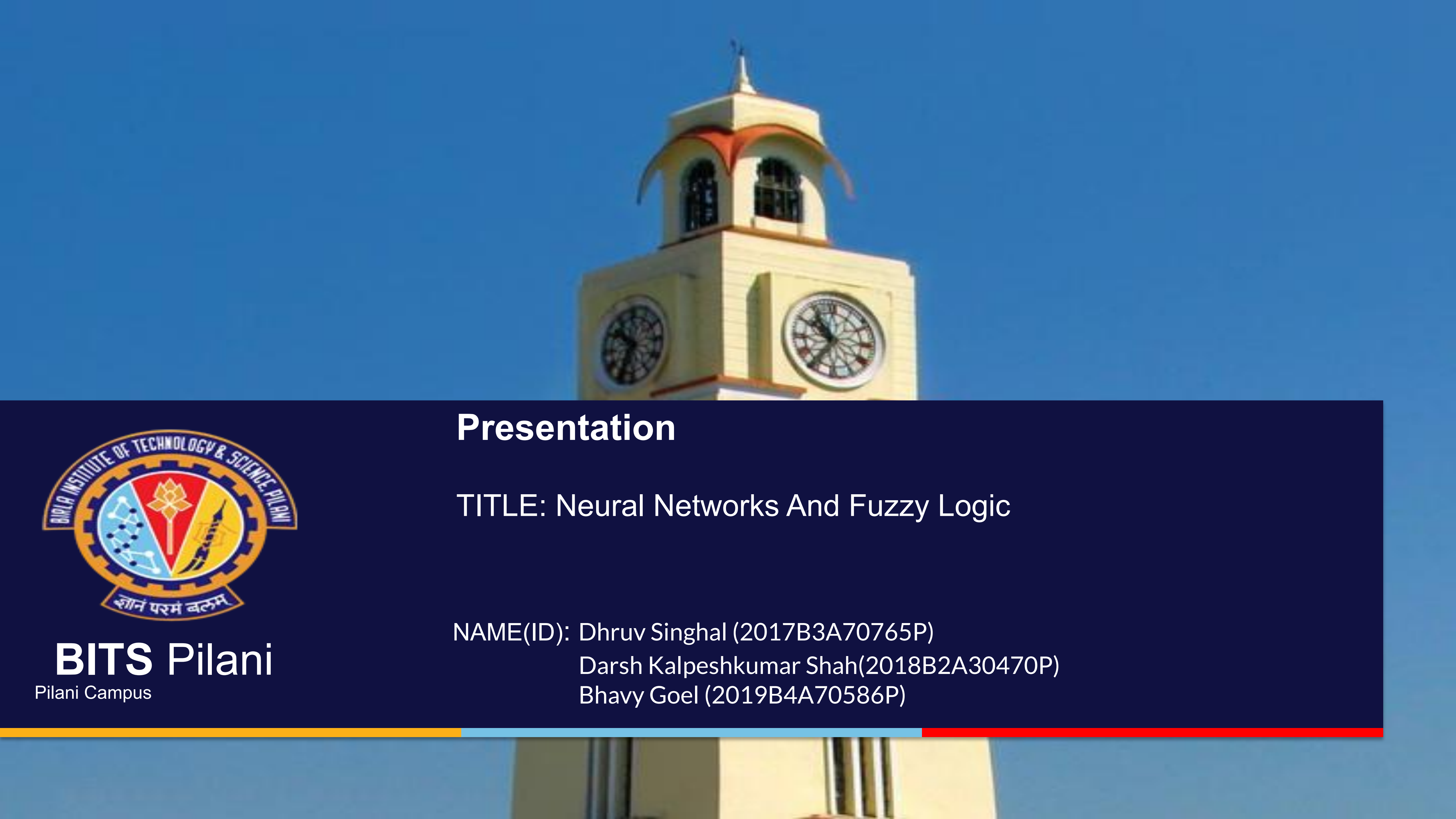


BITS Pilani
Pilani Campus

Presentation

TITLE: Neural Networks And Fuzzy Logic

NAME(ID): Dhruv Singhal (2017B3A70765P)
Darsh Kalpeshkumar Shah(2018B2A30470P)
Bhavy Goel (2019B4A70586P)

Title and Authors

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

Paper Description

AIM:

- CycleGAN is an approach to learning how to translate an image from a source domain X to a target domain Y in the absence of paired examples or supervised learning data

- The aim 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. Since this mapping is highly under-constrained, couple it with an inverse mapping $F : Y \rightarrow X$ and introduce a cycle consistency loss to push $F(G(X)) \approx X$.

Paper Description

- CycleGAN aims to present qualitative results on several tasks where paired training data does not exist, including collection style transfer, object transfiguration, season transfer, photo enhancement, etc.
- The aim of this assignment is to
  - Train the model on the Van Gogh style transfer dataset
  - Plot a graph of training accuracies across epochs
  - Apply the model to images of another painting dataset such as Monet or Cezanne and show results. Save the trained model weights for discriminator and generator.
  - Come up with ideas/implementations/improvements for the model.

Methodology

Let us say that we are given one set of images in domain X and another set in domain Y
We train a mapping,

$$G : X \to Y$$

such that the output,

$$\hat{y} = G(x),\ x \in X,$$

is indistinguishable from images $y \in Y$ by an adversary trained to classify $\hat{y}$ apart from y

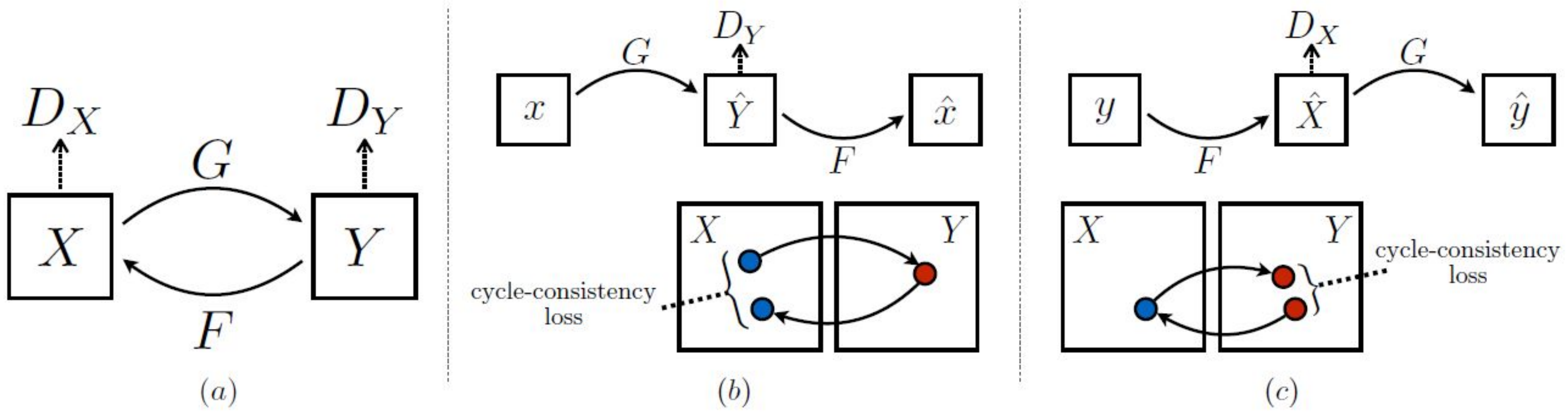
The optimal G thereby translates the domain X to a domain $\hat{Y}$ distributed identically to Y

However, such a translation does not guarantee that an individual input x and output y are paired up in a meaningful way – there are infinitely many permutations of mappings in G that will induce the same distribution over $\hat{y}$.

Methodology

Also, it has been found that sometimes it is difficult to optimize the adversarial objective in isolation: standard procedures often lead to the well known problem of mode collapse, where all input images map to the same output image and the optimization fails to make progress

Therefore, the process aims to exploit the property that translation should be "cycle consistent".



innovate achieve lead

Final Outcomes

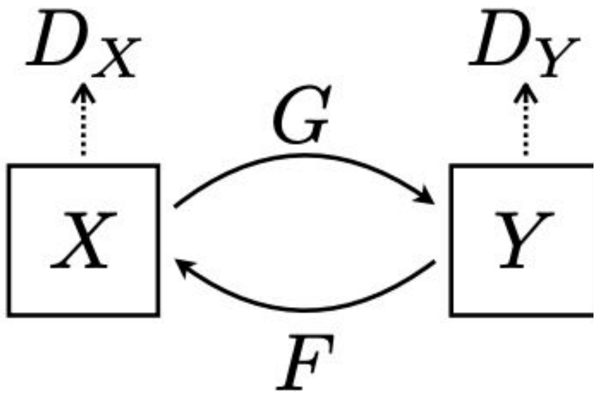
The method can be applied to a wide range of applications, including collection style transfer, object transfiguration, season transfer and photo enhancement.

Upon comparison against previous approaches that rely either on hand-defined factorizations of style and content, or on shared embedding functions, CycleGAN outperforms them.

In this assignment we show that CycleGAN can be trained on the Van Gogh dataset and can be used as a method of style transfer to convert Real Life images and paintings from Monet or Cezanne to the ‘style’ of Van Gogh. We also show the importance of two important aspects that CycleGAN uses, cycle consistency and identity loss, through visual results.

Background Concepts (Formulation):

The goal is to learn mapping functions between two domains X and Y given training samples and as illustrated, the model includes two mappings, thus using 2 Generators $G : X \rightarrow Y$ and $F : Y \rightarrow X$.



Background Concepts (Formulation):

In addition, there exists 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)}.

The objective contains two types of terms:

- adversarial losses for matching the distribution of generated images to the data distribution in the target domain
- cycle consistency losses to prevent the learned mappings G and F from contradicting each other

Background Concepts(Adversarial Losses):

An adversarial loss is applied to both mapping functions.
For the mapping function G : X → Y and its discriminator DY , the objective is expressed as:

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

where G tries to generate images G(x) that look similar to images from domain Y , while D_Y aims to distinguish between translated samples G(x) and real samples y.

Background Concepts(Adversarial Losses):

G aims to minimize this objective against an adversary D_y that tries to maximize it, i.e.,

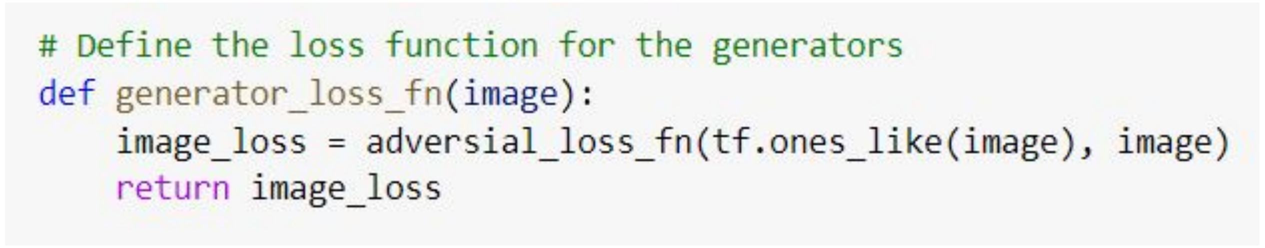
$$\min_G \max_{D_Y} \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y)$$

A similar adversarial loss is introduced for the mapping function F : Y → X and its discriminator D_X as well: i.e.,

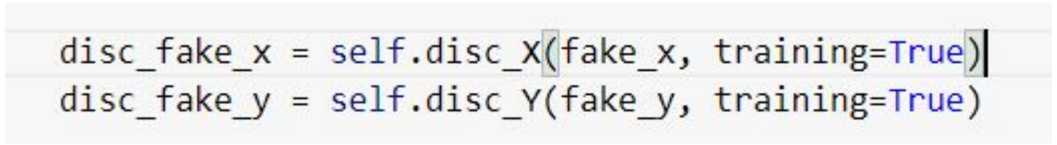
$$\min_F \max_{D_X} \mathcal{L}_{\text{GAN}}(F, D_X, Y, X)$$

Code Snippet

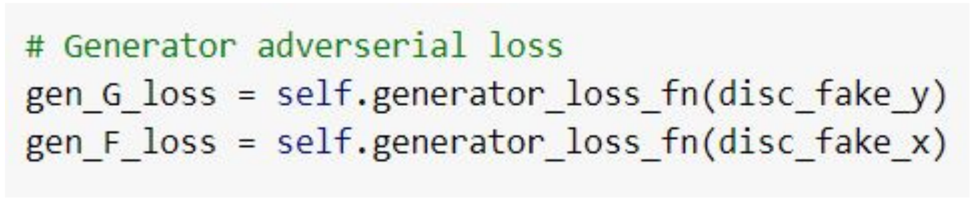
- The L_{GAN} loss is replaced with least square loss rather than negative log-likelihood i.e. for $L_{GAN}(G, D, X, Y)$, we train the G to minimize $E_x{\sim}p_{data}(x)\,[(D(G(x)) - 1)^2]$ and train the D to minimize $E_y{\sim}p_{data}(y)\,[(D(y) - 1)^2] + E_x{\sim}p_{data}(x)\,[D(G(x))^2]$.



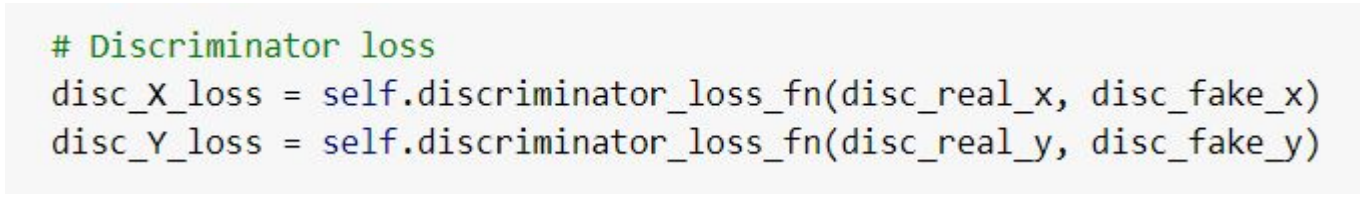
```
# Define the loss function for the generators
def generator_loss_fn(image):
    image_loss = adversial_loss_fn(tf.ones_like(image), image)
    return image_loss
```



```
disc_fake_x = self.disc_X(fake_x, training=True)
disc_fake_y = self.disc_Y(fake_y, training=True)
```



```
# Generator adverserial loss
gen_G_loss = self.generator_loss_fn(disc_fake_y)
gen_F_loss = self.generator_loss_fn(disc_fake_x)
```



```
# Discriminator loss
disc_X_loss = self.discriminator_loss_fn(disc_real_x, disc_fake_x)
disc_Y_loss = self.discriminator_loss_fn(disc_real_y, disc_fake_y)
```

Background Concepts(Cycle Consistency Losses)

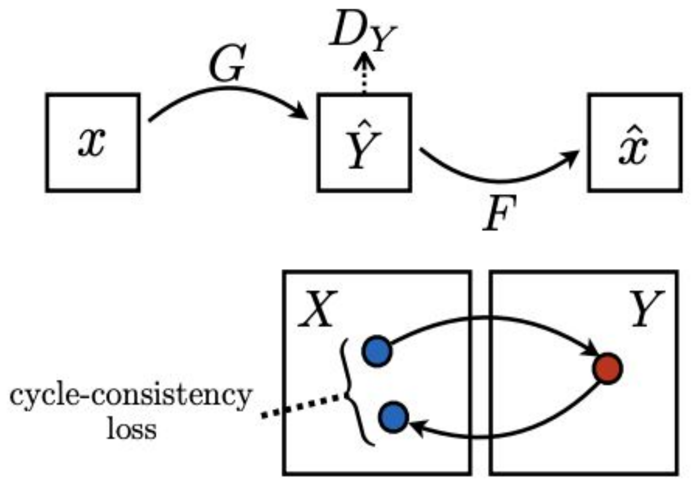
- With a large enough capacity, a network can map the same set of input images to any random permutation of images in the target domain, where any of the learned mappings can induce an output distribution that matches the target distribution.
- Thus, adversarial losses alone cannot guarantee that the learned function can map an individual input x_i to a desired output y_i .

Background Concepts(Cycle Consistency Losses)

- To further reduce the space of possible mapping functions, the learned mapping functions should be cycle-consistent: as shown in the image, for each image x from domain X, the image translation cycle should be able to bring x back to the original image, i.e.,

  $x \to G(x) \to F(G(x)) \approx x.$

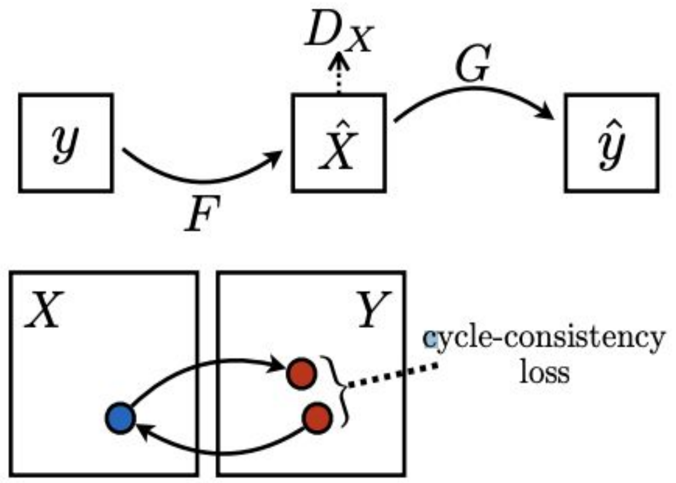
  This is called forward cycle consistency



- Similarly, as illustrated in the following figure, for each image y from domain Y , G and F should also satisfy backward cycle consistency:

  $y \to F(y) \to G(F(y)) \approx y.$

  This is called backward cycle consistency



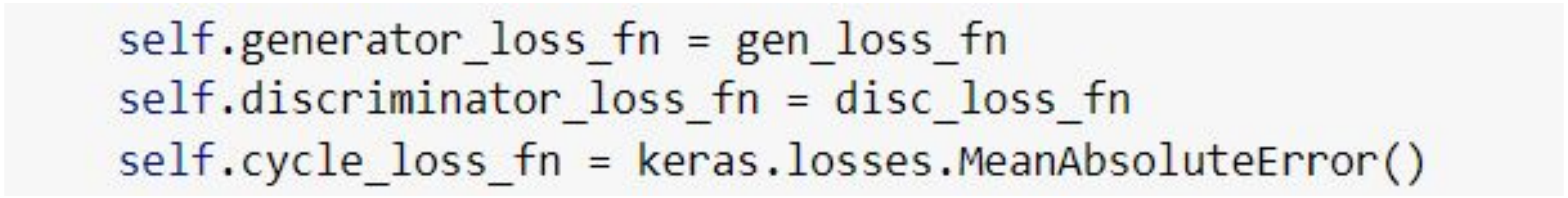
Background Concepts(Cycle Consistency Losses)

- This behaviour is incentivized using a cycle consistency loss:

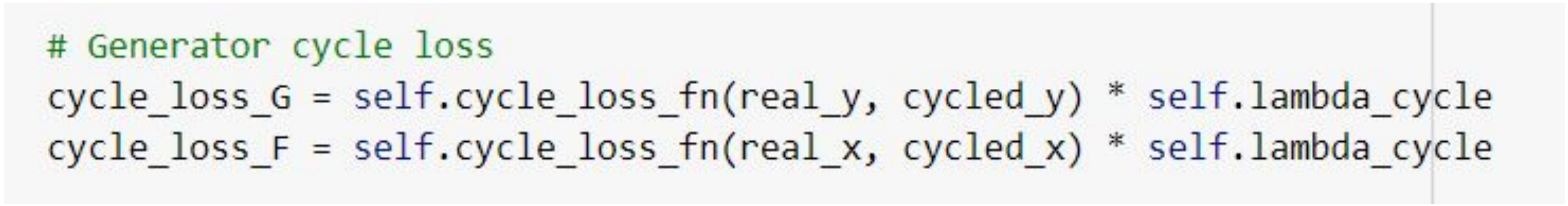
$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G, F) &= \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] \\ &+ \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].\end{aligned}$$

- In preliminary experiments, the authors experimented with 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

Code Snippet



```
    self.generator_loss_fn = gen_loss_fn
    self.discriminator_loss_fn = disc_loss_fn
    self.cycle_loss_fn = keras.losses.MeanAbsoluteError()
```



```
# Generator cycle loss
cycle_loss_G = self.cycle_loss_fn(real_y, cycled_y) * self.lambda_cycle
cycle_loss_F = self.cycle_loss_fn(real_x, cycled_x) * self.lambda_cycle
```

The Full Objective Function

- The full objective is:

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

where λ controls the relative importance of the two objectives.

Thus the aim is to s

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

Deliverables

1. Train the model on the Van Gogh style transfer dataset

2. Plot a graph of training losses across epochs

3. Apply the model to images of another painting dataset such as Monet or Cezanne and show results. Save the trained model weights for discriminator and generator

4. Come up with ideas/implementations/improvements for the model

Details about the dataset

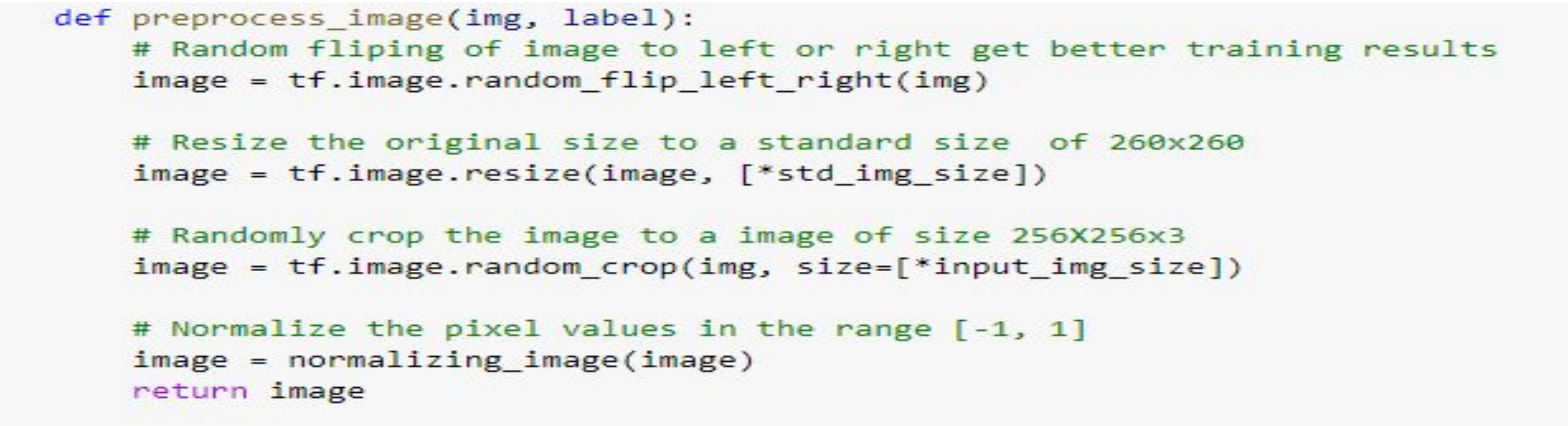
The following details are about the datasets (Van Gogh and Monet) that we use in the assignment

- The vangogh2photo images were downloaded from Tensorflow_Dataset
- The vangogh2photo dataset has the following 4 subfolders -

  testA - 400 Van Gogh test images
  testB - 751 Real test images
  trainA - 400 Van Gogh train images
  trainB - 6287 Real test images
- Training images has been rotated left and right randomly, resized to 260*260 pixels and cropped to 256*256 pixels randomly to increase the randomness of features in dataset used while training and then normalized to [-1,1] values from [0,255]
- The vangogh2photo included only Van Gogh's later works that represent his most recognizable artistic style

Details about the dataset

- The monet2photo dataset has the following subfolder : testA - 121 Monet test images
- Used the monet images in test A sub-dataset to test vangogh style transfer on monet images
- Some other datasets in tensorflow dataset can be trained on thiscyclegan model which include summer2winter, satellite2map, apples2oranges, cityscapes(labels2photo) and horse2zebra



```
def preprocess_image(img, label):
    # Random fliping of image to left or right get better training results
    image = tf.image.random_flip_left_right(img)

    # Resize the original size to a standard size  of 260x260
    image = tf.image.resize(image, [*std_img_size])

    # Randomly crop the image to a image of size 256X256x3
    image = tf.image.random_crop(img, size=[*input_img_size])

    # Normalize the pixel values in the range [-1, 1]
    image = normalizing_image(image)
    return image
```

Implementation Details

- The source code, given by the authors is written purely in Python using PyTorch and we used the code to write our code in tensorflow keras because the author has stated in code repo that tensorflow takes less training time than pytorch
- In particular, we download the dataset, set the appropriate configurations before training, and then train the model from scratch on the Van Gogh Dataset for a total of 19 epochs
- Saved the generator and discriminator model weights in google drive for each epoch
- After this we test our model to give various results on Real Life images as well as images from the Cezanne and Monet paintings datasets
- Objective losses and discriminator losses has been recorded for each epoch to plot loss graph

Implementation Details

- The architectures of D and G uses Instance Normalization instead of batch norm and reflection padding

Generator Architecture

- The generator consists of 9 residual blocks (since image sizes are 256 x 256)
  c7s1-64- d128 -d256 -R256-R256-R256- R256- R256- R256-R256- R256- R256-u128-u64-c7s1-3

Discriminator Architecture

- C64-C128-C256-C512-Convolution for 1d output

- We have used 2 Discriminators and 2 Generators to calculate the cycle loss

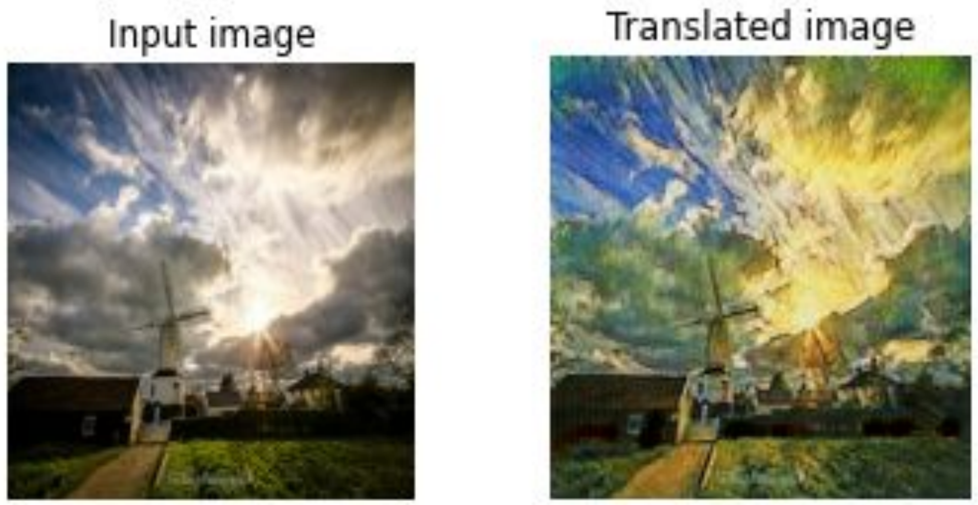
Training Details

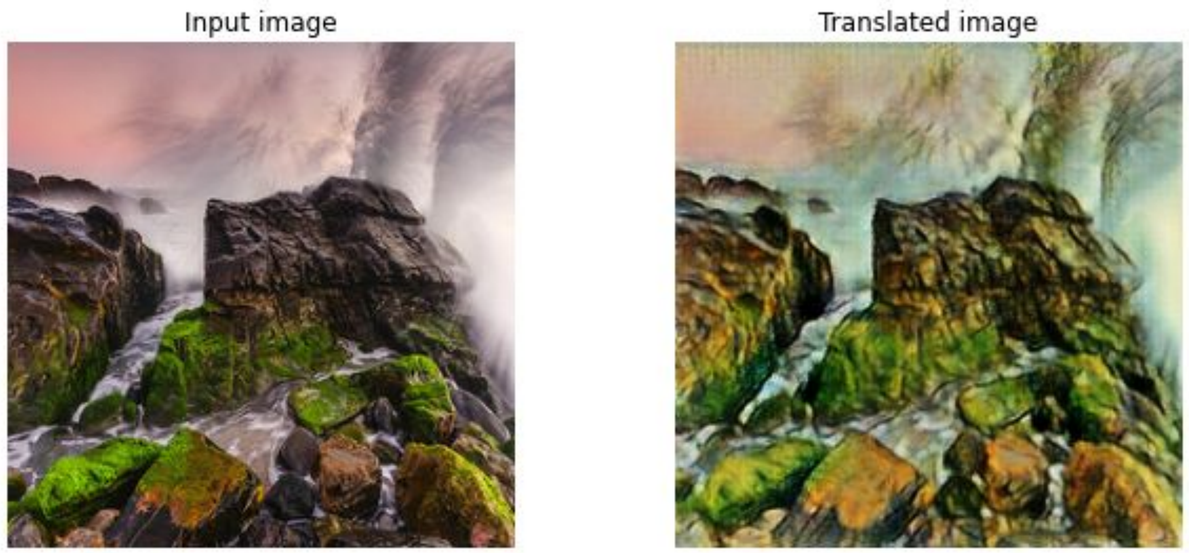
- The model was trained on the Van Gogh dataset, with a subset of the Real Life images (400 images).
- The model was trained for a total of 19 epochs
- The λ in the full objective function is set to 10, batch size is kept at 1, Adam optimizer is used and learning rate is kept at 0.0002 and beta is 0.5.
- As specified by the paper, for the Van Gogh dataset, we introduce another loss component, **identity loss** given by $L_{identity}(G, F) = E_{y}\sim p_{data}(y)\ [||G(y) - y||_1] + E_{x}\sim p_{data}(x)\ [||F(x) - x||_1]$. This is shown to help preserve color composition between the input and output. The importance of this loss can be seen in the results
- This identity loss has a component of 0.05 λ known as identity lambda where lambda is the cycle lambda

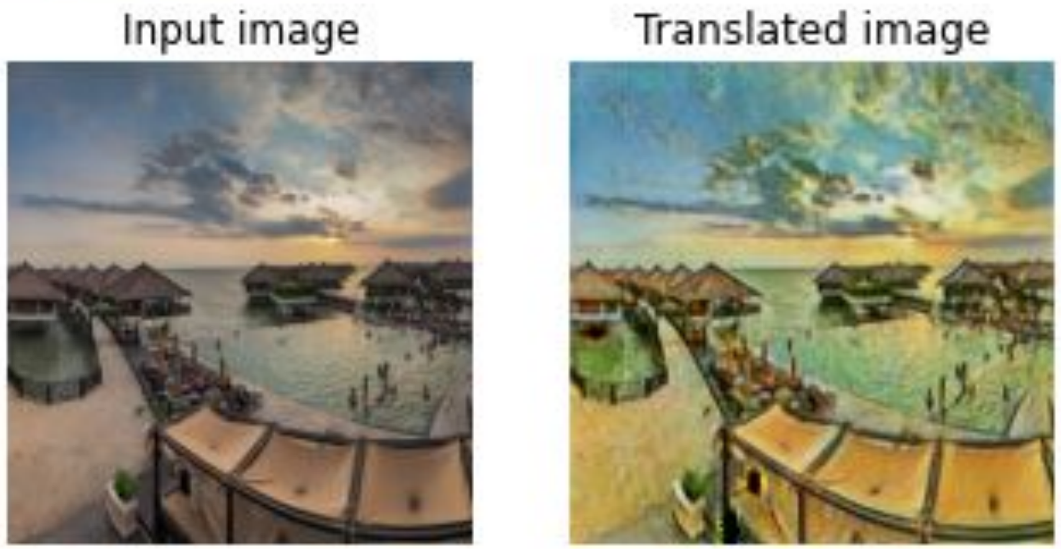
Pseudocode for Implementation

```
1. Create two generators G and F, as well as 2 discriminators D_X and D_Y
2. Initialize the Generator and Discriminator weights
3. For every epoch do:
    a. Collect a samples from Domain X (Van Gogh Paintings) [batchsize was fixed at 1 for our trials]
    b. Collect samples from Domain Y (Real Life Images)
    c. Calculate the losses for generators G and F by passing through them the samples
    d. Calculate the generator loss, discriminator loss,identity loss, and cycle losses
    e. Calculate the full objective loss
    f. Compute the gradients and update weights for the generators
    g. Compute the gradients and update weights for the discriminators.
```

Results (Testing style transfer)



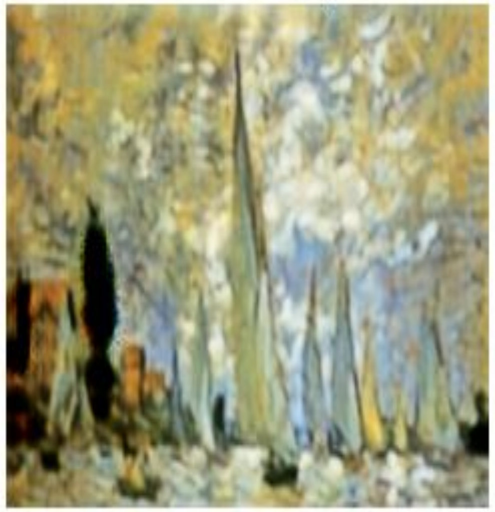




Results (Testing style transfer on Monet)

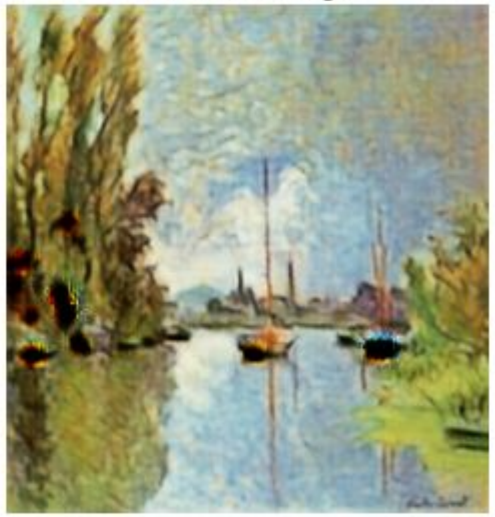
Input image

Translated image



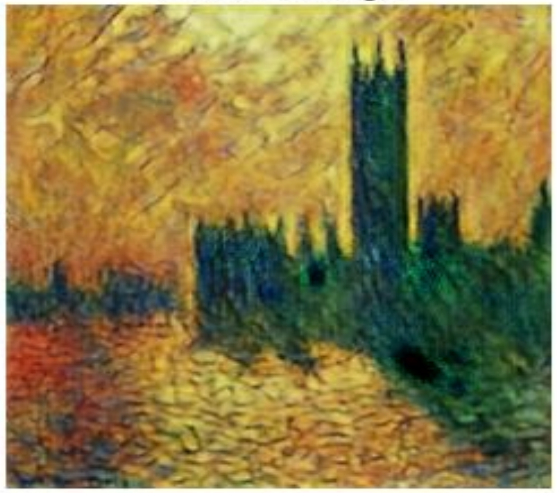
Input image

Translated image

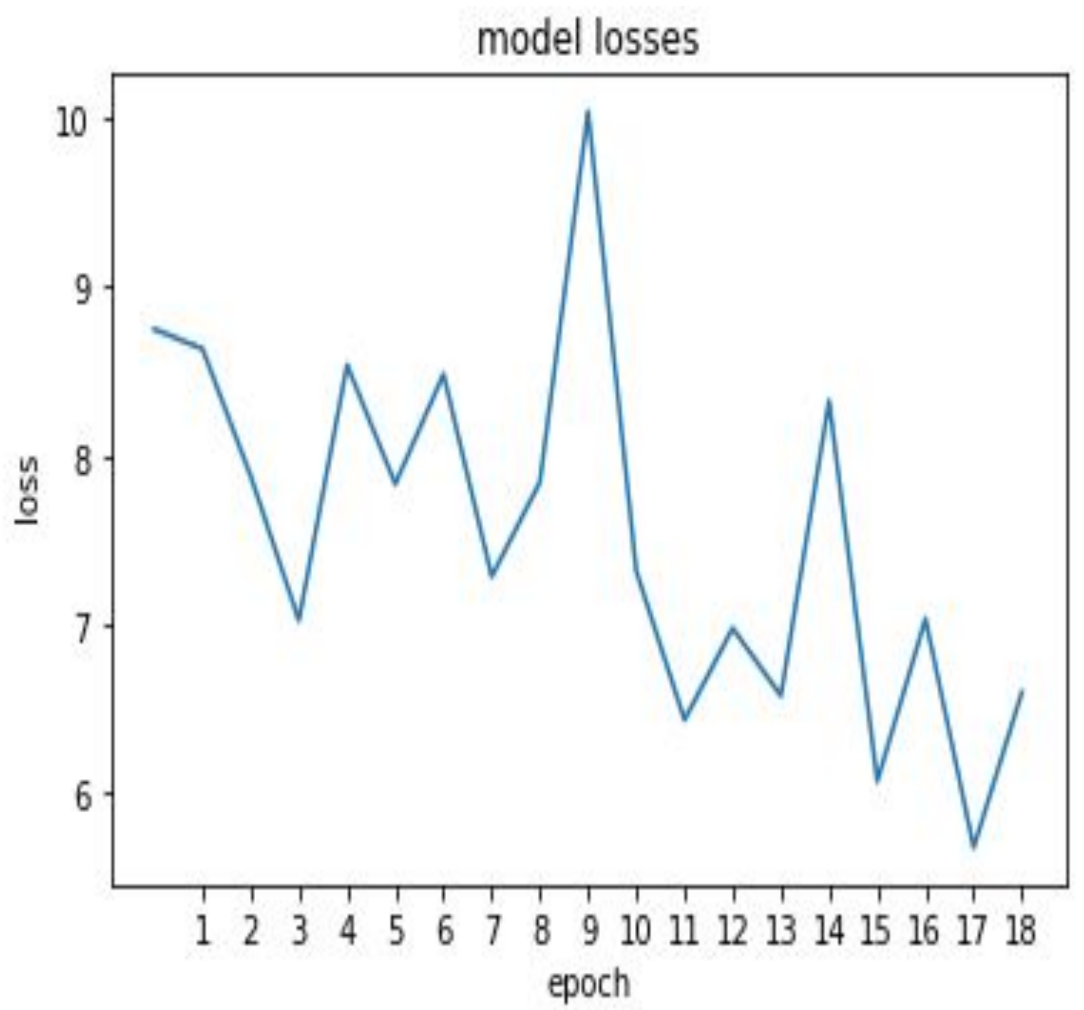


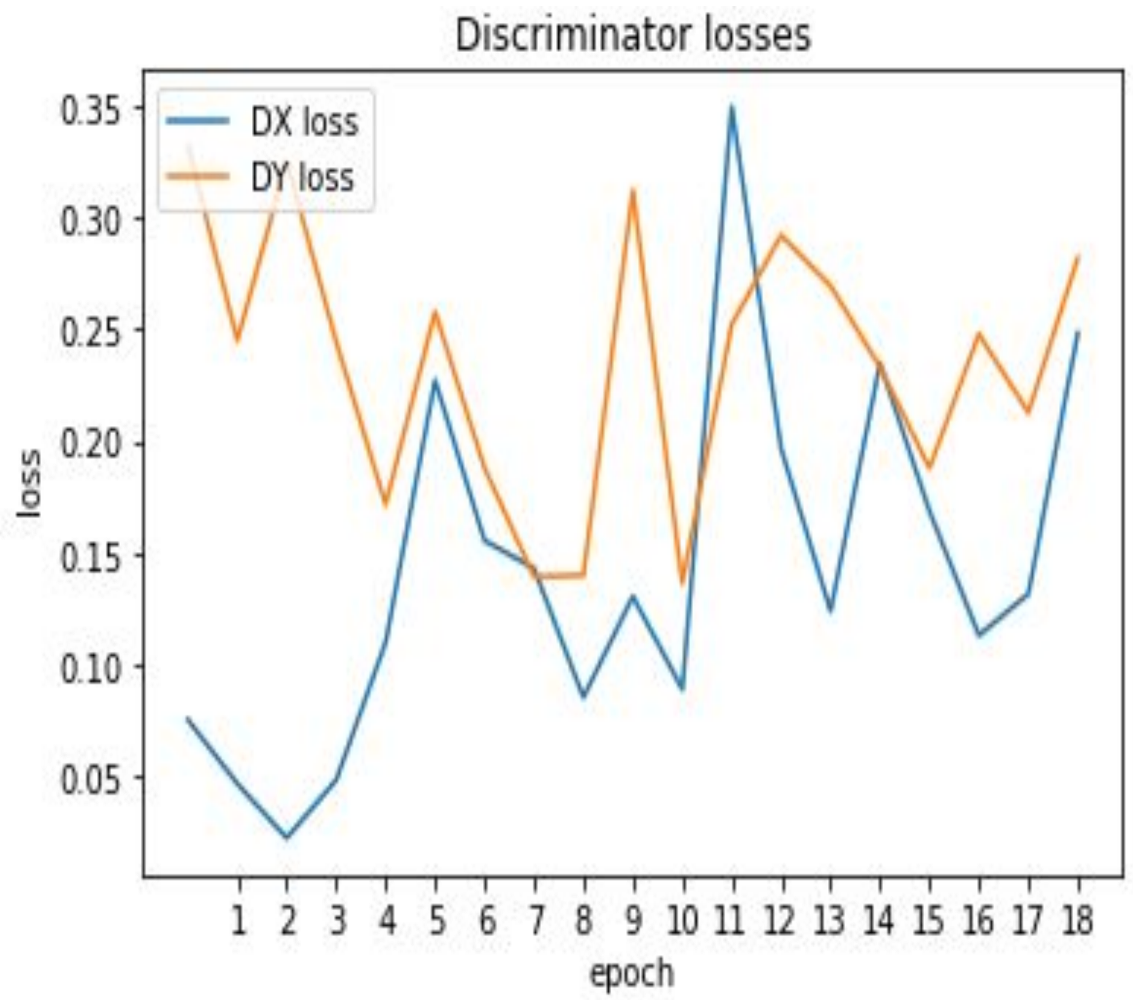
Input image

Translated image



Results (losses)



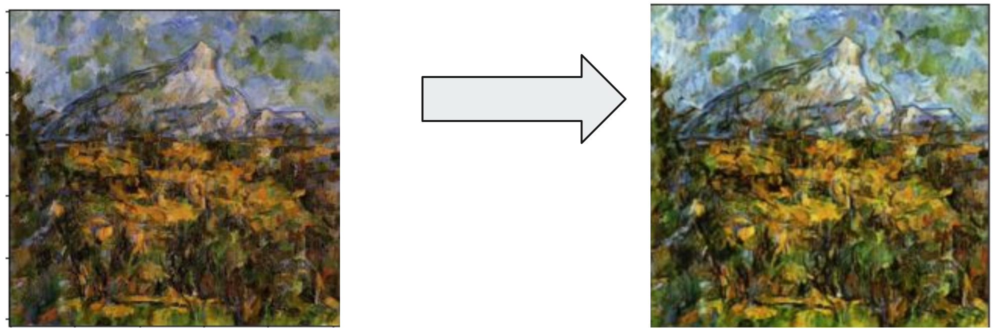


Future Scope and Improvements

- Patch wise training rather than complete image wise training has resulted in better per pixel accuracy and per class accuracy and can be implemented in style transfer using cycle gan
- Image translation using CycleGAN can be used to convert grayscale images to B&W images, change someone's age/gender in images, etc. Image translation has several applications in the fields of art, medical field, videos, and video games.
- We can extend this model to video translation by performing image translation on each frame of a video.

Challenges faced in implementation

- Time was a factor as we were unable to train for more epochs due to epoch training times being large(One epoch took around 20 minutes to train, therefore 200 epochs would take few days)
- We trained our model using Google Colab GPU which keeps on getting disconnected due to inactivity while training and one GPU session allowed us to train only 20 epochs due to session timeout and open source collab session limit
- A few images did not give realistic transformations due to image inconsistencies or similar nature between the input and output domains.

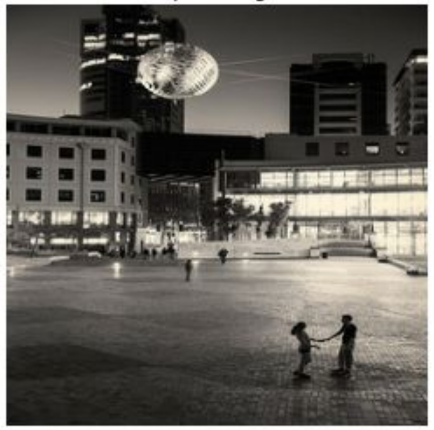


No significant change on this Cezanne painting to 'Van Gogh ' style

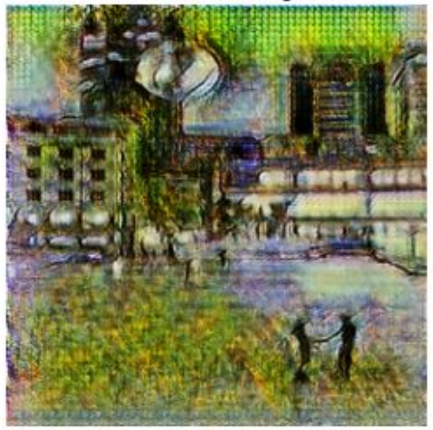
Challenges faced in implementation

- Cycle gan model is not able to give correct results on B/W images due to lack of B/W paintings in the Van Gogh Dataset

Input image



Translated image



- Understanding the complete theory, architecture as well as the relevant loss functions needed for its performance was a challenge that needed to be tackled in order to get the visual results we produced.
- Understanding the existing source code in pytorch and making modification to implement given architecture in tensorflow keras was very time consuming as well.

Experience & Learning Outcomes

- The main learning experience was learning about the breakthrough model of CycleGAN along with its various applications and implementation
- Being able to read and effectively interpret research papers, in order to implement the models and reproduce results.
- We learnt further details about using Python, and particularly the Tensorflow keras framework to create, train and test these models. We also learnt about the vast libraries that Python has to offer
- A great working experience with collab which introduced to us its pros and cons
- The experience of going through online repositories of machine learning and deep learning models with various real-life applications, such as style transfer

innovate achieve lead

Thank You