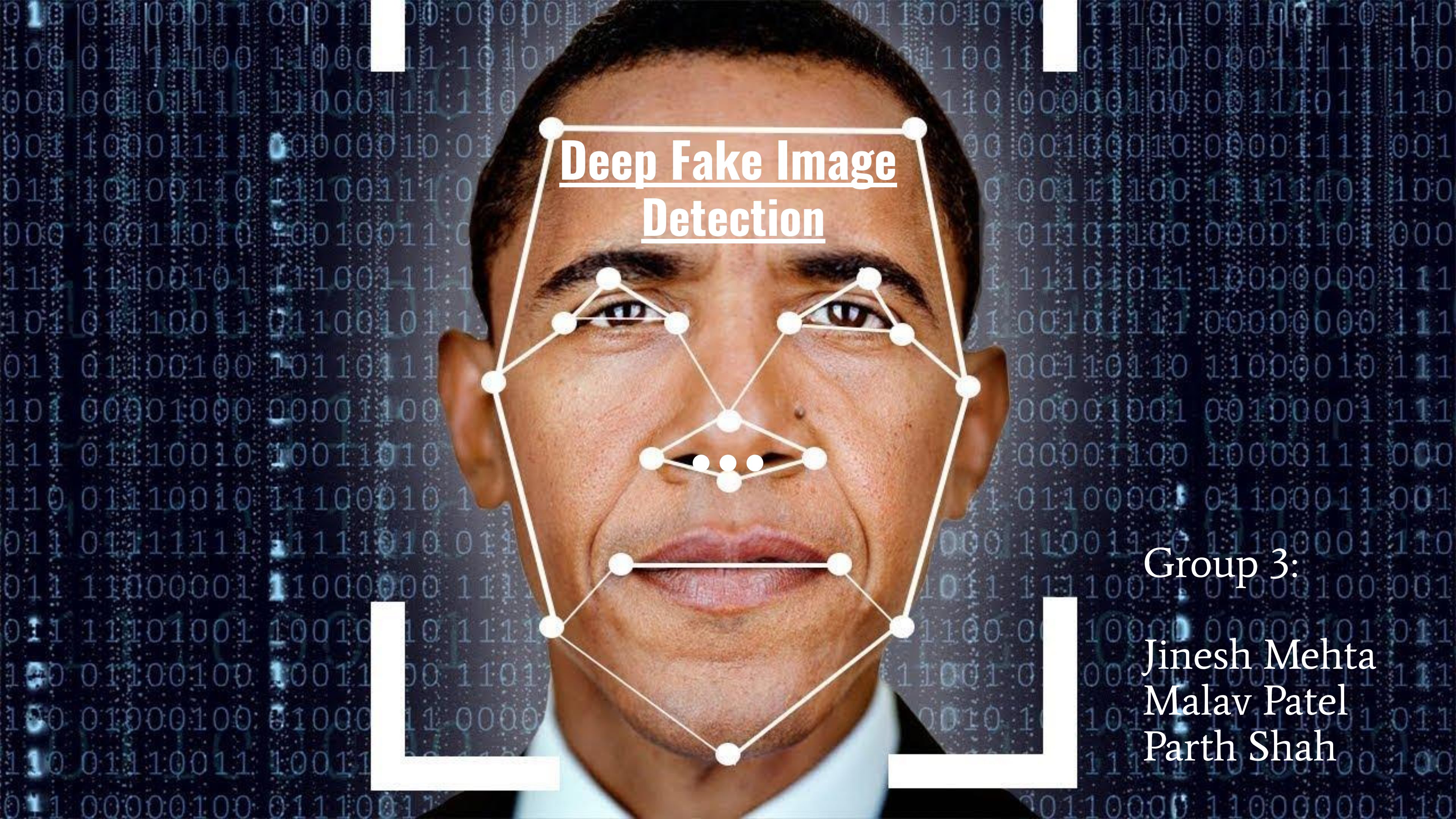


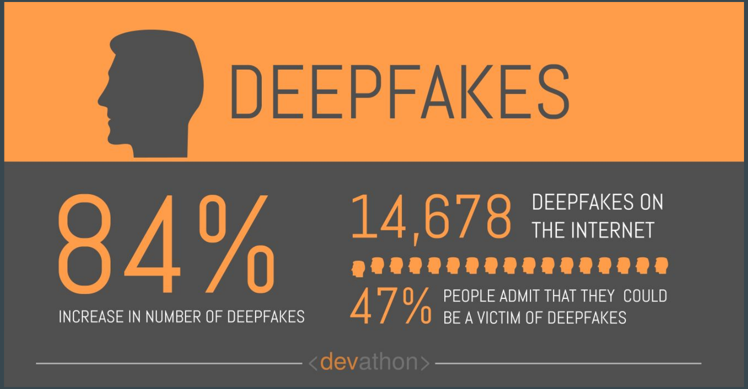
Deep Fake Image Detection

Group 3:

Jinesh Mehta
Malav Patel
Parth Shah

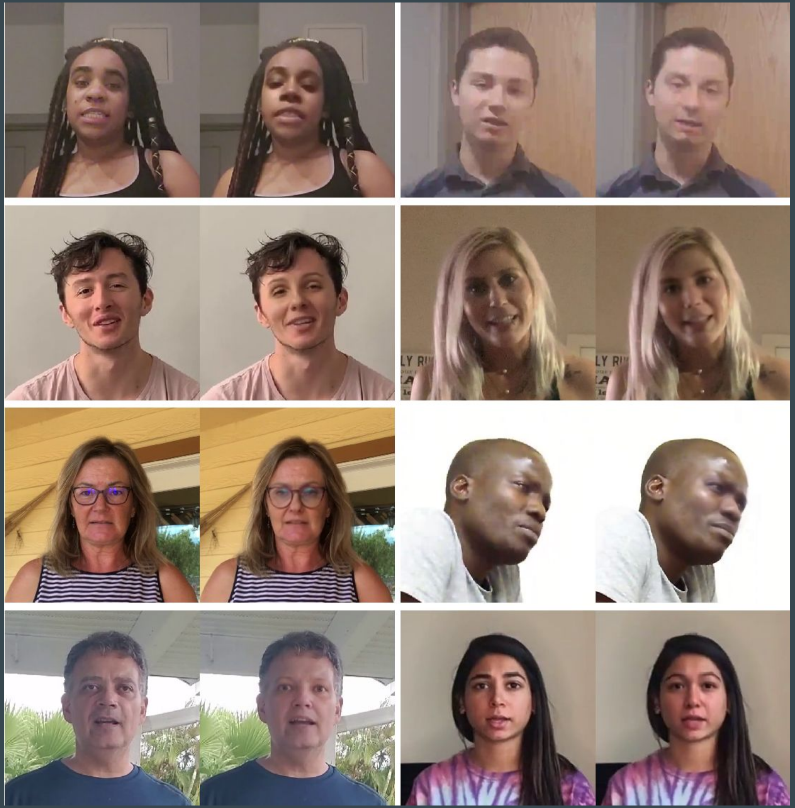
Motivation

- Deep fakes can influence how people assess the legitimacy of online information
- Their potential for malicious use as a source of false information can harm the quality of public discourse and the defense of human rights
- Recent developments in GANs have made it much simpler to generate deep fakes of purposeful distortions to produce manipulated images
- Contrarily, CNNs can be used to detect these GAN-generated deep fakes leave
- The goal of this project is to create new models that aid in the detection of deep fakes


DEEPFAKES
84%
INCREASE IN NUMBER OF DEEPFAKES
14,678
DEEPFAKES ON THE INTERNET
47%
PEOPLE ADMIT THAT THEY COULD BE A VICTIM OF DEEPFAKES
<devathon>

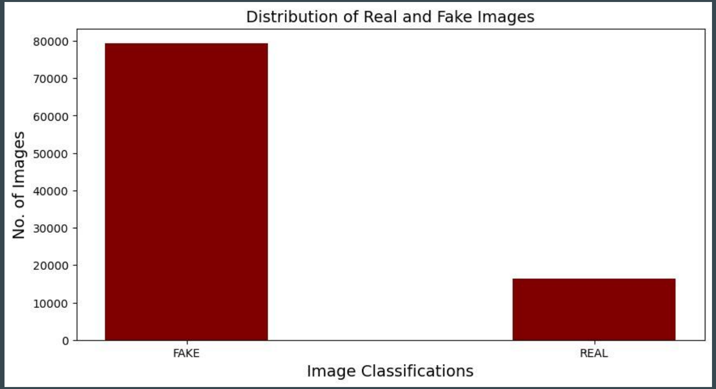
Description

- Objective is to address the issue of deep fake image detection using machine learning classifier models
- The problem is limited to the binary classification of images, which uses an image as input and an output prediction of its accuracy
- A special CNN created to recognize distinguishable features in fake images makes up the first tier
- A pre-trained Xception model is used in the second phase for transfer learning that produces a Real/Fake classification
- We produced an accuracy of 82.66% in a sizable and varied dataset of sophisticated Deep fake images produced by GAN
- This project lays the groundwork for future news organizations and social media platforms to stop the proliferation or abuse of blatantly false content

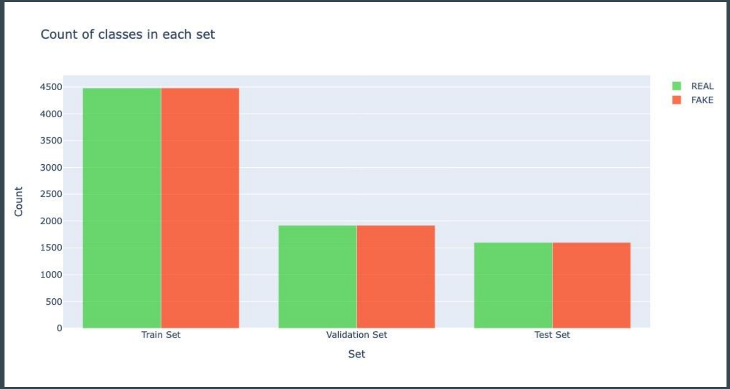

Examples of Real Images (left) and their corresponding Deep Fakes (right)

Dataset

- Over 470GB of mp4 videos make up the original dataset, and a 20GB sample showed that about 83% of the examples are deep fakes
- We made an effort to use a sizable dataset of uniformly sized images, each of which was labeled Real/Fake and had an 80-20 ratio between real and fake images
- The faces were detected using the Python facial recognition package and Dlib from the first frame of the videos and scaled to (224 x 224) pixels since many pre-trained models have this size, and randomly flipped or transformed
- The original size has been added as metadata as it affects blurring and grading
- Each real image yields an average of around 5 fake images per real image
- This arbitrary selection is also the median, implying that half of the faces were enlarged and half were reduced
- This is about 80% of the training set. The remaining 20% were ignored because no faces or multiple faces were detected
- Additionally, 3-4 deep fakes were added to the dataset for each real image
- This dataset contains 95,634 face images, but we only used 16,000 of them due to processing power constraints



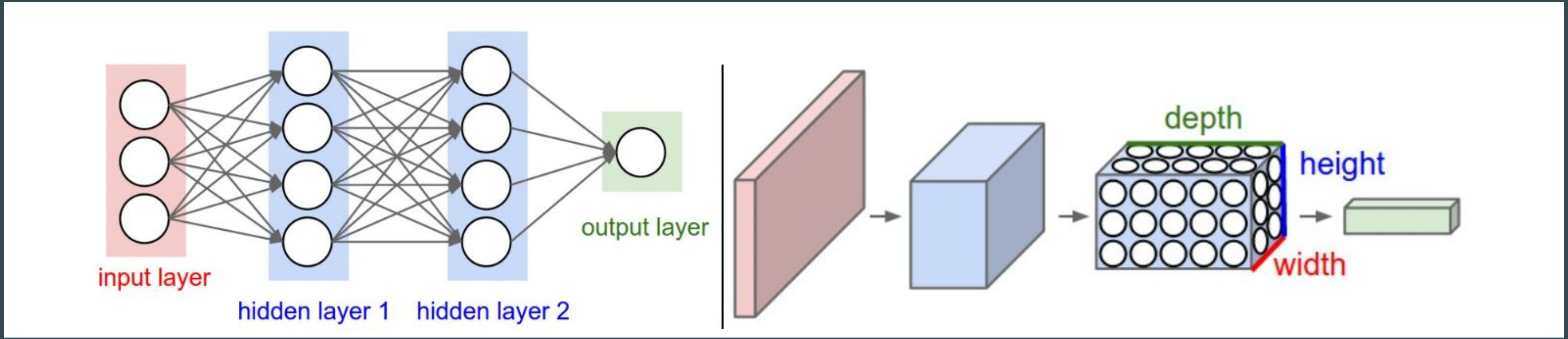
original dataset was distorted, with more fake images (79341) than real images (16293)



Sampled data to get an even mix of real and fake images, and split into train, test and validation set

Methodology - Overview of CNNs

Convolutional Neural Network (CNNs)

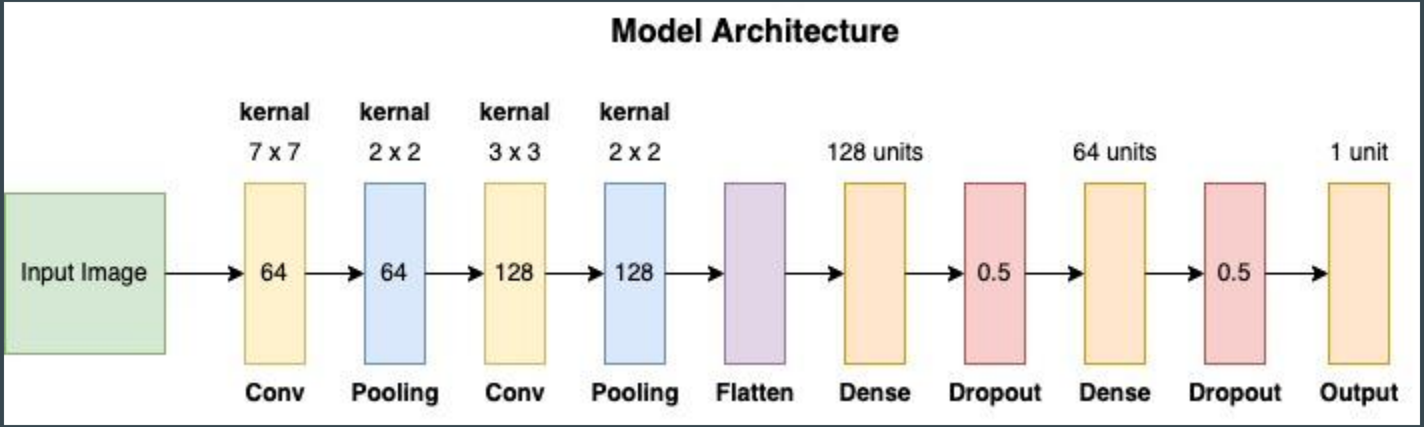


Different types of layers used in CNNs:

- Convolutional Layer
- Pooling Layer
- Dense Layer

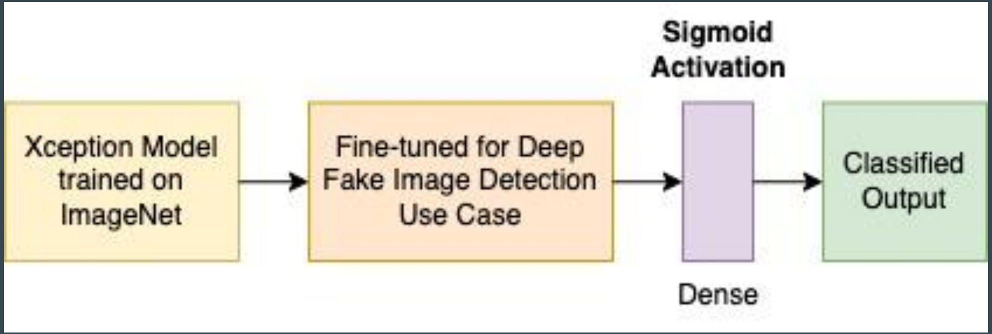
Approach 1: Building the model from scratch

We first set the baseline of our approach by building a neural network from scratch with the help of TensorFlow and Keras. The architecture of our model is as shown in the below diagram:



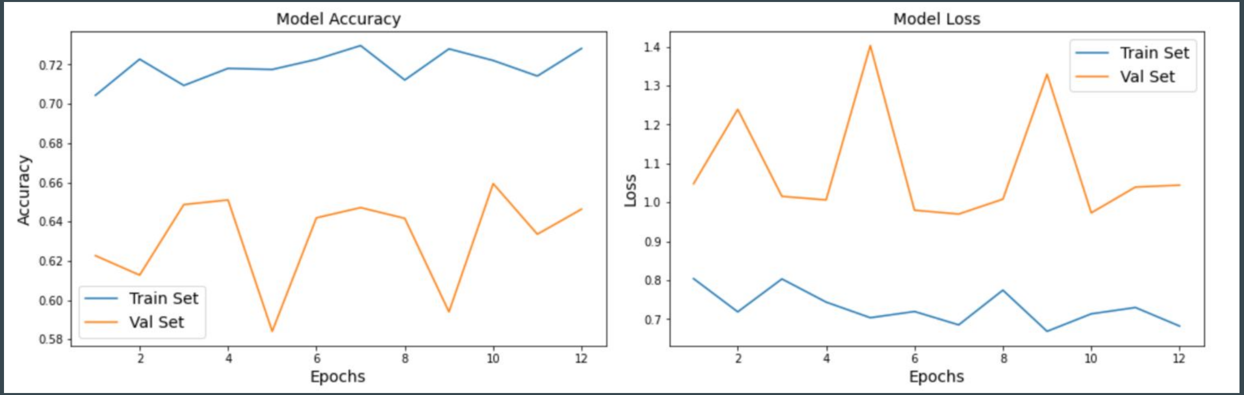
Approach 2: Transfer Learning using ImageNet & Xception

To utilize the existing advanced deep learning pre-trained models, we use another approach where we initialize the initial weights of the pooling layer with pre-trained weights from the pre-trained Xception Model on the ImageNet dataset. We then fine-tune the model and use a dense layer with a sigmoid as an activation function and generate the classified output.


Xception Model trained on ImageNet
Fine-tuned for Deep Fake Image Detection Use Case
Sigmoid Activation
Dense
Classified Output

Ablation Study

- We have experimented by changing multiple hyper parameters and while trying to get the best accuracy. Some of the hyperparameters which were tuned during this process:
    - **Learning Rate :** Experimented with multiple variations of learning rate (0.1,0.01).
    - **Optimizers :** Experimented with multiple optimizers like SGD, Adam and Nadam.
    - **Number of neurons in the Dense Layers** : Used Keras Tuner library to find the ideal number of neurons as well as learning rate.
    - **Dropout**: Changed dropout rate in order to reduce the overfitting of the network on the images.



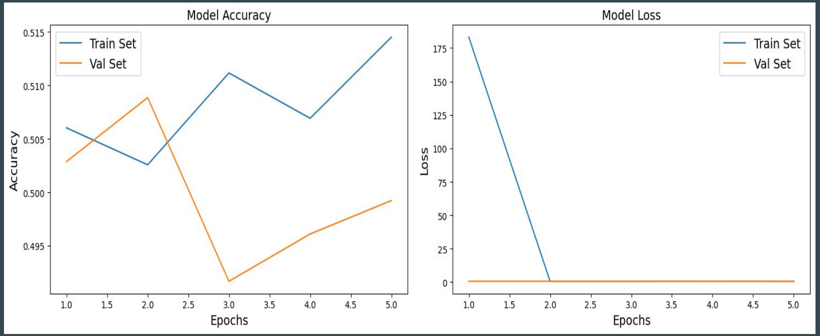
One of the trial performed to achieve desired accuracy ()

Optimizer	Learning rate	Score	Model type
Nadam	0.01	[0.7, 0.54]	Custom CNN
SGD	0.01	[0.70, 0.50]	Custom CNN
Nadam	0.01	[0.64, 0.66]	Augmented Data + Pre-trained model
Adam	0.1	[1.08, 0.63]	Augmented Data + Pre-trained model
SGD	0.1	[1.02, 0.64]	Augmented Data + Pre-trained model
SGD	0.01	[0.66, 0.82]	Augmented Data + Pre-trained model

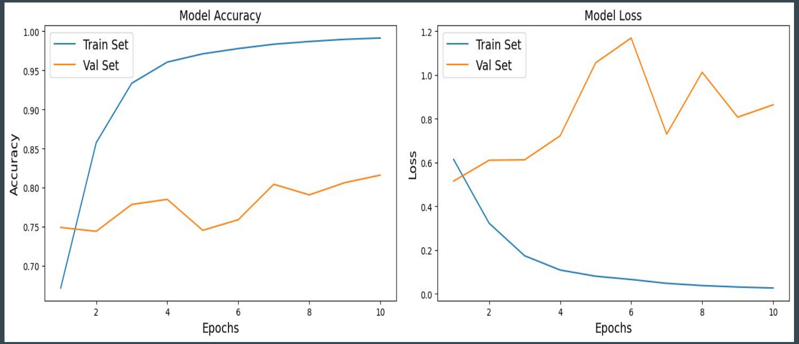
Results

- We used Keras' Evaluate() function on the model to get the result. The evaluate function takes two arguments: input data and target data. It returns a scalar test loss if the model has a single output and no metrics; otherwise, it returns a list of scalars. Below are the graphs which shows results obtained using custom CNN model and pre-trained Xception model.

Custom CNN
[0.6943774223327637, 0.4962500035762787]



Xception Model
[0.7960813641548157,0.8265624642372131]



Conclusion

- Learned about how the speed of training a model as well as space management differed in case of change in the machine as well using the right configurations. As a result, we finally decided to use TensorFlow GPU on our local machines.
- Developed our first custom CNN model which provided low accuracy which compelled us to work on hyperparameters and look into how a pre-trained model can help in increasing the accuracy.
- Using Transfer learning and the Xception Model, we created a high-performance Deepfake classifier, which boosted our final accuracy by roughly 30%, from 49.63% in basic CNN to 82.66% in the pre-trained Xception model using Transfer Learning.
- This model can also extend to multi-modal DeepFakes.
- Deepfakes in videos, audio tempering, scan images in the healthcare domain, and other applications are possible with this approach.
- The main constraints on our local machines were time and space. High-processing-power machines can improve performance by allowing more data to be used for training and yield better results on test datasets.

Responsibilities

Overall all three members of the group have contributed roughly proportionally to this project. All the tasks have been discussed and looked over by each member.

The following is the task distribution:

- Preprocessing and Exploratory Data Analysis - Jinesh Mehta
- Modeling - Parth Shah
- Performance metrics - Malav Patel
- Documentation and Presentation - Everyone

Thank You!