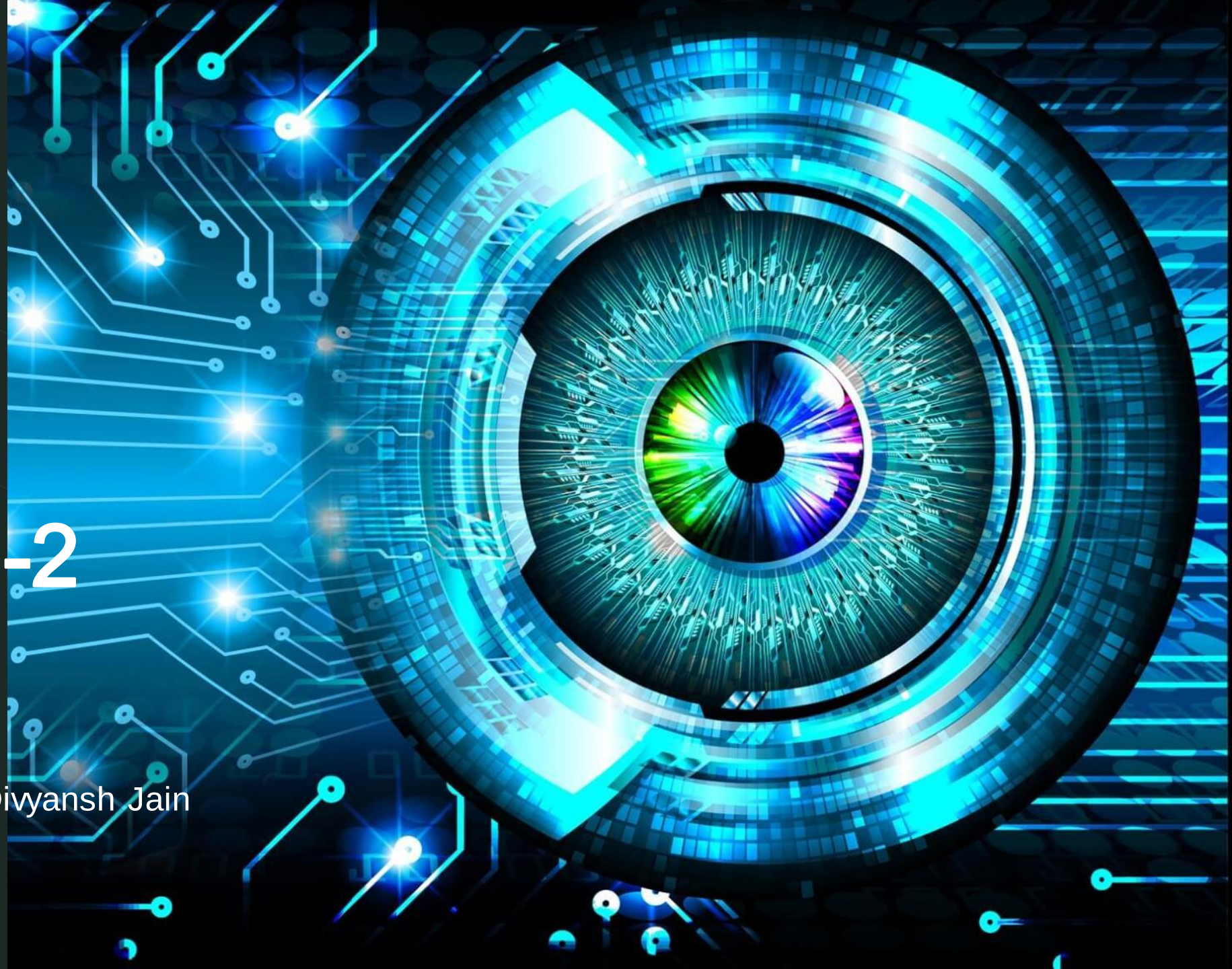
Mosaic PS-2



By Team DMD

Dhruv Sharma | Mitra P. | Divyansh Jain

What We Know and Have:

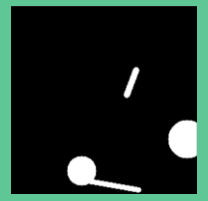
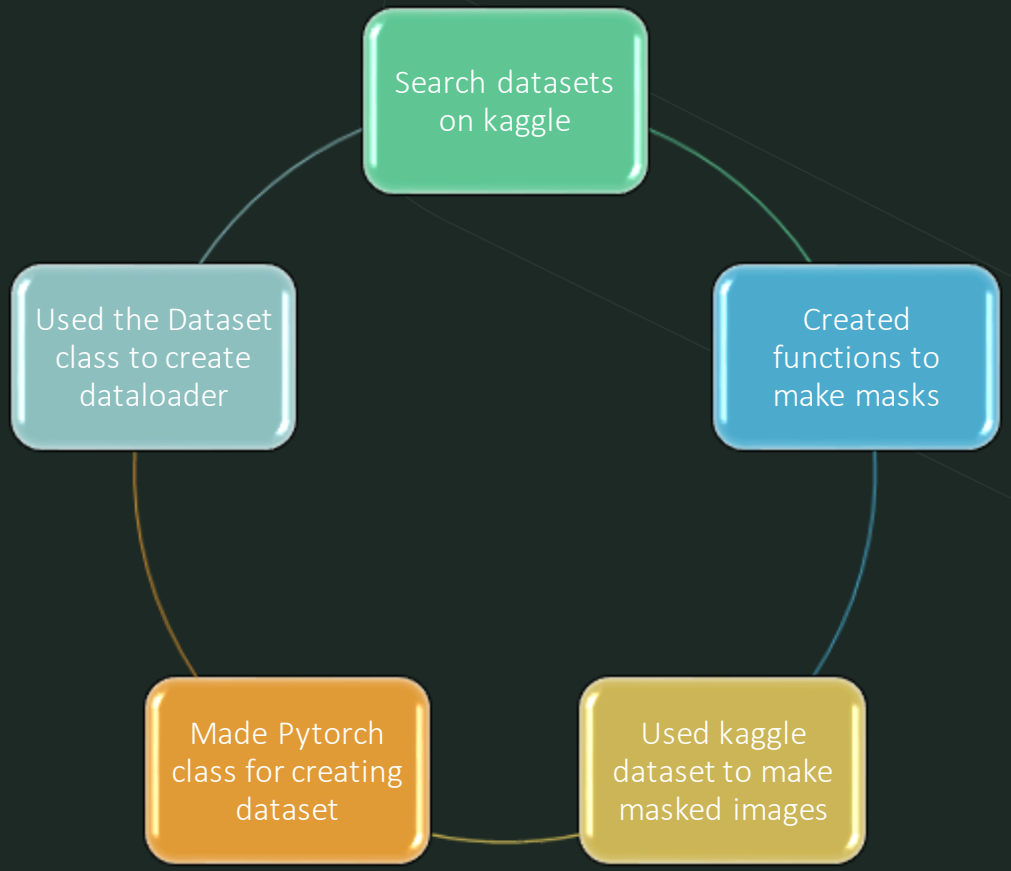


Image inpainting invloves artificially generating masked parts of the image, of which we have no information beforehand, from the features learned by the model.

Such a task is achieved by making generative CNN models, which are able to produce synthetic images from the features it has learned during training, to fill the masked parts

So we have constructed a Generator model which involves consecutive downsampling and upsampling (similar in concept to an autoencoder), and a discriminative model, which challenges the generator by classifying the image as real or fake.

Preparing the dataset



LIBRARIES USED

OpenCV
- To create and apply masks on images

OS
- To navigate through the file system for loading images downloaded from kaggle

PyTorch
- Main framework used to create the GAN model

Matplotlib
- To create plots of generator and discriminator model losses after training is done

Numpy
- To handle matrices, perform mathematical functions on image arrays

Dataset available on kaggle, downloaded on local computer

Created random masks using OpenCV library

Applied on downloaded images to create final dataset

Followed lots of blogs on Medium and research papers for Image Inpainting

Made GAN model from all the ideas gathered, tried a few and finalised the one which gave best results

Trained on dataset created using Reconstruction Loss (custom made) and MSELoss() as adversarial loss

Trained for almost 30 epochs which took more than 4 hours

Model description

- The model is a Generative Adversarial Network(GAN) implemented using PyTorch.
- It takes two inputs: a 256x256x3 RGB image and a same sized mask(binary).
- The architecture consists of 2 networks, Generator and Discriminator which compete with each other to give the best possible results.
- The generator is given the masked image and it tries using the reconstruction loss to recreate the masked areas, with complete image given as label.
- The discriminator is given the output of generator to classify as real or fake, and its adversarial loss is calculated as sum of loss on its given input and the full image.

Loss functions used

Pixel-wise Loss : Reconstruction Loss

For generator

$$\mathcal{L}_{rec}(x) = \|\hat{M} \odot (x - F((1 - \hat{M}) \odot x))\|_2^2,$$

Mathematical Representation

```python
# loss function created to rebuild masked area
class ReconstructLossL2(nn.Module):
    def __init__(self):
        super(ReconstructLossL2, self).__init__()

    def forward(self, mask, y_pred, ipt):
        diff = ipt - y_pred
        loss = torch.mul(mask, diff)
        l1loss = torch.sum(torch.abs(loss))

        return l1loss
```

Code implementation with

square replaced as absolute

to use as an L1 Metric to minimise

Adversarial Loss For discriminator


```python
adversarial_loss = nn.MSELoss()
```

Model Architecture Brief

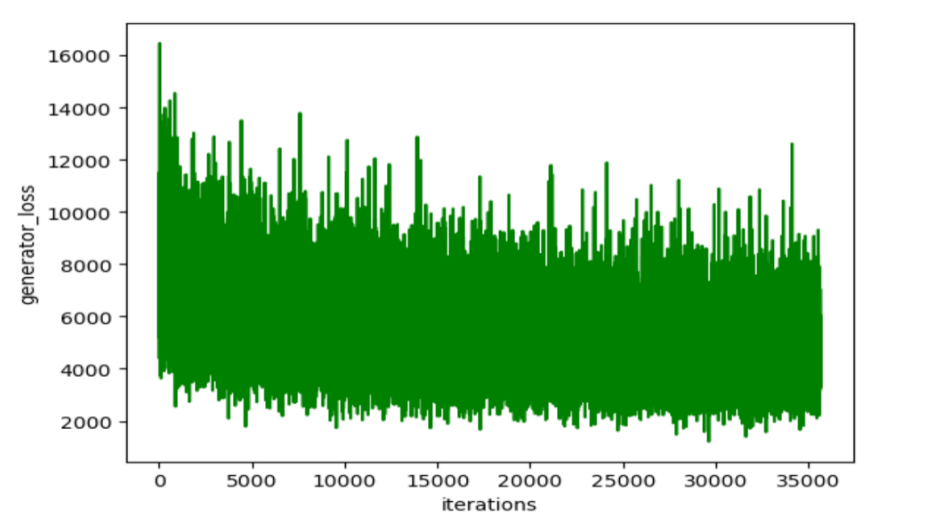
Generator

First part: Used for upsampling, consists of sequence of Conv2D layers with LeakyReLU activation, which is the the best choice here, as it prevents the vanishing gradient problem, which is pretty common in such Deep Models.
Second Part: Used for downsampling, consists of a sequence of Transposed Convolutional (2D) layers along with Batch Normalization (2D) and ELU activation, again with same motive of preventing vanishing gradient.

Discriminator

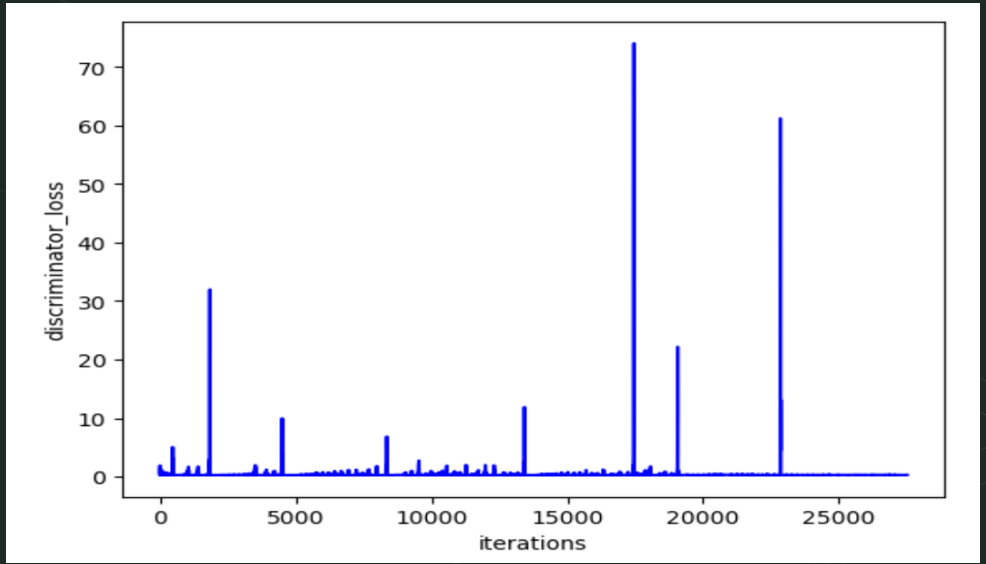
It's input size is of the image itself, 256x256x3, and with its architecture simply consists of Conv2D layers with activation of LeakyReLU. It outputs a simple 2D tensor of numbers between 0 – 1, with all near 1 meaning real image and near 0 meaning fake image.

Some Results



Graph representing loss of generator function vs no. of iterations for first 35000 iterations

Graph representing the loss of dicriminator vs no of iterations for the first 35000 iterations



THANK YOU!