Indian Institute of Technology, Kharagpur
Deep Learning
CS60010

Final Project Report
Prof. Sudeshna Sarkar

Submitted by: Group-21

Arpan Singh (22BM6JP10)

Saransh Sharma (20CS30065)

Zara Rahman (22BM6JP60)

Vemuri Harsha (20CS10071)

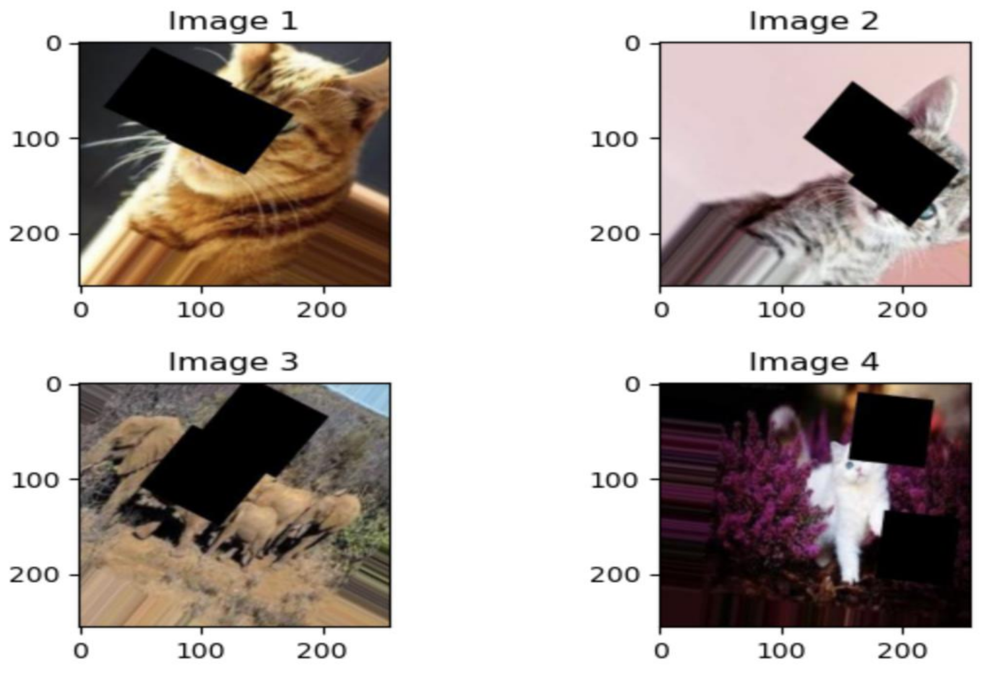
Model Link:
https://drive.google.com/file/d/15yDnlgzrcDiBrHj1urNIZvw9M_t23XUi/view?usp=sharing

Resources Link:
https://drive.google.com/drive/folders/194-ZaWShr5kjjRCom7B4m4_FzZiFBiwa?usp=sharing

1. Dataset Description:
    a. Training Dataset: Image dataset with **1750** masked and unmasked sections is provided for each of the 4 classes of animals: Dog, Cat , Elephant, Tiger. Each image is of size 256*256.
    b. Masked input images have two square patches of size 75*75 each.
    c. Mask section's information is provided in the masked_info.csv file.
    d. Test Dataset: 200 image's dataset with masked sections is provided along with patch information in masked_info.csv

2. Data pre-processing:
    a. Data pre-processing: Every image of the dataset goes through a set of transformations which include resizing the image to 286 X 286 and then randomly cropping a patch from image to make it back to 256 X 256 and then normalizing. Normalization helps in faster convergence of the model and reduces the impact of illumination variations on the image data
    b. Data augmentation: to make the model more robust, the 7000 training dataset was augment with 4 transformations applied to every image making the total size of the dataset to 35000. Every image in the datatset was once flipped, rotated 90 degree, 180 degree and 270 degree to make the model learn complex relations in images. Data processing techniques such as data augmentation can help create additional training data by applying transformations to existing data.

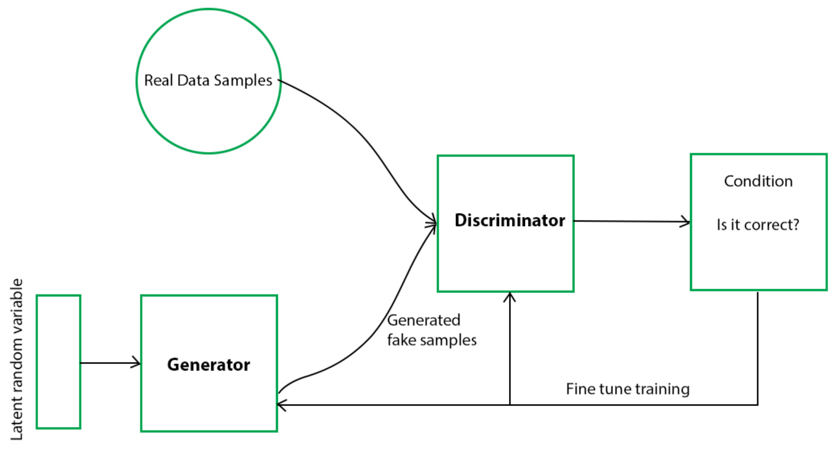
Examples of Augmented images:



3. Models implemented:

a. **Vanilla GAN architecture:**

The generator takes random noise as input and generates a sample that is intended to resemble the real data. The discriminator takes both real data and the generated data as input and tries to distinguish between them.
The generator and the discriminator are trained in an adversarial manner, meaning that the generator tries to generate data that can fool the discriminator, while the discriminator tries to accurately classify whether the data is real or generated.



In this model, the generator takes an input tensor of shape (batch_size, 256, 256, 3) and consists of a series of downsample and upsample blocks with skip connections between them. The final layer is a Conv2DTranspose layer that produces an output image of the same shape as the input.

The discriminator takes in two input images, input_image and target_image, both of shape (batch_size, 256, 256, 3). It has four downsampling blocks followed by two convolutional layers, which output a tensor of shape (batch_size, 30, 30, 1), representing the probability that the input images are real.

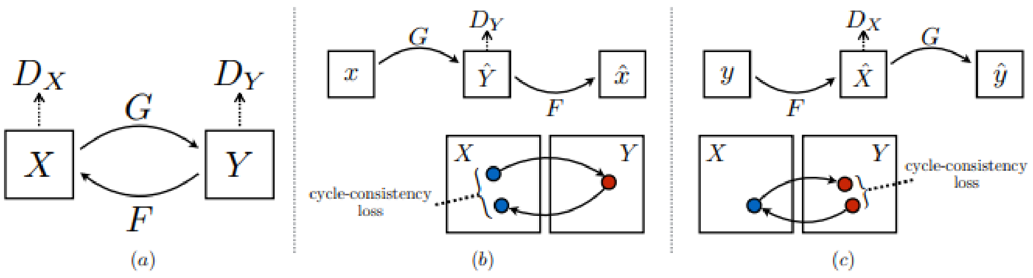
The best score that we could get on the leaderboard using this model was around **0.24881**

b) Stable diffusion pre trained model
The Stable Diffusion Model is a probabilistic generative model that is used for density estimation and data generation tasks. It is based on the idea of iteratively diffusing a set of particles through a latent space, where the density of the particles represents the density of the data.

We tried using pre-trained stable diffusion model from hugging face and fine tuning with existing dataset: https://huggingface.co/stabilityai/stable-diffusion-2-inpainting

We managed to get a score of around **0.23722** on the leaderboard using this model.

c) CycleGAN pretrained model

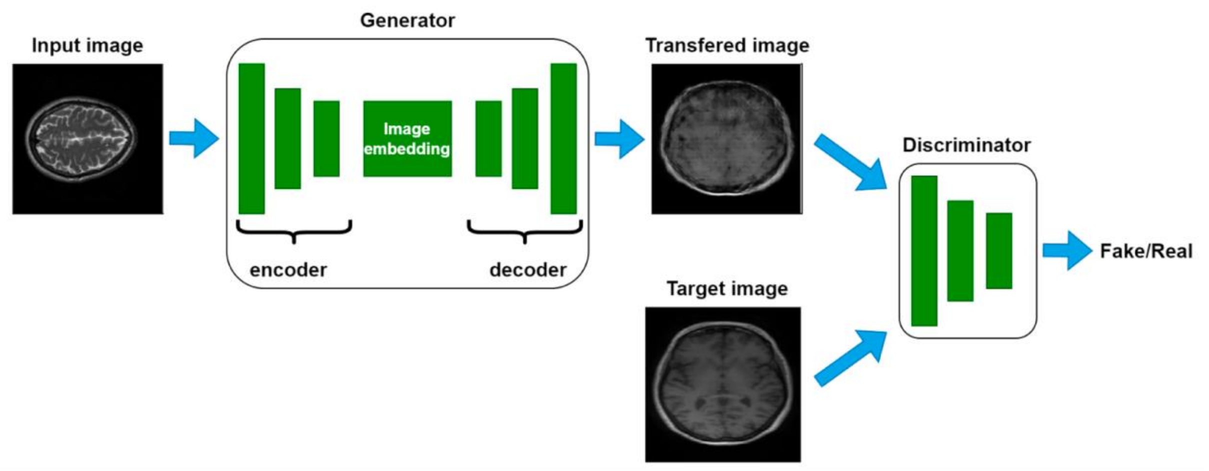


We tried two pretrained cycleGAN models:
1) https://huggingface.co/Rietta/CycleGAN_DL
2) https://huggingface.co/keras-io/CycleGAN

Fine tuning was done for both models using ‘AutoModelForCycleGAN’ in the transformers package from hugging face.

The best score we managed to get using this architecture was **0.23562**

d) Pix2Pix Model (Current Model)



Motivation to use this model: the main difference between standard GAN and Pix2Pix is that Pix2Pix is a conditional GAN that learns to generate output images conditioned on input images, while a standard GAN learns to generate output samples from random noise.During training, the model learns to map input images to output images by minimizing the difference between the generated output image and the ground truth output image.

Architecture of this model:

- The generator network takes in the input image and produces an output image. It consists of an encoder-decoder architecture with skip connections. The encoder network down-samples the input image and

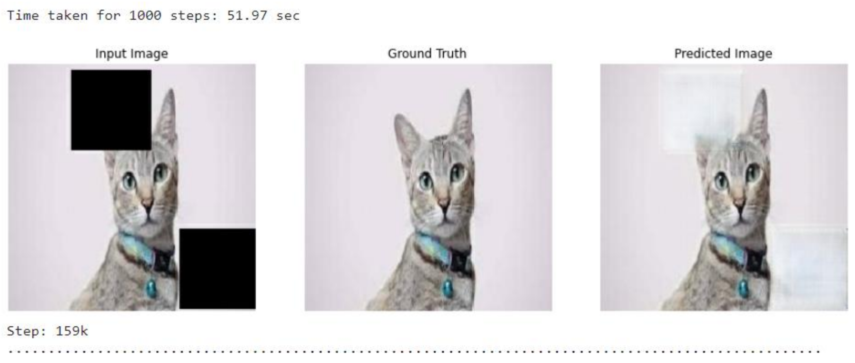
  extracts features, while the decoder network up-samples the features to produce the output image.
- The discriminator network takes in a pair of images, one real and one fake, and tries to distinguish between them. It uses a PatchGAN architecture that classifies image patches as real or fake.
- During training, The generator tries to produce output images that fool the discriminator, while the discriminator tries to correctly classify real and fake images.
- The loss function used in the Pix2Pix model is a combination of adversarial loss (BinaryCrossentropy loss) and pixel-wise loss. The adversarial loss encourages the generator to produce realistic-looking output images, while the pixel-wise loss ensures that the output image is similar to the ground truth output image.
- In addition to the loss function, the Pix2Pix model also uses L1 regularization to encourage smoothness in the output image.

Total params: 54,425,859
Trainable params: 54,414,979
Non-trainable params: 10,880

Hyperparameter tuning i.e., increasing/decreasing the number of epochs, data augmentation, learning rate manipulation, changing the relative weighage of l1 loss in generator helped in improving rmse score from 0.20673 to 0.18742 to 0.17339 to **0.16714 (best score)**.
Hyperparameters:

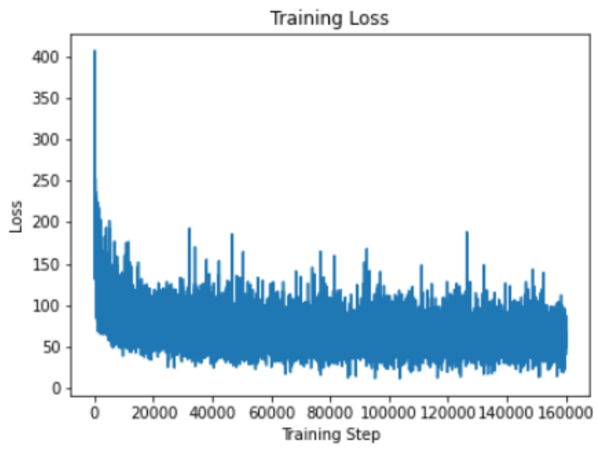
BUFFER_SIZE = 1000	#For shuffling of dataset
BATCH_SIZE = 1	#For breaking dataset into batches
IMG_WIDTH = 256	#Input/output image dimension
IMG_HEIGHT = 256	
Steps = 160000	#Number of epochs
LAMBDA = 1000	#Relative weightage of L1 loss in loss
Learning rate = 3e-4	#Adam optimizer hyperparameter
Beta_1 = 0.5	#Adam optimizer hyperparameter

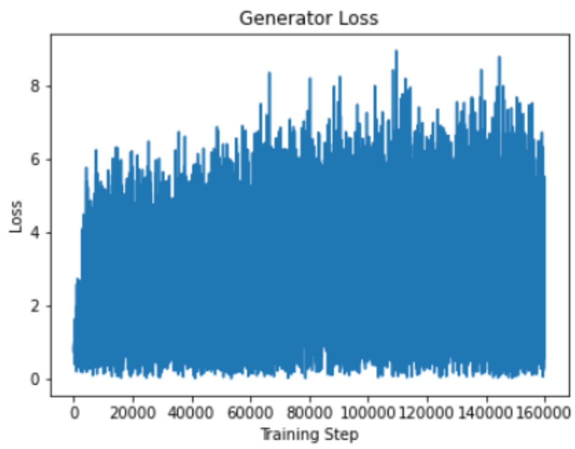
Prediction on training set:

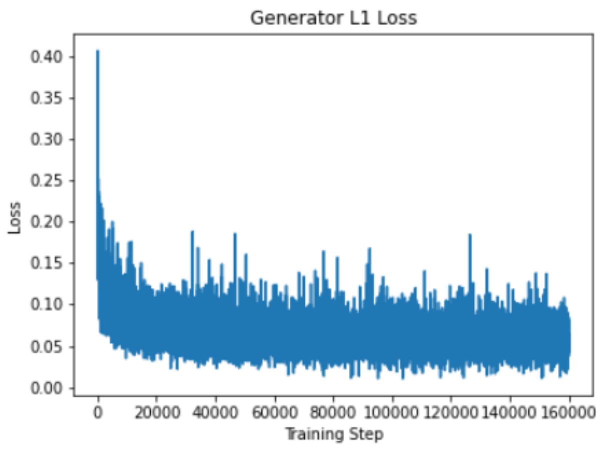


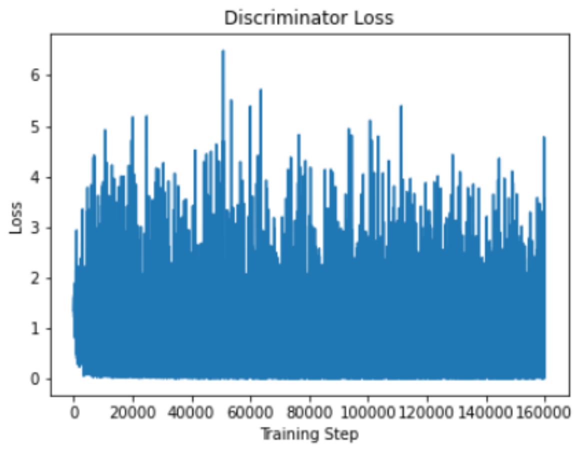
```
Time taken for 1000 steps: 51.97 sec

Step: 159k
....................................................................................................
```

Loss Plots:









Test Set predictions