architecture have fewer parameters as compared to a full-image discriminator, and it also can work on arbitrarily sized images.
3. All Convolutional Layers in Discriminator uses Leaky Relu as Activation Function.
4. All Convolutional Layers use a Kernel = 4x4 and Stride =2.
5. Except the first layer, all layers use Instance Normalization as normalization layers. However, the first layer has a slope of 0.2 according to the paper specification.
6. After the last layer, we pass the output to a convolutional layer to produce a 1D output with 4x4 kernel and stride=1.
7. We also added random noise to the image to improve our training and distinguishing the images with more accuracy.

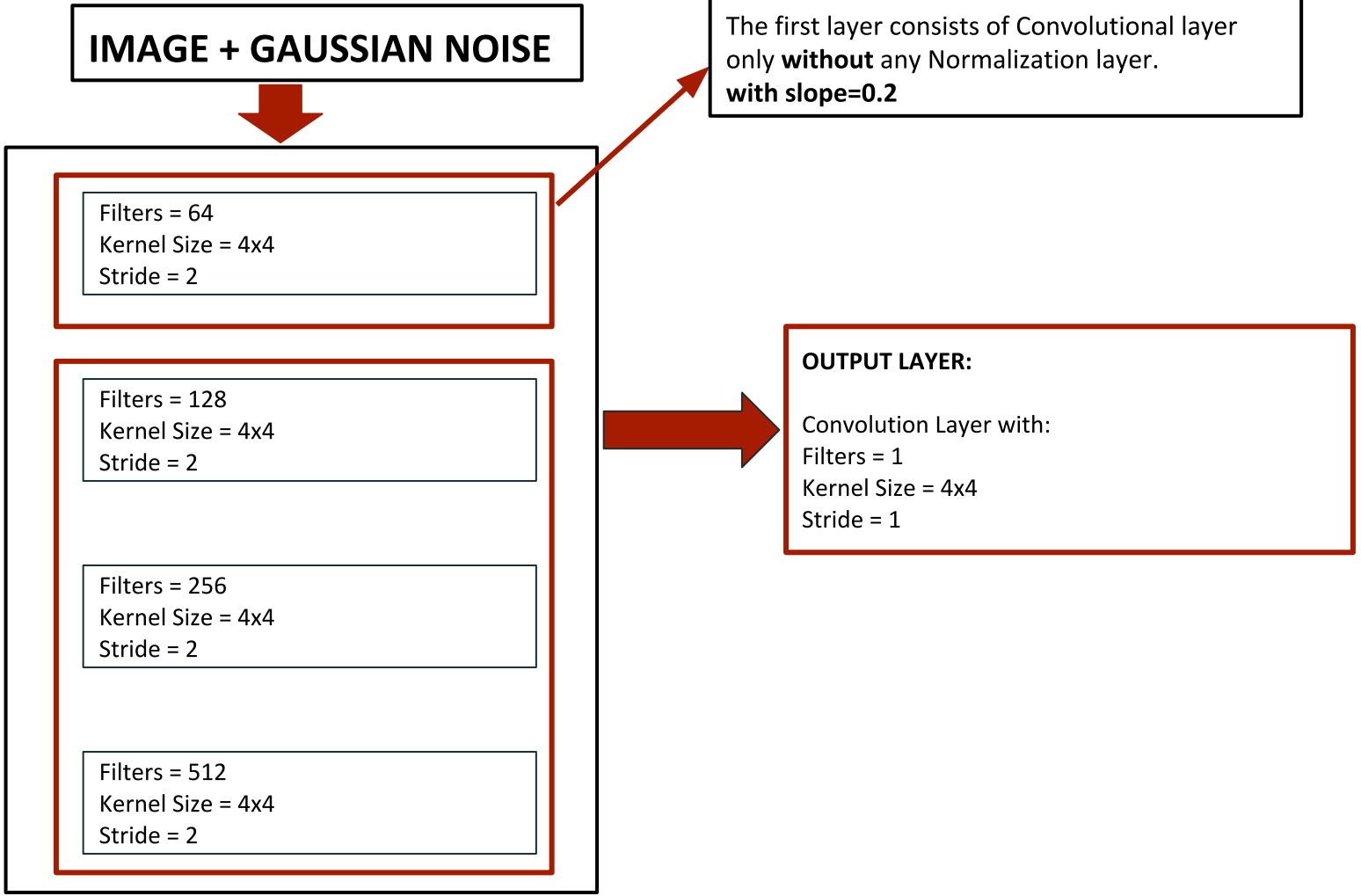
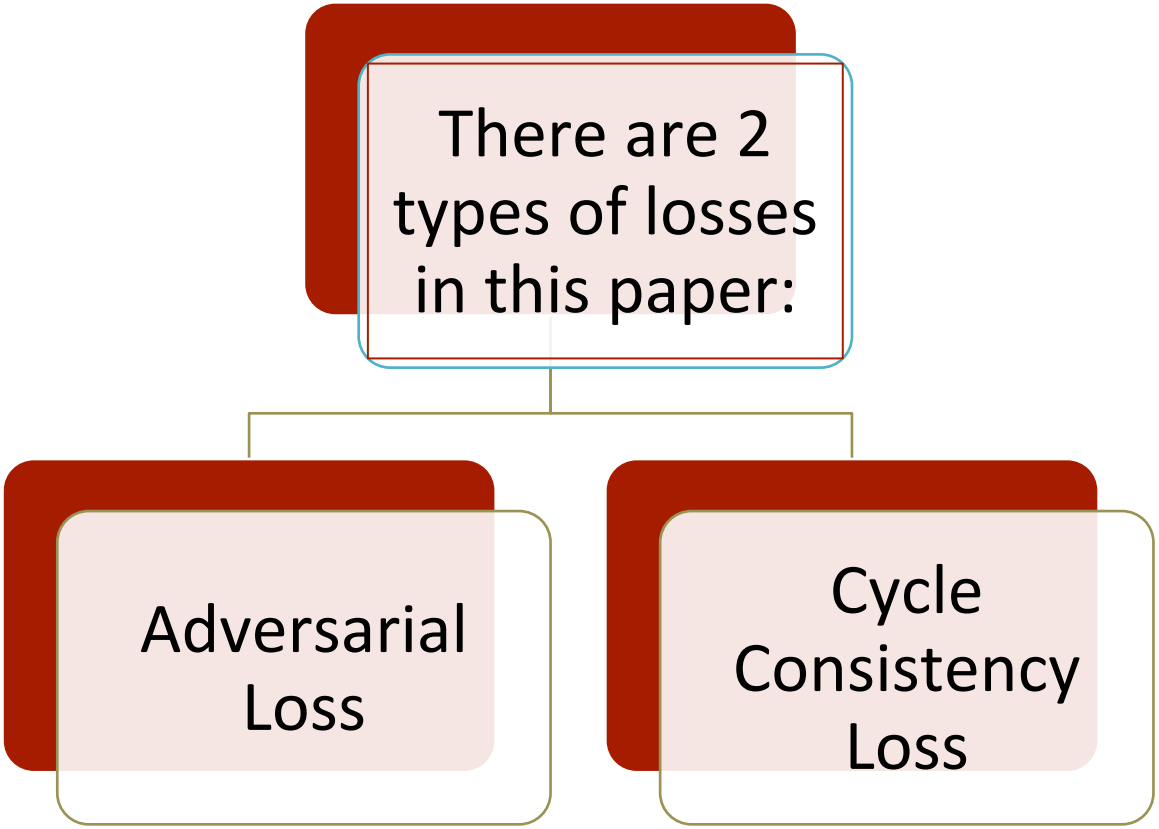
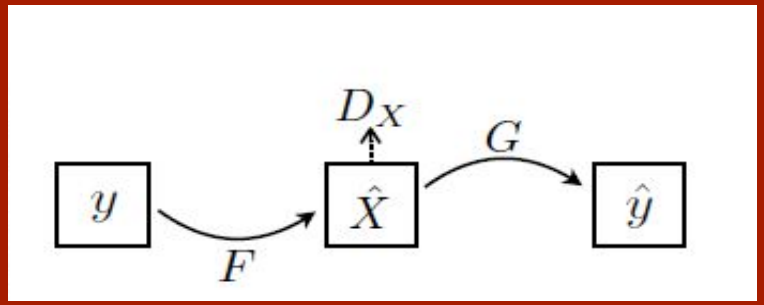


IMAGE + GAUSSIAN NOISE
The first layer consists of Convolutional layer only **without** any Normalization layer.
**with slope=0.2**
Filters = 64
Kernel Size = 4x4
Stride = 2
Filters = 128
Kernel Size = 4x4
Stride = 2
Filters = 256
Kernel Size = 4x4
Stride = 2
Filters = 512
Kernel Size = 4x4
Stride = 2
**OUTPUT LAYER:**
Convolution Layer with:
Filters = 1
Kernel Size = 4x4
Stride = 1

# LOSS


There are 2 types of losses in this paper:
Adversarial Loss
Cycle Consistency Loss

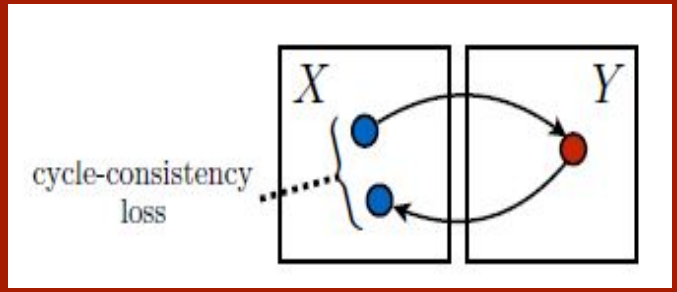
## ADVERSARIAL LOSS




The loss between a Generator and it's Discriminator, is the Adversarial loss of that particular pair.

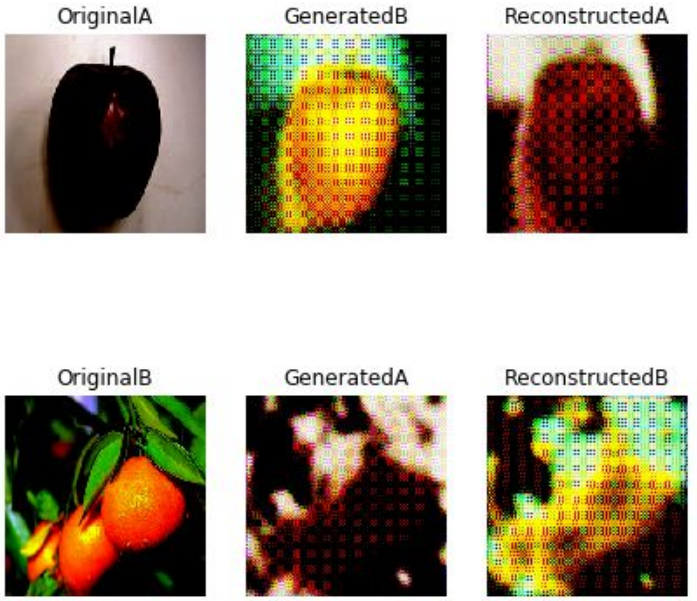
Therefore, there we have 2 Adversarial Loss functions. We use mean square loss for calculating the loss.

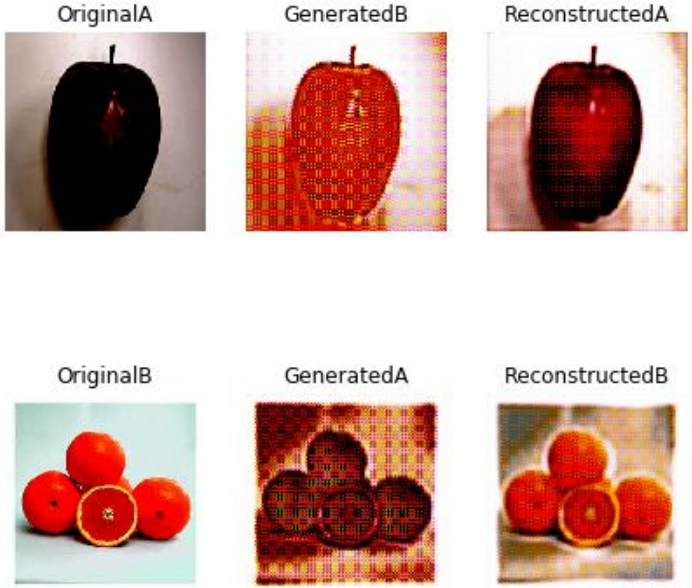
## CYCLE CONSISTENCY LOSS




The adversarial losses alone cannot guarantee that the learned function can map an individual input to a desired output.
Therefore we calculate the total loss occurring after the total cycle. We use absolute difference for calculating this difference.

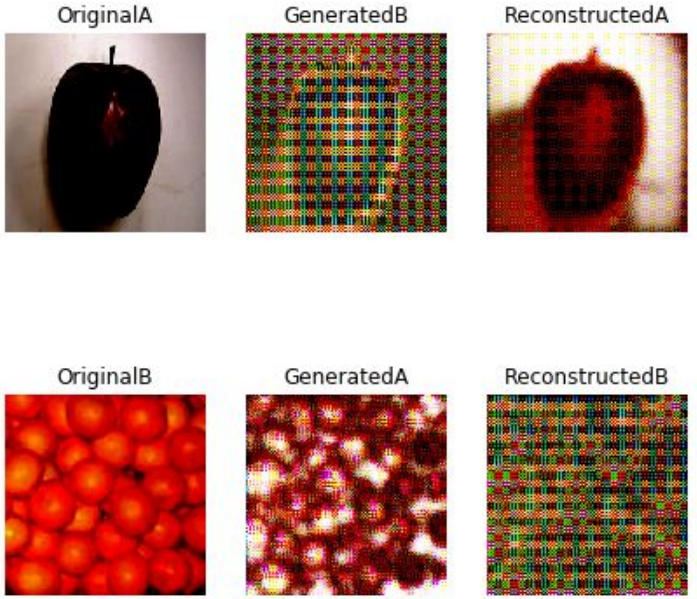
# Ablation study with only **Adversarial** loss

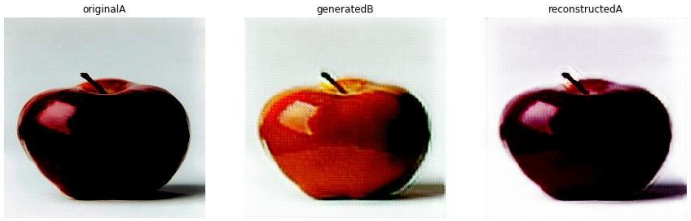





# Ablation study with **cycle** loss

Ablation study with **cycle loss in one direction (forward cycle loss)**





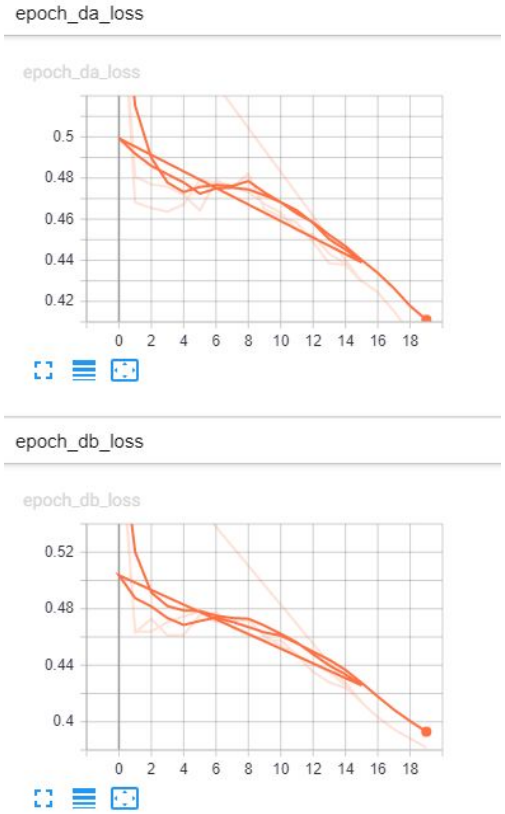

Results without applying
any ablation study shows the loss functions play critical roles in arriving at high-quality results.

# TENSORBOARD CHART

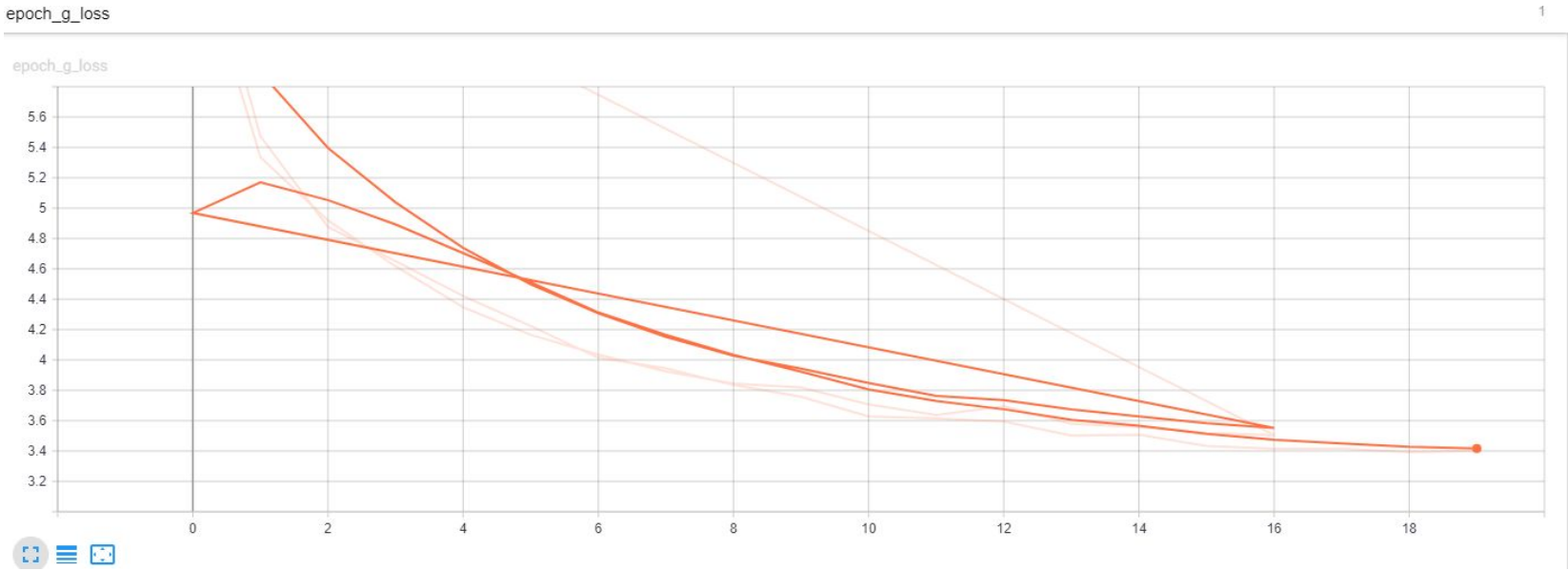
**Discriminator A and B loss.**

These results observed without slowing down the rate of D like the previous slide.

As we can see, there's a huge inconsistency unlike the previous slide.



# GENERATOR LOSS WHEN BOTH THE GENERATOR AND DISCRIMINATORS LEARNING IS SAME.

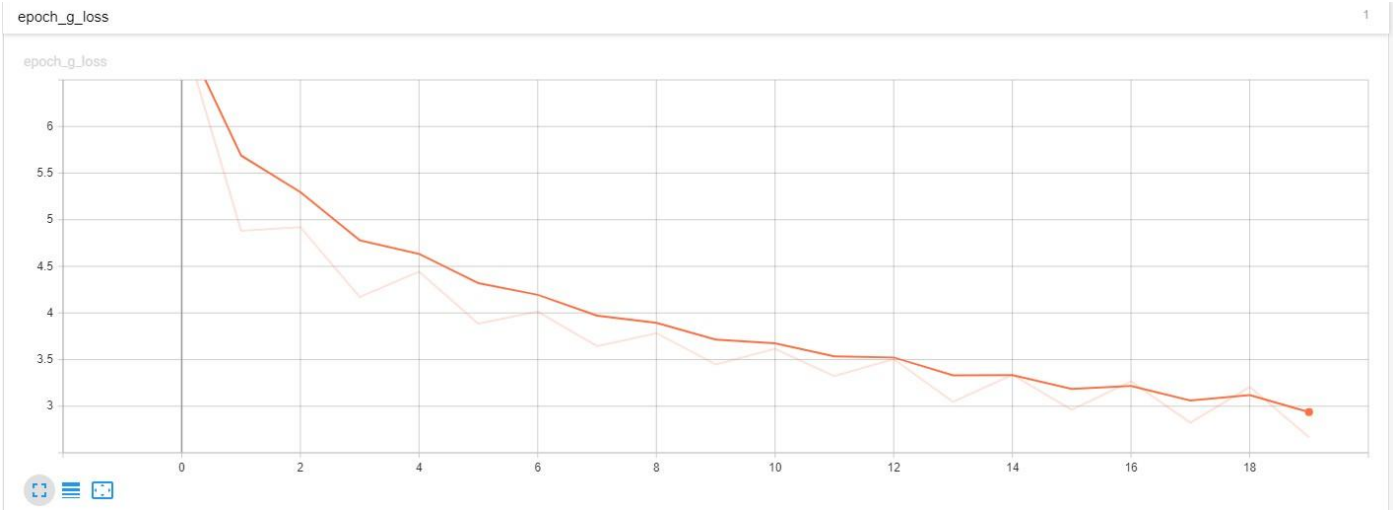


**Discriminator A and B loss.**

These results was observed from slowing down the rate at which the Discriminator learns, relative to the rate of Generator.

The reason behind this shape of the loss is due to the slowing down of the discriminator loss which was done by only setting D to work only by even epochs .

# GENERATOR LOSS WHEN DISCRIMINATORS LEARNING IS SLOWED DOWN.



We trained our model on thispublic datasets:

- Apple to Oranges
- Scenery

Our results are shown in the next couple of slides.

# THANKS FOR LISTENING