PROJECT REPORT

To Design and develop a Deep Learning Model to classify the video as deep fake or pristine.

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING (INFORMATION TECHNOLOGY)

6TH SEMESTER

Supervisor:

Dr. Neelam Goel
Assistant Professor (IT) UIET, PU

Submitted By:

Abhishek Naval [UE208010]
Himanshu Aggarwal [UE208043]
Kshitiz Sharma [UE208053]
Madhup Sharma [UE208057]

TO:

Department of Information Technology

University Institute of Engineering and Technology

Panjab University, Chandigarh

2023

Declaration

We hereby declare that the project report entitled "A SYSTEM TO DETECT VIDEO AS DEEPFAKE OR PRISTINE "submitted by us to University Institute of Information and Technology in partial fulfillment of the requirement for the award of the degree of Bachelors of Technology in Information Technology Department is a record of bona fide project work carried out by us under guidance of **Dr. Neelam Goel Assistant Professor (IT)** UIET PU.

We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Name: Abhishek Naval (UE208010)

Himanshu Aggarwal (UE208043)

Kshitiz Sharma (UE208053)

Madhup Sharma (UE208057)

Signature:

Acknowledgement

Whenever a module of work is completed successfully, a source of motivation and guidance is always there for student. This Project is Golden Opportunity for learning about upcoming new Technologies and Self Development. We hereby take the opportunity to thank all the wonderful people who provided this golden opportunity to us.

We would like to express our gratitude towards our respected **Dr. Neelam Goel**, **Assistant Professor (IT)** UIET, PU for the guidance and help we received from her during this Project Report. We as a team are highly obliged and appreciate him for her timely help and guidance.

TABLE OF CONTENTS

1. Introduction

1.1 Idea/Motivation

With the advent of new technological enhancements in artificial intelligence, new sophisticated AI techniques are used to create fake videos. Such videos can pose a great threat to the society in various social and political ways and can be used for malicious purposes. These fake videos are called deepfakes. Deepfakes refer to manipulated videos, or other digital representations produced by sophisticated artificial intelligence, that yield fabricated images and sounds that appear to be real. A deep-learning system can produce a persuasive counterfeit by studying photographs and videos of a target person from multiple angles, and then mimicking its behaviour and speech patterns. Detecting these videos is a massive problem because of the increasing developments in more realistic deepfake creation technologies emerging every now and then.

Forging of data is nothing new in this era having a backbone made up of artificial intelligence and machine learning. Distortion of reality is becoming a huge problem these days as more and more fake images and videos are emerging every day on the internet. These deep fakes are getting better with time, to the extent that they cannot be distinguished as fake or real by the human eye, hence are increasingly resistant to detection.

While deep-fake technology will bring with it certain benefits, it also will introduce many harms. In an era brimming with so much truth decay, nothing is more dangerous than people taking such videos at their face value. These deep fakes can be used for various malicious purposes such as defamation of celebrities, creating political bias, personal sabotage, intimidation and exploitation, false propaganda, piracy and other vengeful activities. The most targeted feature in a deepfake is the face. Thus, many algorithms and techniques can be used to identify manipulation of faces. These face manipulations can be of two types Expressions and Identity. In the first type, the expressions of one person are transferred to another in real time. In identity manipulation, faces of two people are swapped. This type of manipulation can be used to spread false information among public by swapping with the face of a famous person.

1.2 Issues / Challenges in current scenario

1) Accuracy

2) Data set of people

3) Identifying facial features

4) Detecting faces from video input

1.3 Project Objective and Motivation:

The Study field will be a distorted grouping. This area is the mechanism by which minor imperfections are found in doctored videos and exposes the false representation of originality. The creation of a doctored image/video needs information to be examined in order to reveal the dishonest, particularly facial expression variables. It is defined as an in-depth image/video study to find minor imperfections such as boundary points, background incoherence, double eyebrows or irregular twitch of the eye. The impetus behind this research is to recognise these distorted media, which is technically demanding and which is rapidly evolving. Many engineering companies have come together to unite a great deal of dataset. Competitions and actively include data sets to counter deepfakes. Deepfake videos are now so popular that multiple political parties utilise this tool to produce faked images of the leader of their opposing party to propagate hate against them. Fake political videos telling or doing things that have never happened is a threat to election campaigns. These images are the primary source of false media controversies and propagate misleading news.

2. Technology Used

2.1 Hardware

In this project, a computer with sufficient processing power is needed. This project requires too much processing power, due to the image and video batch processing.

- **Client-side Requirements:** Browser: Any Compatible browser device

Sr. No.	Parameter	Minimum Requirement
1	Intel i3	1005G1 CPU @ 1.20GHz 1.19GHz
2	RAM	8 GB
3	Graphic card	NVIDIA GeForce GTX Titan (12 GB RAM)

Table 2.1

2.2 Software

Git: For version Control

For Model Training:

1. Python
2. Artificial intelligence
3. Jupyter Notebook
4. Colab / Tenserflow

3. Software/System Analysis

3.1 Requirement Gathering

We analysed the problem statement and found the feasibility of the solutionof the problem. We read different research paper. After checking the feasibility of the problem statement. The next step is the data-set gathering and analysis. We analysed the data set in different approach of training like negatively or positively trained i.e training the model with onlyfake or real video's but found that it may lead to addition of extra bias in the model leading to inaccurate predictions. So, after doing lot of research, we found that the balanced training of the algorithm is the best way to avoid the bias and variance in the algorithm and get a good accuracy.

- Solution Constraints

We analyzed the solution in terms of cost, speed of processing, requirements, levelof expertise, availability of equipment's.

- Parameter Identified

1. Blinking of eyes
2. Teeth enchantment
3. Bigger distance for eyes
4. Moustaches
5. Double edges, eyes, ears, nose
6. Iris segmentation
7. Wrinkles on face
8. Inconsistent head pose
9. Face angle
10. Skin tone
11. Facial Expressions
12. Lighting
13. Different Pose
14. Double chins
15. Hairstyle
16. Higher cheek bones

3.1.1 Design

After research and analysis, we developed the system architecture of the solutionas mentioned. We decided the baseline architecture of the Model which includes the different layers and their numbers.

3.1.2 Development

After analysis we decided to use the PyTorch framework along with python3 language for programming. PyTorch is chosen as it has good support to CUDA i.e Graphic Processing Unit (GPU) and it is customize-able. Google Cloud Platformfor training the final model on large number of data-set.

3.1.3 Evaluation

We evaluated our model with a large number of real time dataset which include YouTube videos dataset. Confusion Matrix approach is used to evaluate the accuracyof the trained model.

3.1.4 Outcome

The outcome of the solution is trained deepfake detection models that will help the users to check if the new video is deepfake or real.

3.1.5 Applications

The model will pre-process the video and predict whether the uploaded video is a deepfake or real video.

3.2 Feasibility Study

3.2.1 Identification

Before the training, we need to prepare thousands of images for both persons. We can take a shortcut and use a face detection library to scrape facial pictures fromtheir videos. Spend significant time to improve the quality of your facial pictures. It impacts your final result significantly.

1. Remove any picture frames that contain more than one person.
2. Make sure you have an abundance of video footage. Extract facial picturescontain different pose, face angle, and facial expressions.
3. Some resembling of both persons may help, like similar face shape.

3.2.2 Analysis

In Deepfakes, it creates a mask on the created face so it can blend in with the target video. To further eliminate the artifacts

1. Apply a Gaussian filter to further diffuse the mask boundary area.
2. Configure the application to expand or contract the mask further.
3. Control the shape of the mask.

3.3 Design/use case diagrams/ Flowcharts

3.3.1 Use Case View

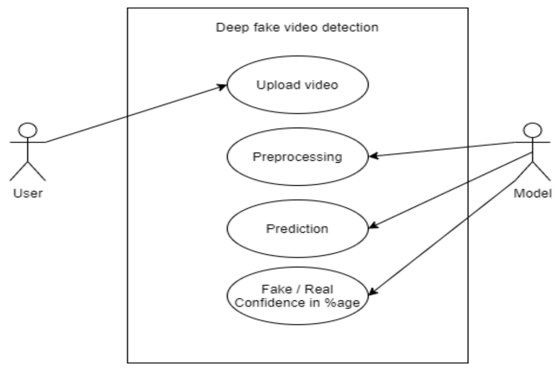


Figure 3.1: Use case diagram

Functional Model and Description

A description of each major software function, along with data flow (structured anal- ysis) or class hierarchy (Analysis Class diagram with class description for object-oriented system) is presented.

3.3.2 Data Flow Diagram

DFD Level-0

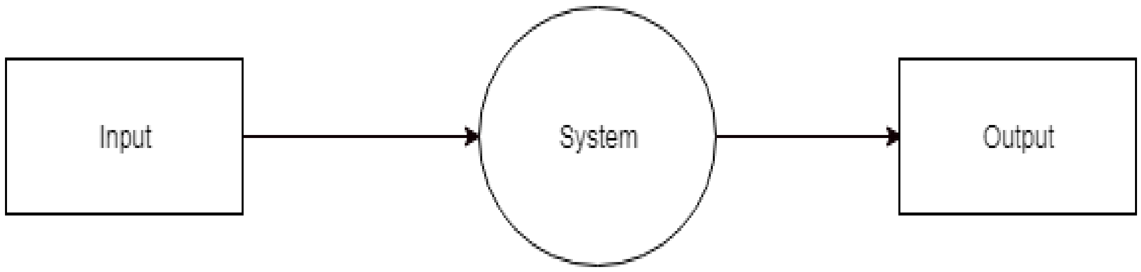


Figure 3.2: DFD Level 0

DFD level – 0 indicates the basic flow of data in the system. In this System Input isgiven equal importance as that for Output.

- Input: Here input to the system is uploading video.
- System: In system it shows all the details of the Video.
- Output: Output of this system is it shows the fake video or not.

  Hence, the data flow diagram indicates the visualization of system with its inputand output flow.

DFD Level-1

DFD Level – 1 gives more in and out information of the system.
Where system gives detailed information of the procedure taking place.

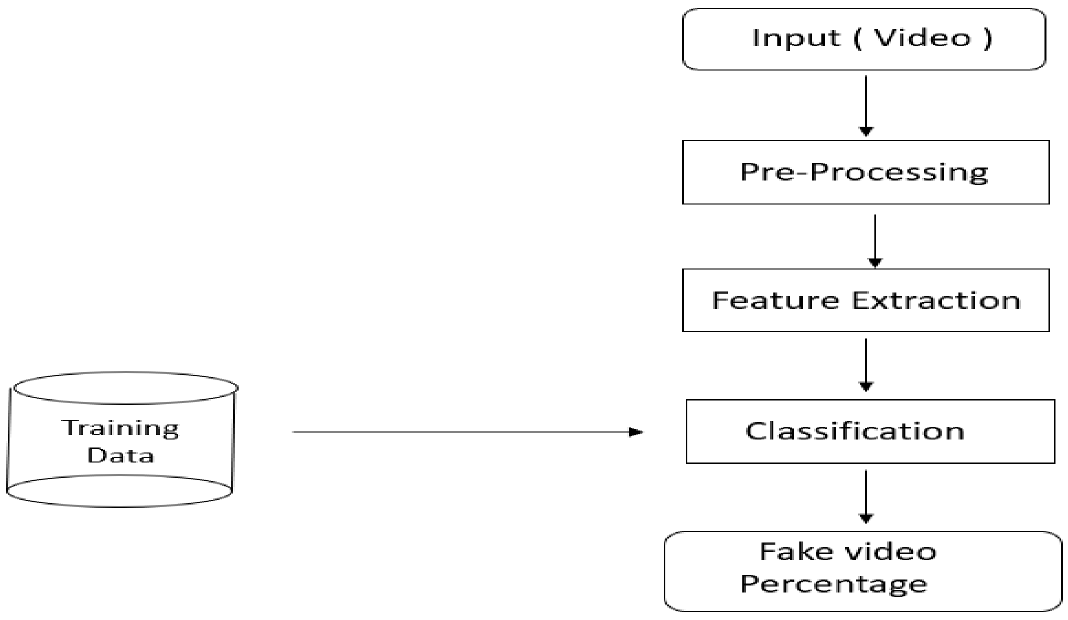


Figure 3.3: DFD Level 1

DFD Level-2

[1] DFD level-2 enhances the functionality used by user etc.

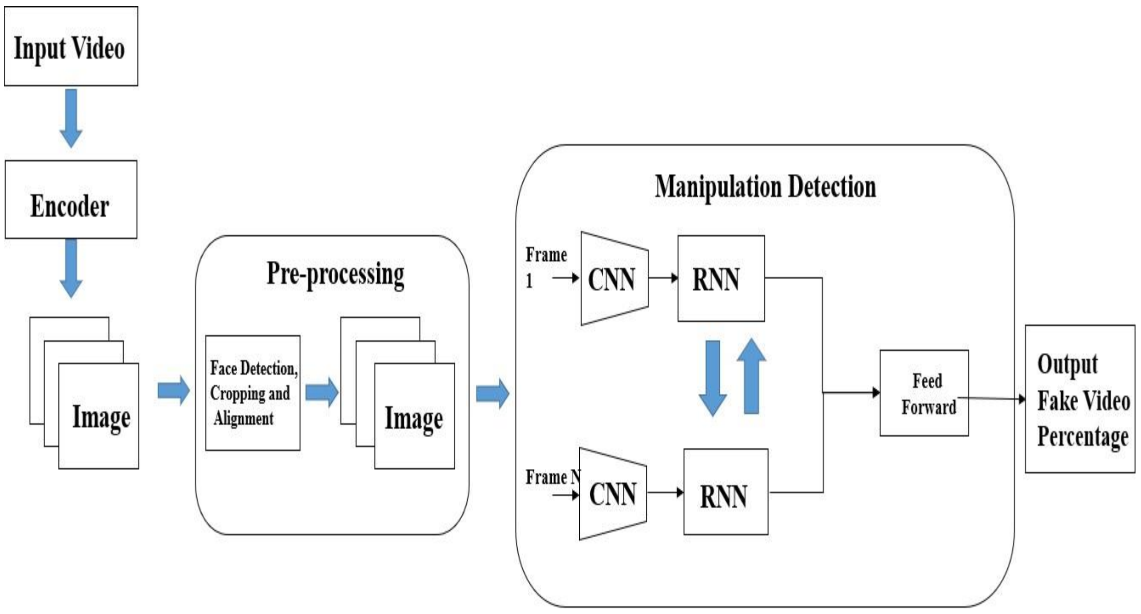


Figure 3.4: DFD Level 2

3.3.3 Activity Diagram:

Training Workflow:

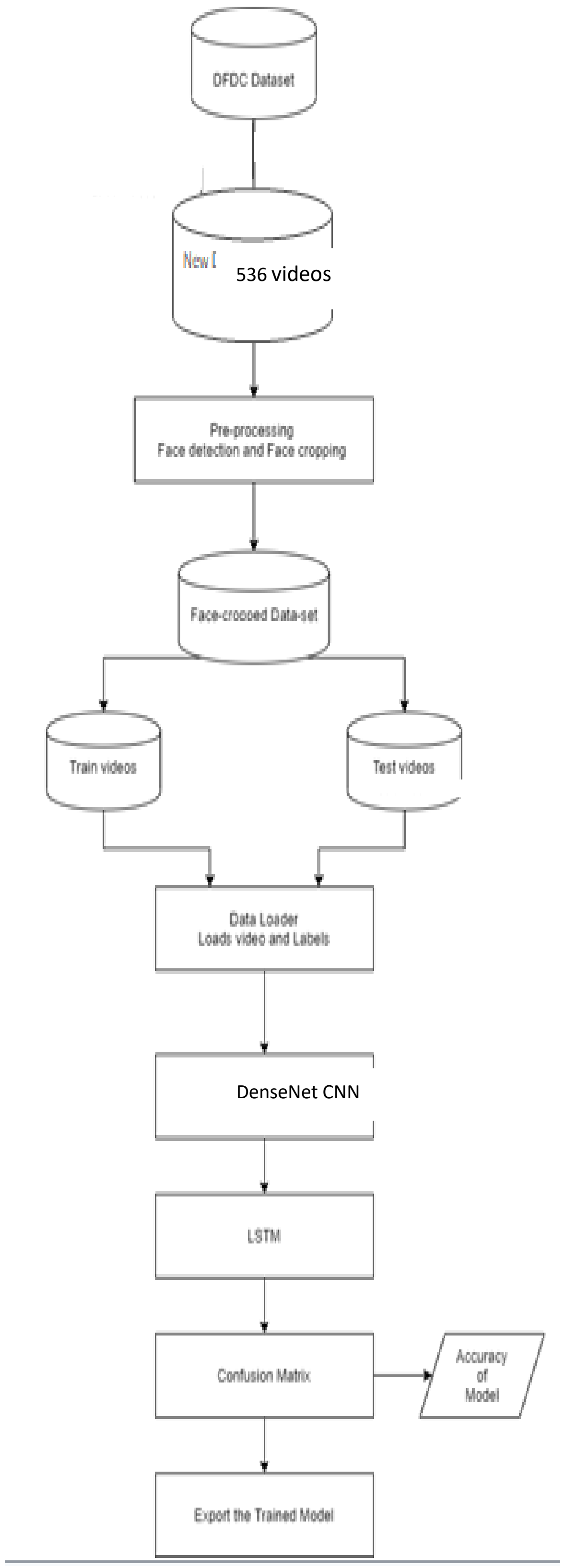


Figure 3.5: Training workflow

3.3.4 Testing Workflow:

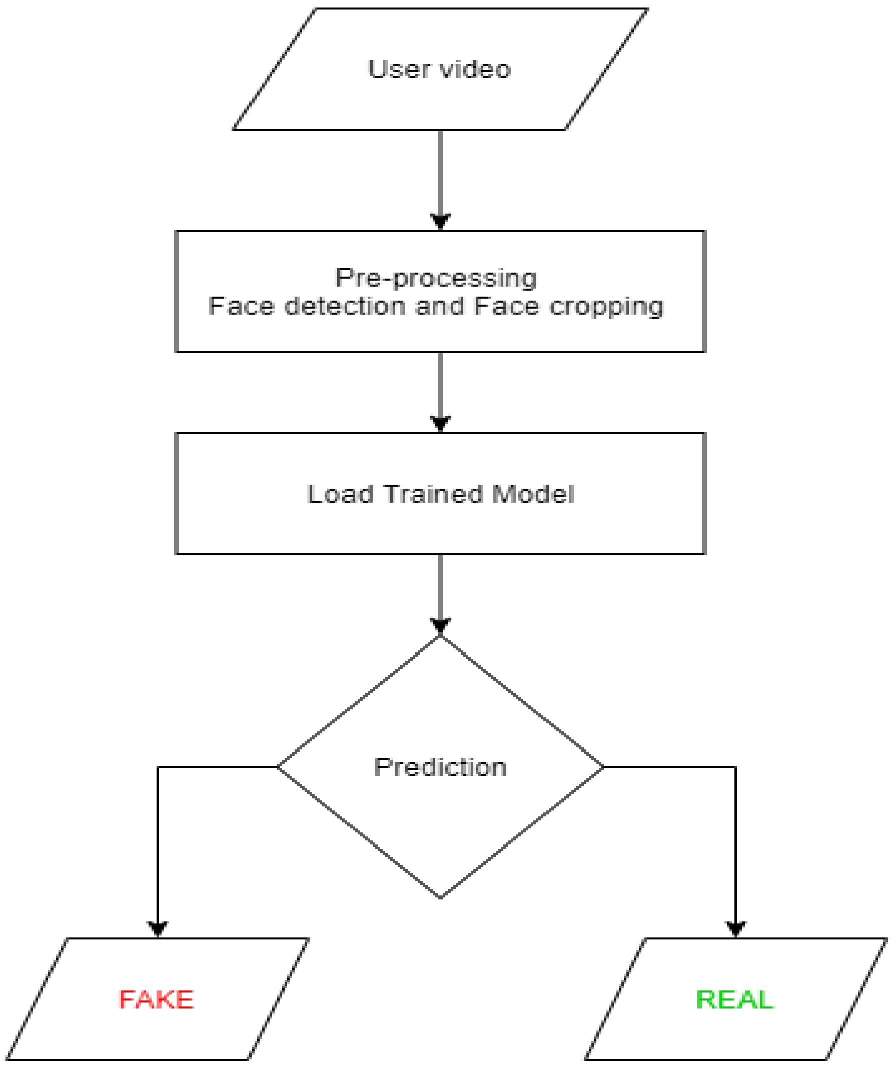


Figure 3.6: Testing Workflow

Non-Functional Requirements:

Performance Requirement

- The software should be efficiently designed so as to give reliable recognition offake videos and so that it can be used for more pragmatic purpose.
- The design is versatile and user friendly.
- The application is fast, reliable and time saving.
- The system has universal adaptations.

Safety Requirement

- The Data integrity is preserved. Once the video is uploaded to the system. It is only processed by the algorithm. The videos are kept secured from the human interventions, as the uploaded video is not are not able for human manipulation.
- To extent the safety of the videos uploaded by the user will be deleted after 30 min from the server.

Security Requirement

- While uploading the video, the video will be encrypted using a certain symmetric encryption algorithm. On server also the video is in encrypted format only. The video is only decrypted from pre-processing till we get the output. After getting the output the video is again encrypted.
- This cryptography will help in maintain the security and integrity of the video.
- SSL certification is made mandatory for Data security.

3.3.5 Sequence Diagram

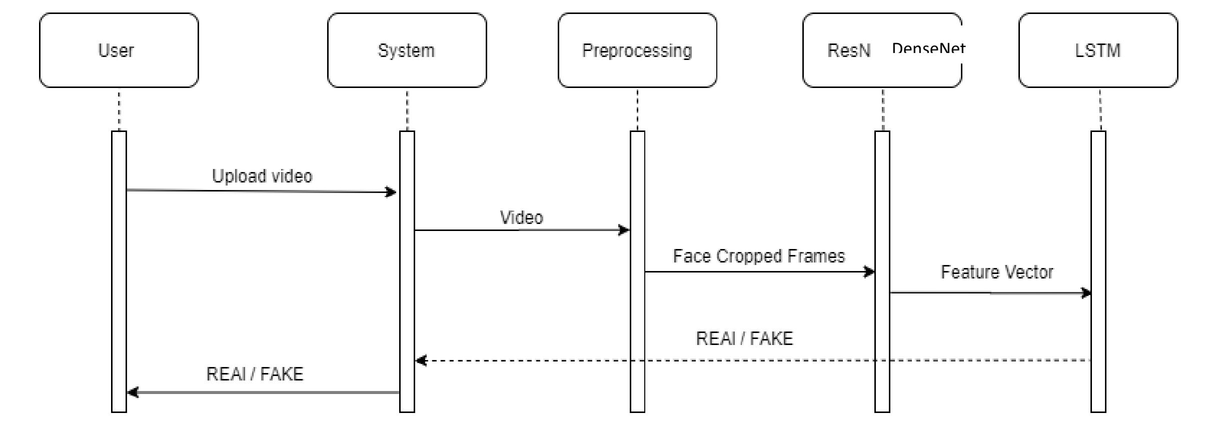


Figure 3.7: Sequence Diagram

3.4 Implementation

3.4.1 Introduction

There are many examples where deepfake creation technology is used to mis-lead the people on social media platform by sharing the false deepfake videos ofthe famous personalities like Mark Zuckerberg Eve of House A.I. Hearing, Don-ald Trump’s Breaking Bad series where he was introduces as James McGill, Barack Obama’s public service announcement and many more. These types of deepfakes creates a huge panic among the normal people, which arises the need to spot these deepfakes accurately so that they can be distinguished from the real videos.

Latest advances in the technology have changed the field of video manipulation. The advances in the modern open-source deep learning frameworks like TensorFlow,Keras, PyTorch along with cheap access to the high computation power has driven the paradigm shift. The Conventional autoencoders and Generative Adversarial Network (GAN) pretrained models have made the tampering of the realistic videos and images very easy. Moreover, access to these pretrained models through the smartphones and desktop applications like FaceApp and Face Swap has made the deepfake creation a childish thing. These applications generate a highly realistic synthesized transformation of faces in real videos. These apps also provide the user with more functionalities like changing the face hair style, gender, age and other attributes. These apps also allow the user to create a very high quality and indistin- guishable deepfakes. Although some malignant deepfake videos exist, but till now they remain a minority. So far, the released tools that generate deepfake videos are being extensively used to create fake celebrity pornographic videos or revenge porn. Some of the examples are Brad Pitt, Angelina Jolie nude videos. The real looking nature of the deepfake videos makes the celebraties and other fa- mous personalities the target of pornographic material, fake surveillance videos, fake news and malicious hoaxes. The Deepfakes are very much popular in creating the political tension. Due to which it becomes very important to detect the deepfake videos and avoid the percolation of the deepfakes on the social media platforms.

3.4.2 Tools and Technologies Used

Planning

1. OpenProject

Programming Languages

2. Python3

Programming Frameworks

3. PyTorch

IDE

4. Google Colab
5. Jupyter Notebook
6. Visual Studio Code

Versioning Control

7. Git

Cloud Services

8. Google Cloud Platform

Application and web servers:

9. Google Cloud Engine

Libraries

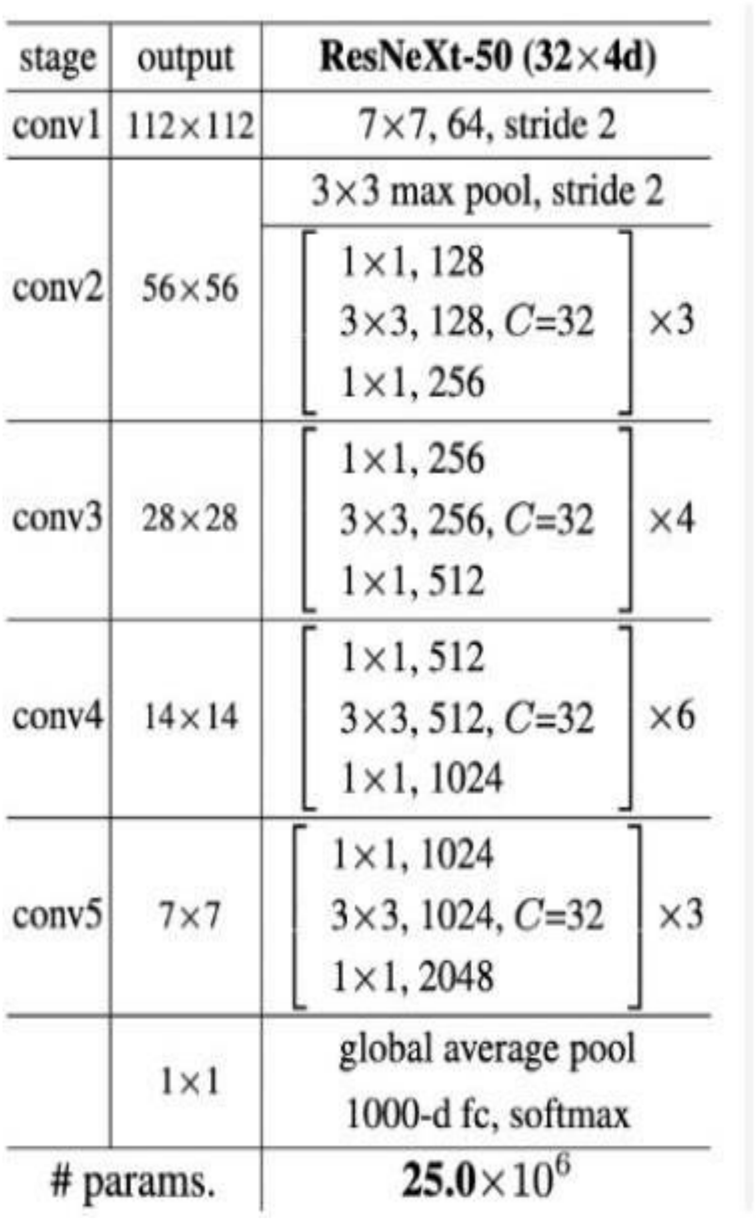
10. torch
11. torchvision
12. os
13. numpy
14. cv2
15. matplotlib
16. face_recognition
17. json
18. pandas
19. copy
20. glob
21. random
22. sklearn

3.4.3 Preprocessing Details

- Using glob we imported all the videos in the directory in a python list.
- cv2.VideoCapture is used to read the videos and get the mean number of framesin each video.
- To maintain uniformity, based on mean a value 150 is selected as idea value for creating the new dataset.
- The video is split into frames and the frames are cropped on face location.
- The face cropped frames are again written to new video using VideoWriter.
- The new video is written at 30 frames per second and with the resolution of 112 x 112 pixels in the mp4
- Instead of selecting the random videos, to make the proper use of LSTM for temporal sequence analysis the first 150 frames are written to the new video.

3.4.4 Model Details

ResNext CNN: The pre-trained model of Residual Convolution Neural Net- work is used. The model name is resnext50_32x4d() .This model consists of 50 layers and 32 x 4 dimensions. Figure shows the detailed implementation of model.



stage	output	ResNeXt-50 (32×4d)
conv1	112×112	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C=32 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3	28×28	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C=32 \\ 1\times1, 512 \end{bmatrix} \times 4$
conv4	14×14	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C=32 \\ 1\times1, 1024 \end{bmatrix} \times 6$
conv5	7×7	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C=32 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	global average pool 1000-d fc, softmax
# params.		**25.0**×10^6

Figure 3.8 ResNext Architecture

Ref:https://iq.opengenus.org/resnext/#:~:text=ResNext%20is%20a%20simple%20network,and%20the%20hyperparameters%20are%20%20redu

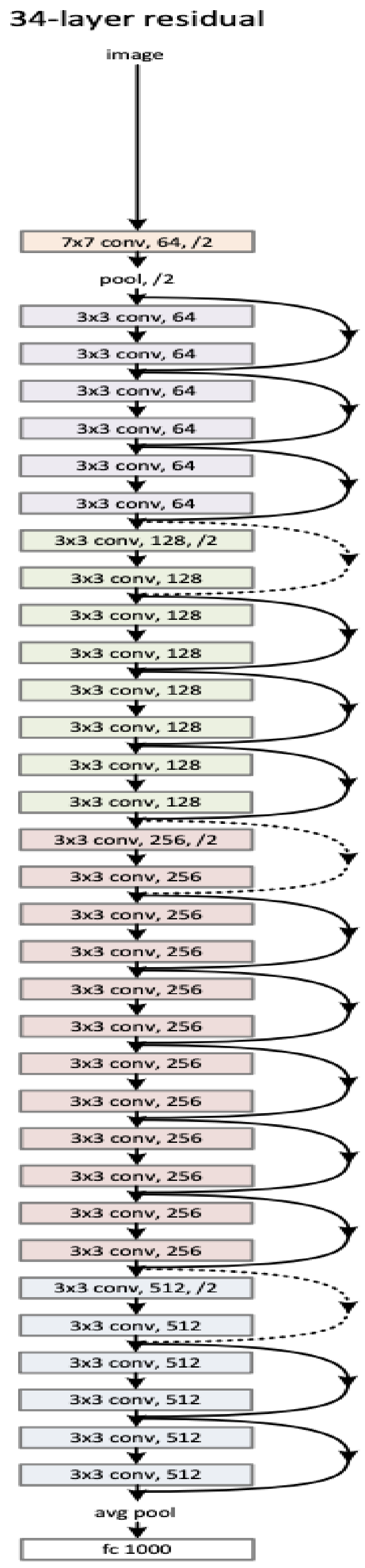


Fig. 3.9 Residual network with 34 parameter layers
Ref: https://viso.ai/deep-learning/resnet-residual-neural-network/

VGG16 CNN:

A convolutional neural network is also known as a ConvNet, which is a kind of artificial neural network. A convolutional neural network has an input layer, an output layer, and various hidden layers. VGG16 is a type of CNN (Convolutional Neural Network) that is considered to be one of the best computer vision models to date. The creators ofthis model evaluated the networks and increased the depth using an architecture with very small (3 × 3) convolution filters, which showed a significant improvement on the prior-art configurations. They pushed the depth to 16–19 weightlayers making it approx. — 138 trainable parameters.

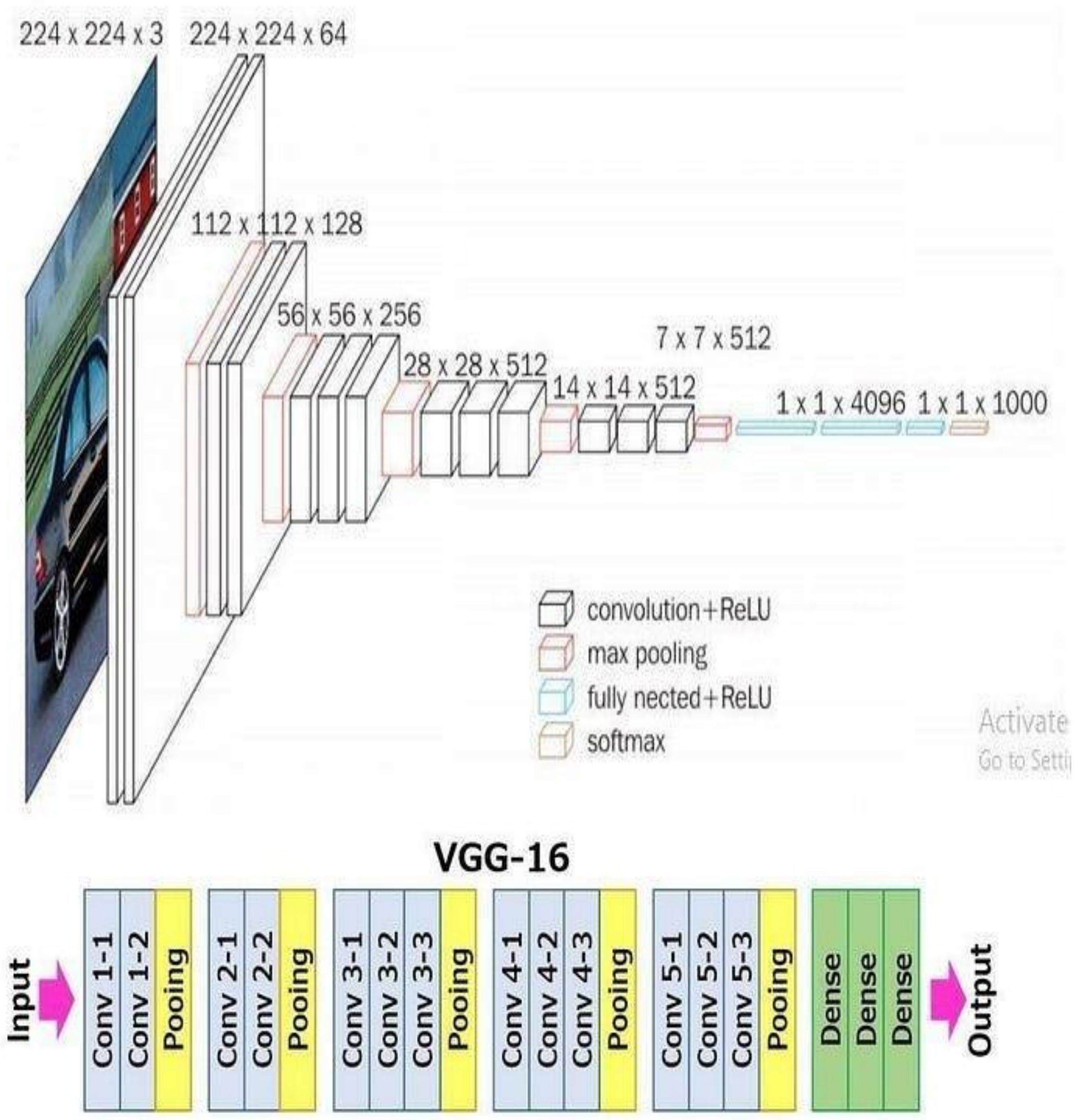


Figure 3.10: VGG16 Architecture

Ref: https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918

- The 16 in VGG16 refers to 16 layers that have weights. In VGG16 there are thirteen convolutional layers, five Max Poolinglayers, and three Dense layers which sum up to 21 layers but it has only sixteen weight layers i.e., learnable parameters layer.

- VGG16 takes input tensor size as 224, 244 with 3 RGB channel

- Most unique thing about VGG16 is that instead of having a large number of hyper-parameters they focused on having convolution layers of 3x3 filter with stride 1 and always used the same padding and maxpool layer of 2x2 filter of stride 2.

- The convolution and max pool layers are consistently arranged throughout the whole architecture

- Conv-1 Layer has 64 number of filters, Conv-2 has 128 filters, Conv-3 has 256 filters, Conv 4 and Conv 5 has 512 filters.

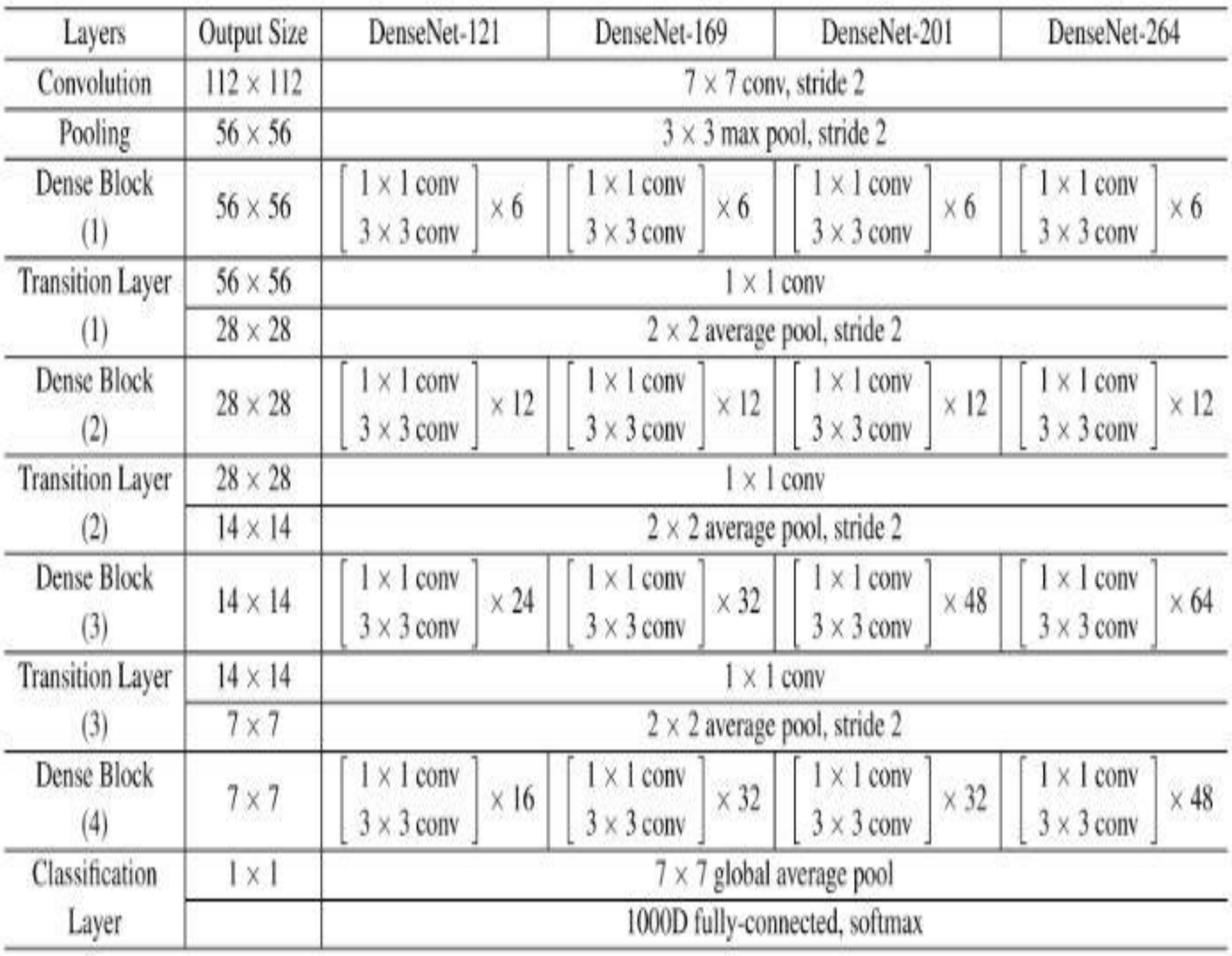
- Three Fully-Connected (FC) layers follow a stack of convolutional layers: the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-maxlayer.

DenseNet CNN: The pre-trained model of Residual Convolution Neural Net- work is used. The model's name is DenseNet121. Here we have removed last two layers of the DenseNet model that are AdaptiveAvgPool2d layer and linear layer also known as fully connected layer which together performs classification.

DenseNet-121 has the following layers:

1. 1 7x7 Convolution
2. 58 3x3 Convolution
3. 61 1x1 Convolution
4. 4 AvgPool
5. 1 Fully Connected Layer

In a traditional feed-forward Convolutional Neural Network (CNN), each convolutional layer except the first one (which takes in the input), receives the output of the previous convolutional layer and produces an output feature map that is then passed on to the next convolutional layer. Therefore, for 'L' layers, there are 'L' direct connections; one between each layer and the next layer.



Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv			
	28×28	2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv			
	14×14	2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1×1	7×7 global average pool			
		1000D fully-connected, softmax			

Figure 3.11: DenseNet Architecture

Ref: https://iq.opengenus.org/architecture-of-%20densenet121/#:~:text=In%20short%2C%20DenseNet%2D121%20has%20120%20Convolutions%20and%204%20AvgPo%20ol,use%20features%20extracted%20early%20on.

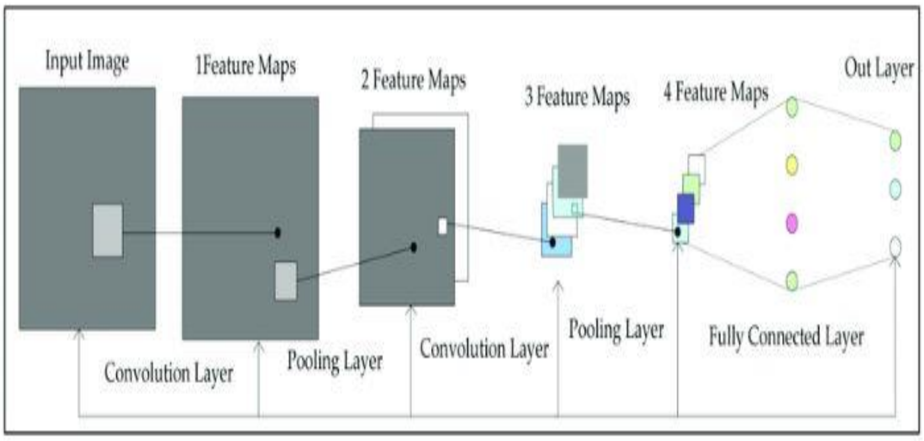


Figure 3.12: Overview of DenseNet Architecture

Ref: https://iq.opengenus.org/architecture-of-%20densenet121/#:~:text=In%20short%2C%20DenseNet%2D121%20has%20120%20Convolutions%20and%204%20%20AvgPool,use%20features%20extracted%20early%20on

However, as the number of layers in the CNN increase, i.e. as they get deeper, the 'vanishing gradient' problem arises. This means that as the path for information from the input to the output layers increases, it can cause certain information to 'vanish' or get lost which reduces the ability of the network to train effectively.
DenseNets resolve this problem by modifying the standard CNN architecture and simplifying the connectivity pattern between layers. In a DenseNet architecture, each layer is connected directly with every other layer, hence the name Densely Connected Convolutional Network. For 'L' layers, there are L(L+1)/2 direct connections.

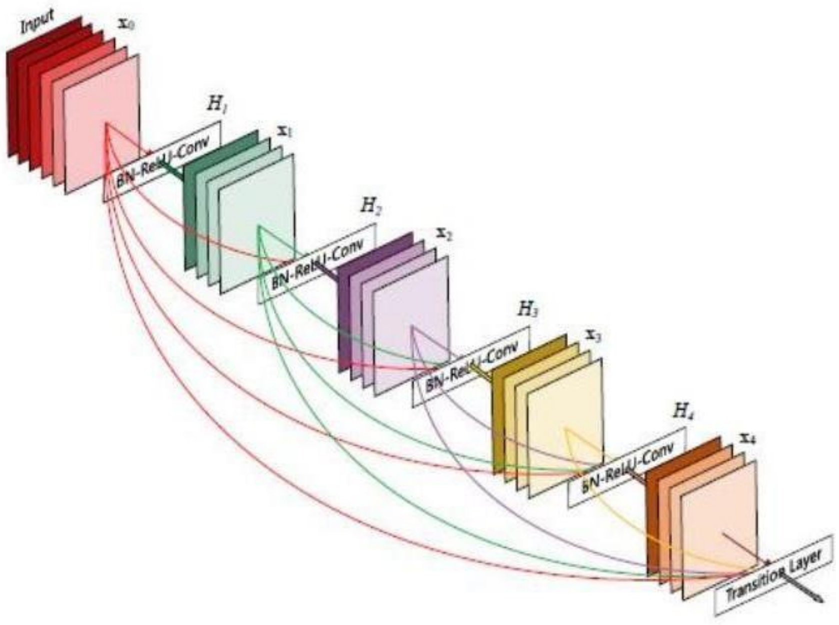


Figure 3.13

Ref:https://iq.opengenus.org/architectureof%20densenet121/#:~:text=In%20short%2C%20DenseNet%2D121%20has%20120%20Convolutions%20and%204%20AvgPool,use%2%200features%20extracted%20early%20on

Connectivity:

In each layer, the feature maps of all the previous layers are not summed, but concatenated and used as inputs. Consequently, DenseNets require fewer parameters than an equivalent traditional CNN, and this allows for feature reuse as redundant feature maps are discarded. So, the l^{th} layer receives the feature-maps of all preceding layers, $x_0,...,x_{l-1}$, as input:

$$\mathbf{x}_\ell = H_\ell([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}]),$$

where $[x_0,x_1,...,x_{l-1}]$ is the concatenation of the feature-maps, i.e. the output produced in all the layers preceding l (0,...,l-1). The multiple inputs of H_l are concatenated into a single tensor to ease implementation.

DenseBlocks:

The use of the concatenation operation is not feasible when the size of feature maps changes. However, an essential part of CNNsis the down-sampling of layers which reduces the size of feature-maps through dimensionality reduction to gain higher computation speeds.

To enable this, DenseNets are divided into DenseBlocks, where the dimensions of the feature maps remains constant within ablock, but the number of filters between them is changed. The layers between the blocks are called Transition Layers which reduce the the number of channels to half of that of the existing channels.

For each layer, from the equation above, H_l is defined as a composite function which applies three consecutive operations: batch normalization (BN), a rectified linear unit (ReLU) and a convolution (Conv).

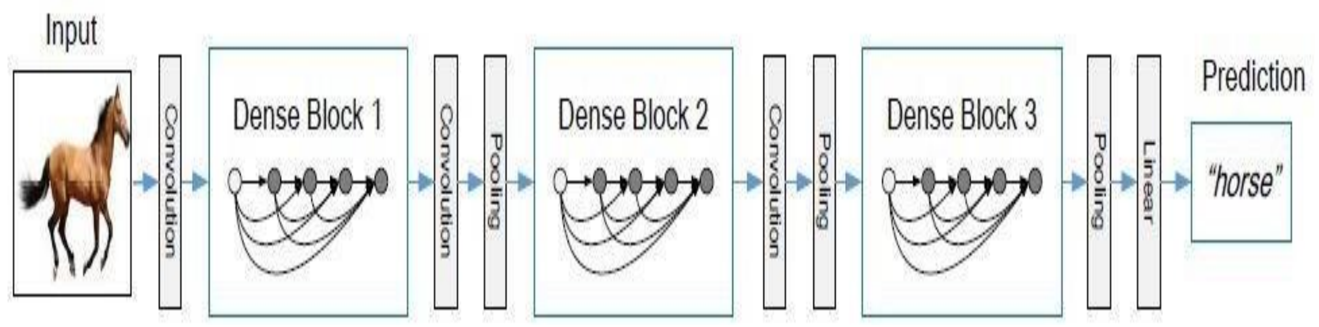


Figure 3.14

Ref: https://iq.opengenus.org/architecture-of-%20densenet121/#:~:text=In%20short%2C%20DenseNet%2D121%20has%20120%20Convolutions%20and%204%20AvgPool,use%2%200features%20extracted%20early%20on

In the above image, a deep DenseNet with three dense blocks is shown. The layers between two adjacent blocks are the transition layers which perform downsampling (i.e. change the size of the feature-maps) via convolution and pooling operations, whilst within the dense block the size of the feature maps is the same to enable feature concatenation.

Growth Rate: One can think of the features as a global state of the network. The size of the feature map grows after a pass through each dense layer with each layer adding 'K' features on top of the global state (existing features). This parameter 'K' is referred to as the growth rate of the network, which regulates the amount of information added in each layer of the network. If each function H $_l$ produces k feature maps, then the layer has

$$k_l = k_0 + k * (l - 1)$$

input feature-maps, where k0 is the number of channels in the input layer. Unlike existing network architectures, DenseNets can have very narrow layers.

Bottleneck layers: Although each layer only produces k output feature-maps, the number of inputs can be quite high, especially for further layers. Thus, a 1x1 convolution layer can be introduced as a bottleneck layer before each 3x3 convolution to improve the efficiency and speed of computations.

Sequential Layer: Sequential is a container of Modules that can be stacked together and run at the same time. Sequential layer is used to store feature vector returned by the ResNext model in a ordered way. So that it can be passed to the LSTM sequentially.

LSTM Layer: LSTM is used for sequence processing and spot the temporal change between the frames.2048-dimensional feature vectors is fitted as the input to the LSTM. We are using 1 LSTM layer with 2048 latent dimensions and 2048 hidden layers along with 0.4 chance of dropout, which is capable todo achieve our objective. LSTM is used to process the frames in a sequential manner so that the temporal analysis of the video can be made, by comparing the frame at ‘t’ second with the frame of ‘t-n’ seconds. Where n can be any number of frames before t.

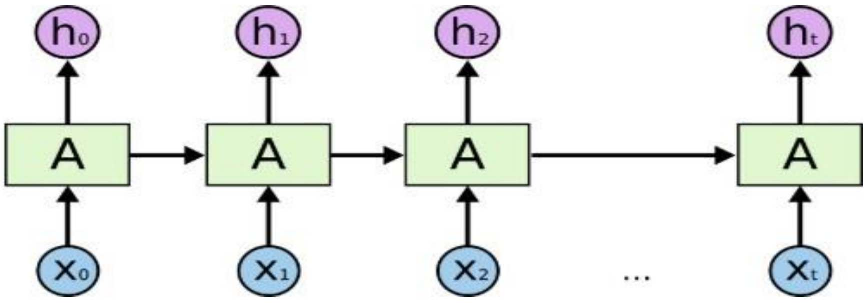


Figure 3.15: Overview of LSTM Architecture

Ref: https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e

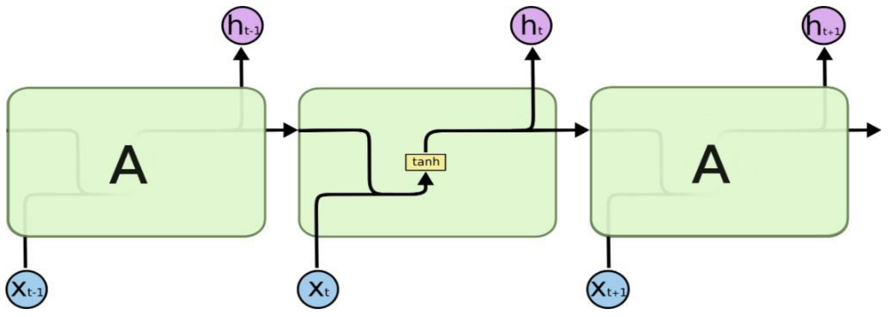


Figure 3.16: Internal LSTM Architecture

Ref: https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e

- **ReLU:** A Rectified Linear Unit is activation function that has output 0 if the input is less than 0, and raw output otherwise. That is, if the input is greater than 0, the output is equal to the input. The operation of ReLU is closer to the way our biological neurons work. ReLU is non-linear and has the advantageof not having any backpropagation errors unlike the sigmoid function, also for larger Neural Networks, the speed of building models based off on ReLU is very fast.

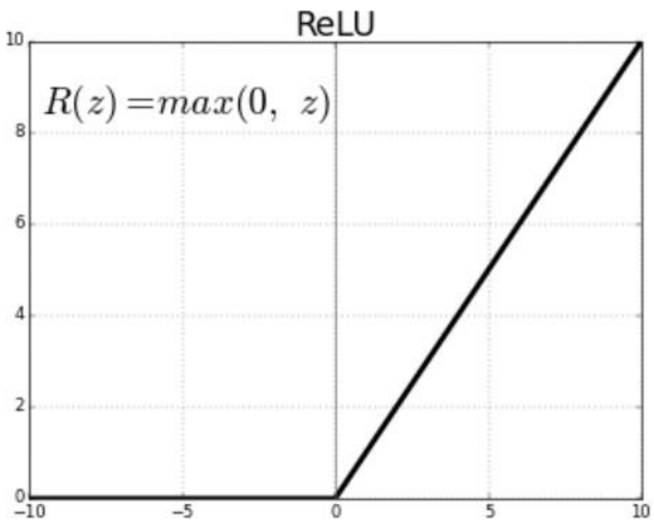


Figure 3.17: Relu Activation function

Ref: https://www.google.com/imgres?imgurl=https://scaler.com/topics/images/pytorch%20-%20relu.webp&tbnid=OlS47bUDqNafbM&vet=1&imgrefurl=https://www.scaler.com/topics/pytorch/deep%20-learning-with-%20pytorch/&docid=GVnRbyAYchqaNM&w=1700&h=1036&source=sh/x/im

- **Dropout Layer:** Dropout layer with the value of 0.4 is used to avoid over- fitting in the model and it can help a model generalize by randomly setting the output for a given neuron to 0. In setting the output to 0, the cost function becomes more sensitive to neighbouring neurons changing the way the weights will be updated during the process of backpropagation.

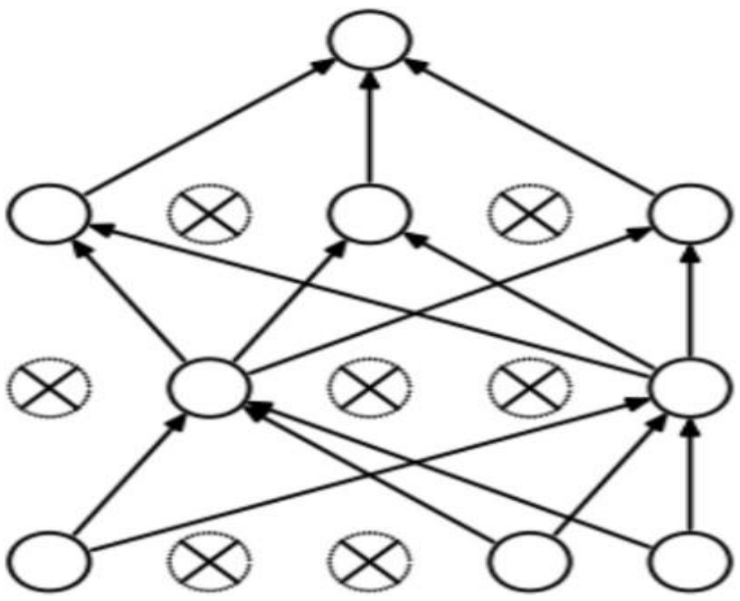


Figure 3.18: Dropout layer overview

Ref: https://www.google.com/imgres?imgurl=https://cdn.analyticsvidhya.com/wp-%20content/uploads/2018/04/Screen-Shot-2018-04-03-at-11.52.06-%20PM.png&tbnid=ROuk5PY0xlPQaM&vet=1&imgrefurl=https://www.analyticsvidhya.com%20/blog/2018/04/fundamentals-deep-learning-regularization-%20techniques/&docid=Fnt3qpwWYdCcFM&w=331&h=328&source=sh/x/im

- **Adaptive Average Pooling Layer:** It is used to reduce variance, reduce com- putation complexity and extract low level features from neighbourhood.2-dimensional Adaptive Average Pooling Layer is used in the model.

3.4.5 Model Training Details

- **Train Test Split:** The dataset is split into train and test dataset with a ratio of 80% train videos (296) and 20% (74) test videos.
- **Data Loader:** It is used to load the videos and their labels with a batch size of 4.
- **Training:** The training is done for 20 epochs with a learning rate of 1e-5 (0.00001), weight decay of 1e-3 (0.001) using the Adam optimizer.
- **Adam optimizer:** To enable the adaptive learning rate Adam optimizer with the model parameters is used.
- **Cross Entropy:** To calculate the loss function Cross Entropy approach is used because we are training a classification problem.

- **SoftMax Layer:** A SoftMax function is a type of squashing function. Squashing functions limit the output of the function into the range 0 to 1. This allows the output to be interpreted directly as a probability. Similarly, SoftMax functions are multi-class sigmoid, meaning they are used in determining probability of multiple classes at once. Since the outputs of a SoftMax function can be interpreted as a probability (i.e., They must sum to 1), a SoftMax layer is typically the final layer used in neural network functions. It is important to note that a SoftMax layer must have the same number of nodes as the output later.

  In our case SoftMax layer has two output nodes i.e REAL or FAKE, also Soft- max layer provide us the confidence(probability) of prediction.

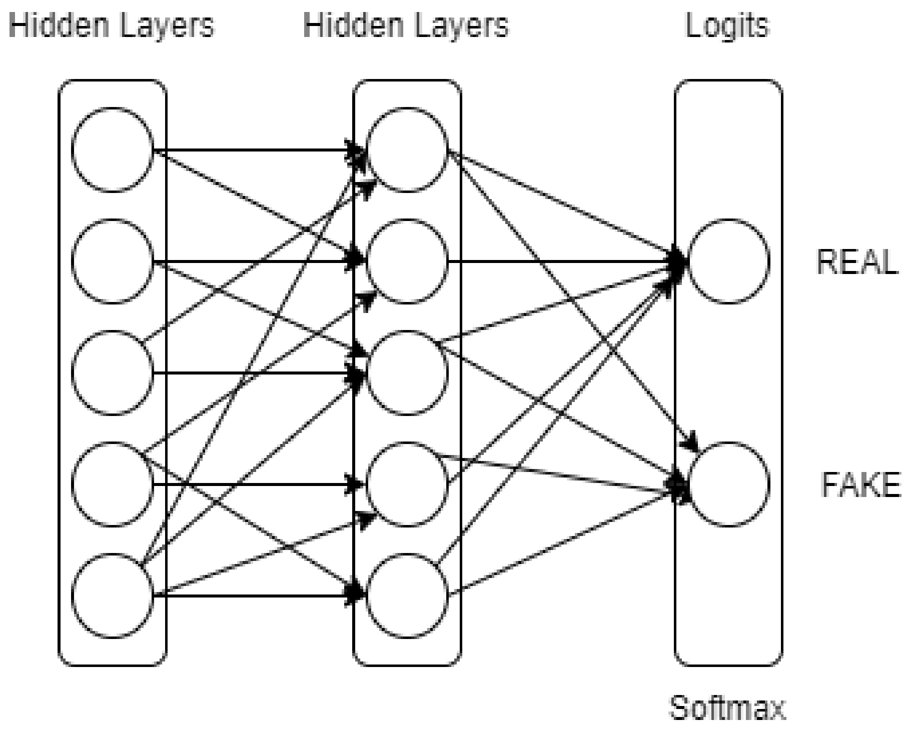


Figure 3.19: **SoftMax Layer**

- **Confusion Matrix:** A confusion matrix is a summary of prediction results ona classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your

  classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly thetypes of errors that are being made.

  Confusion matrix is used to evaluate our model and calculate the accuracy.

- **Export Model:** After the model is trained, we have exported the model. Sothat it can be used for prediction on real time data.

3.4.6 Model Prediction Details

- The model is loaded in the application
- The new video for prediction is pre-processed and passed to the loaded model for prediction
- The trained model performs the prediction and returns if the video is a real or fake along with the confidence of the prediction.

3.5 Testing

3.5.1 Type of Testing Used

Functional Testing

1. Unit Testing
2. Integration Testing
3. System Testing

Non-functional Testing

1. Performance Testing
2. Load Testing
3. Compatibility Testing

3.5.2 Test Cases and Test Results

Test Cases

Case id	Test Case Description	Expected Result	Actual Result	Status
1	Videos with many faces	Fake / Real	Fake	Pass
2	Deepfake video	Fake	Fake	Pass
3	Upload a Real video	Real	Real	Pass
4	Upload a face croppedreal video	Real	Real	Pass
5	Upload a face croppedfake video	Fake	Fake	Pass

Table 3.1: Test Case Report

3.6 Maintenance

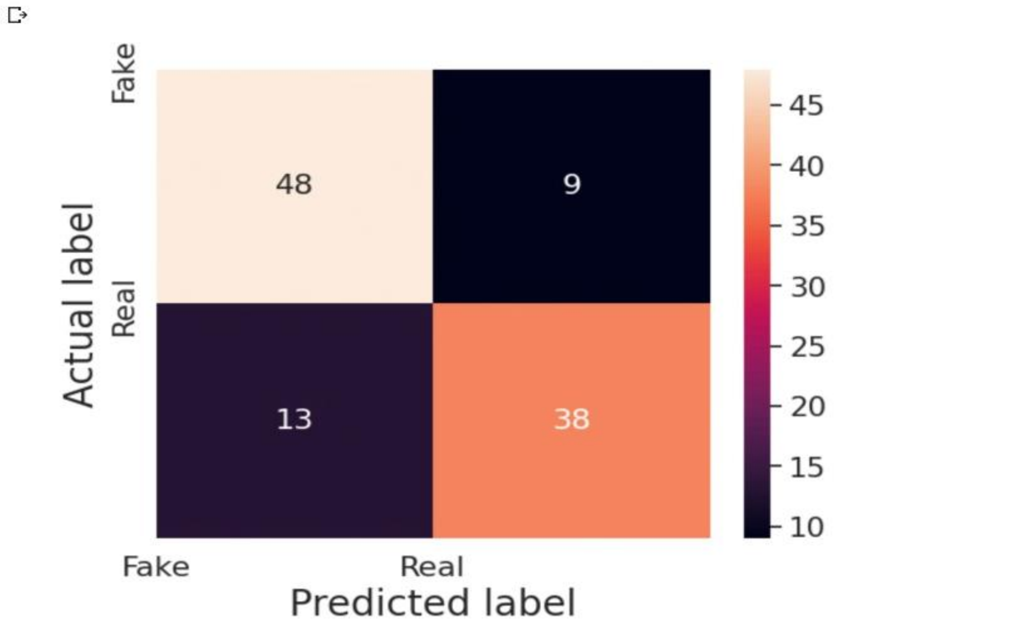
Following are the steps to be followed for the updating the code to the latest versionof the application.

1. Stop the production server using Ctrl + C.
2. git pull
   Note: As its a private repository only, authorized users will be able see the codeand do the process of deployment
3. pip install -r requirement.txt
4. python manage.py migrate
5. Copy all the trained models into model's folder (optional: Do this if any new models are added).
6. python manage.py run server 0.0.0.0:8000

4. Project Snapshots:

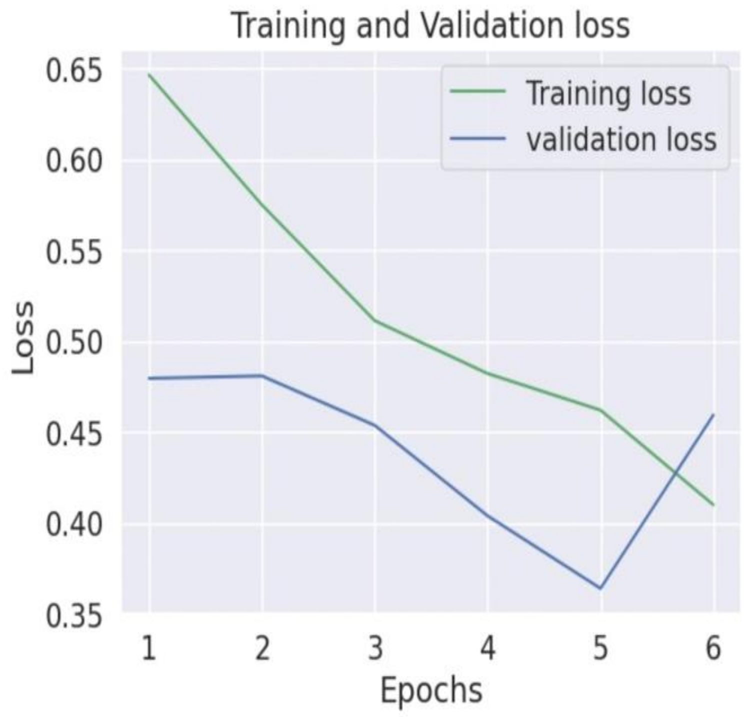
We have Implemented various other models for feature extraction and compared different models to come to some point that which one is giving the best result in the given problem scenario, there accuracy is analyzed and graphs are studied thoroughly. Based on that we found that Densenet121 is giving better results as compared to VGG16 and ResNext50.

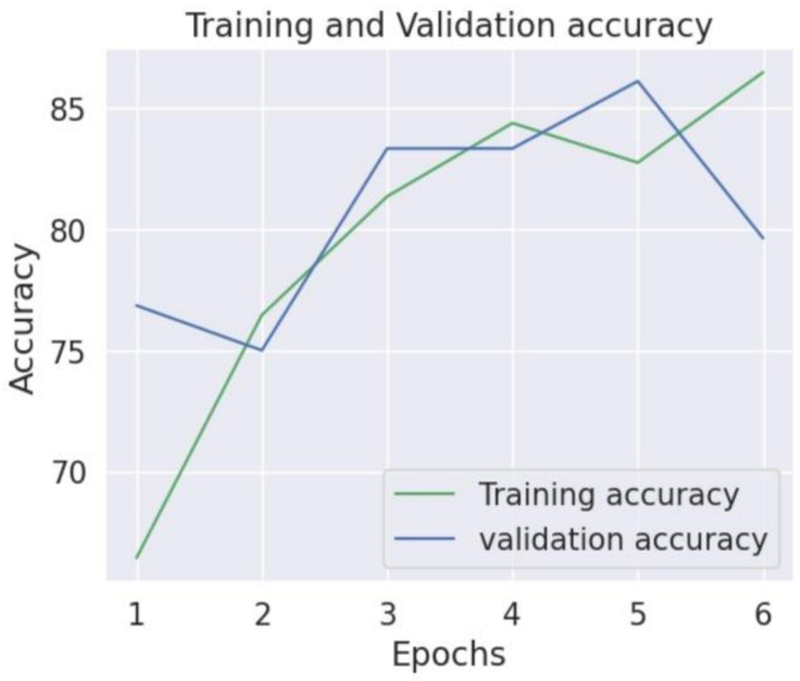
4.1 ResNext:



```
Calculated Accuracy 79.62962962962963
```

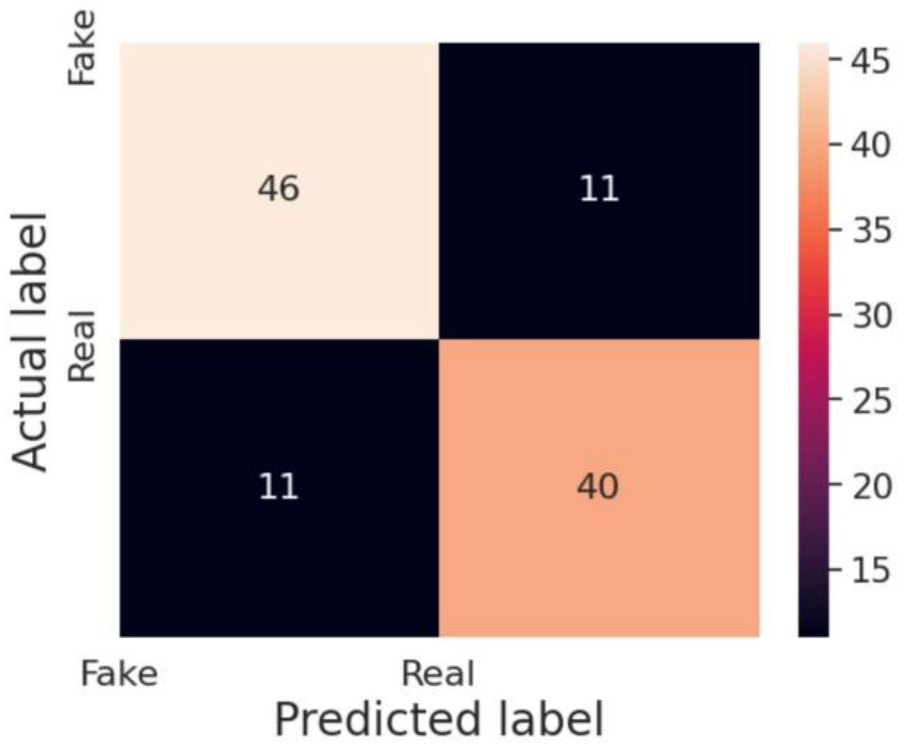
```
Accuracy 79.62962962962963
6
```



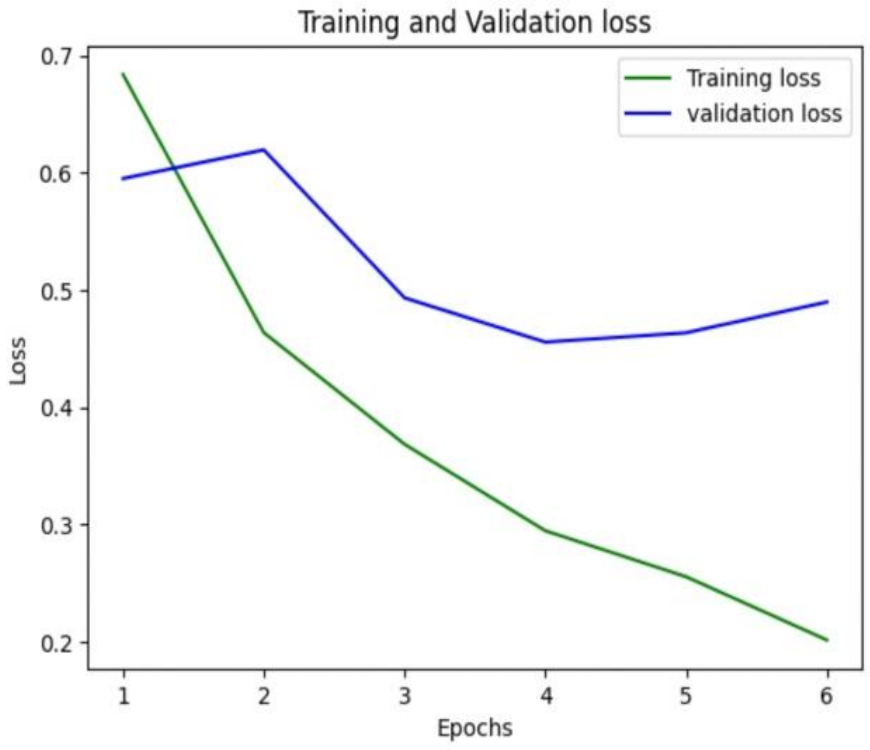


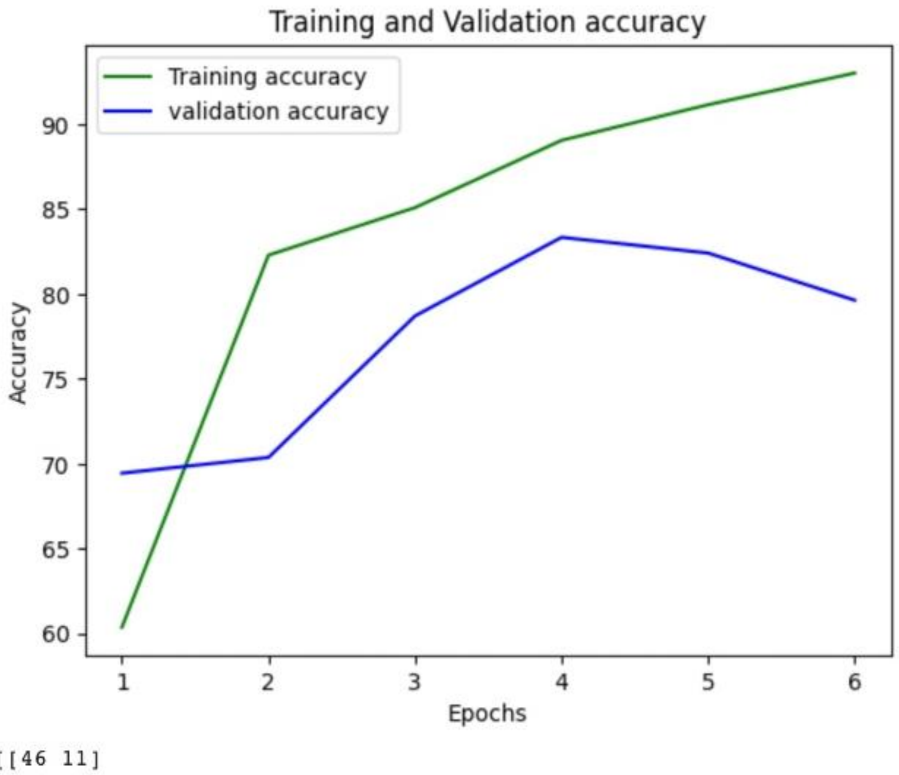
```
[[48  9]
 [13 38]]
True positive =  48
False positive =  9
False negative =  13
True negative =  38
```

4.2 VGG 16:



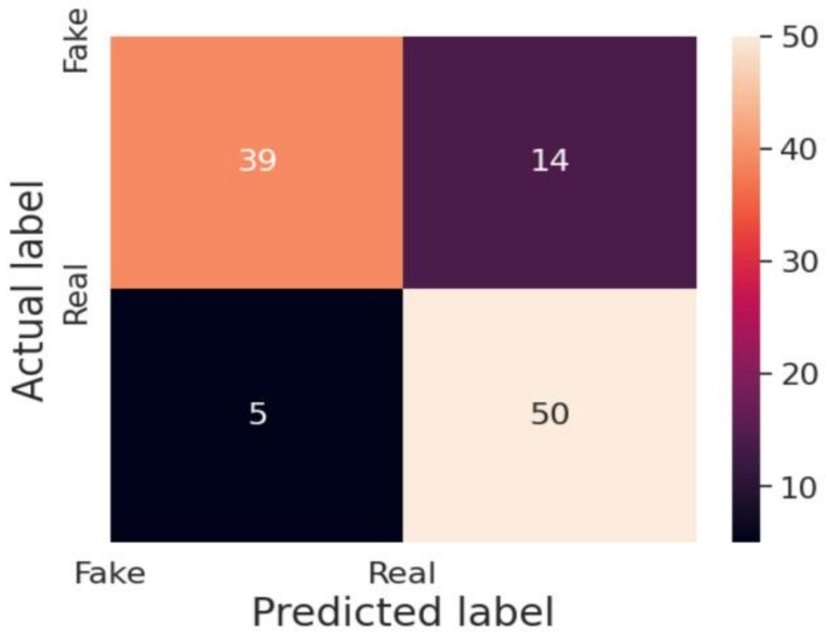
Calculated Accuracy 79.7286



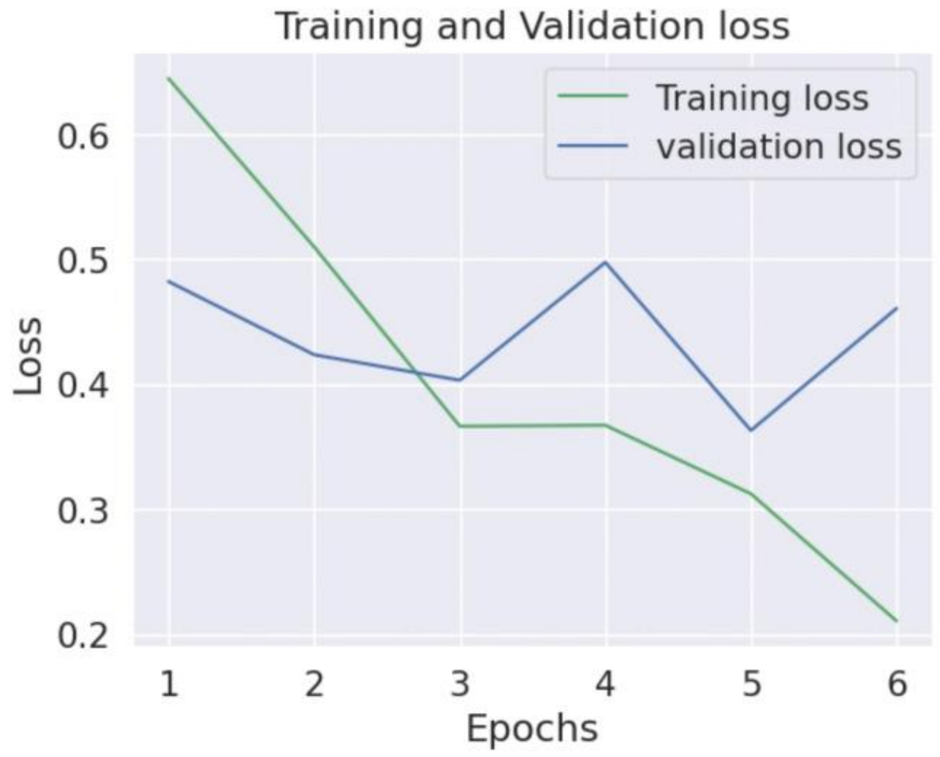


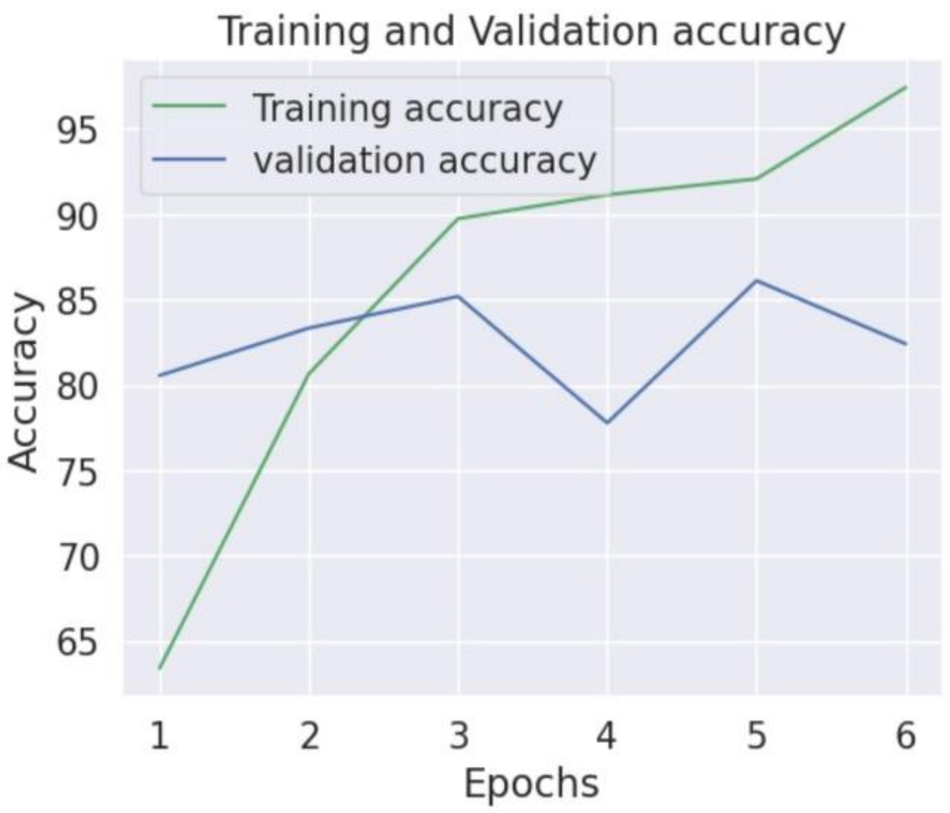
```
[[46 11]
 [11 40]]
True positive =  46
False positive =  11
False negative =  11
```

4.3 DenseNet



```
Calculated Accuracy 82.4074074074074
```



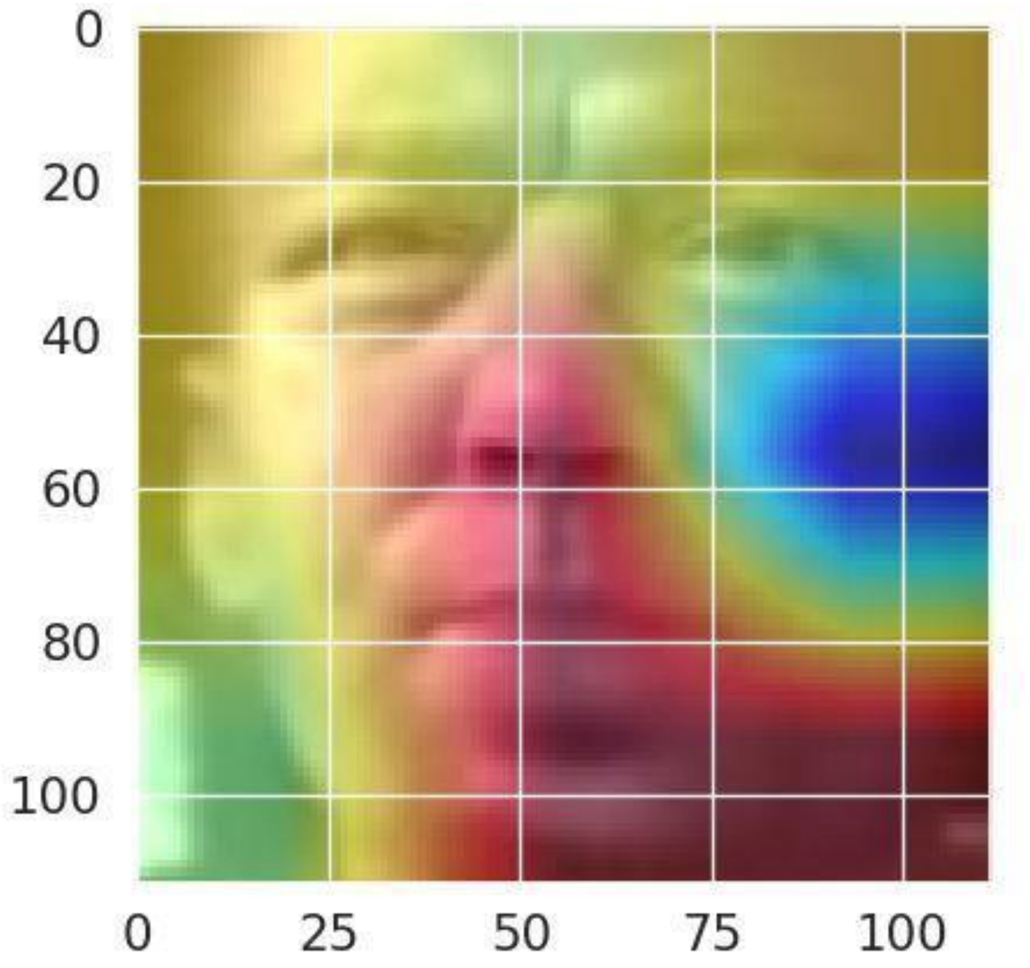


```
[[39 14]
 [ 5 50]]
```

Models	Accuracy
VGG 16	**79.7286**
ResNext	**79.6296**
DenseNet	**82.4046**

Table: 4.1 Summary of Results

Prediction:



FAKE

5. Future Scope

There is always a scope for enhancements in any developed system, especially when the project build using latest trending technology and has a good scope in future.

- Web based platform can be upscaled to a browser plugin for ease of access tothe user.
- Currently only Face Deep Fakes are being detected by the algorithm, but thealgorithm can be enhanced in detecting full body deep fakes.
- Increasing the dataset for increasing accuracy

6. Bibliography:

[1] Nataraj, L., et al. (2019) Detecting GAN Generated Fake Images Using Co-Occurrence Matrices. Electronic Imaging, 2019, 532-1-532-7. https://doi.org/10.2352/ISSN.2470-1173.2019.5.MWSF-532

[2] Wang, S.-Y., Wang, O., Zhang, R., Owens, A. and Efros, A.A. (2020) CNN-Generated Images Are Surprisingly Easy to Spot... for Now. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, 13-19 June 2020, 8695-8704. https://doi.org/10.1109/CVPR42600.2020.00872

[3] Hsu, C.-C., Lee, C.-Y. and Zhuang, Y.-X. (2018) Learning to Detect Fake Face Images in the Wild. 2018 IEEE International Symposium on Computer, Consumer and Control (IS3C), Taichung, 6-8 December 2018, 388-391. https://doi.org/10.1109/IS3C.2018.00104

[4] Vaccari, C. and Chadwick, A. (2020) Deepfakes and Disinformation: Exploring the Impact of Synthetic Political Video on Deception, Uncertainty, and Trust in News. Social Media + Society, 6, 1-13. https://doi.org/10.1177/2056305120903408

[5] Mirza, M. and Osindero, S. (2014) Conditional Generative Adversarial Nets.

[6] Kwok, A.O. and Koh, S.G. (2020) Deepfake: A Social Construction of Technology Perspective. Current Issues in Tourism, 1-5. https://doi.org/10.1080/13683500.2020.1738357

[7] Westerlund, M. (2019) The Emergence of Deepfake Technology: A Review. Technology Innovation Management Review, 9, 40-53. https://doi.org/10.22215/timreview/1282

[8] Güera, D. and Delp, E.J. (2018) Deepfake Video Detection Using Recurrent Neural Networks. 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Auckland, 27-30 November 2018, 1-6. https://doi.org/10.1109/AVSS.2018.8639163

[9] Li, Y. and Lyu, S. (2018) Exposing Deepfake Videos by Detecting Face Warping Artifacts.

[10] Yang, X., Li, Y. and Lyu, S. (2019) Exposing Deep Fakes Using Inconsistent Head Poses. 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, 12-17 May 2019, 8261-8265. https://doi.org/10.1109/ICASSP.2019.8683164

[11] Marra, F., Gragnaniello, D., Cozzolino, D. and Verdoliva, L. (2018) Detection of Gan-Generated Fake Images over Social Networks. 2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), Miami, 10-12 April 2018, 384-389. https://doi.org/10.1109/MIPR.2018.00084

[12] Grekousis, G. (2019) Artificial Neural Networks and Deep Learning in Urban Geography: A Systematic Review and Meta-Analysis. Computers, Environment and Urban Systems, 74, 244-256. https://doi.org/10.1016/j.compenvurbsys.2018.10.008

[13] Hopfield, J.J. (1982) Neural Networks and Physical Systems with Emergent Collective Computational Abilities. Proceedings of the National Academy of Sciences, 79, 2554-2558. https://doi.org/10.1073/pnas.79.8.2554

[14] Pouyanfar, S., et al. (2018) A Survey on Deep Learning: Algorithms, Techniques, and Applications. ACM Computing Surveys (CSUR), 51, 1-36. https://doi.org/10.1145/3234150

[15] Goodfellow, I., Bengio, Y., Courville, A. and Bengio, Y. (2016) Deep Learning (No. 2). MIT Press, Cambridge.

[16] Elman, J.L. (1990) Finding Structure in Time. Cognitive Science, 14, 179-211. https://doi.org/10.1207/s15516709cog1402_1

[17] Bengio, Y., Simard, P. and Frasconi, P. (1994) Learning Long-Term Dependencies with Gradient Descent Is Difficult. IEEE Transactions on Neural Networks, 5, 157-166. https://doi.org/10.1109/72.279181

[18] Hochreiter, S. and Schmidhuber, J. (1997) Long Short-Term Memory. Neural Computation, 9, 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

[19] Schuster, M. and Paliwal, K.K. (1997) Bidirectional Recurrent Neural Networks. IEEE Transactions on Signal Processing, 45, 2673-2681. https://doi.org/10.1109/78.650093

[20] Faceswap: Deepfakes Software for All. https://github.com/deepfakes/faceswap

[21] FakeApp 2.2.0. https://www.malavida.com/en/soft/fakeapp

[22] Keras-VGGFace: VGGFace Implementation with Keras Framework. https://github.com/rcmalli/keras-vggface

[23] CycleGAN. https://junyanz.github.io/CycleGAN/

[24] Tariq, S., Lee, S., Kim, H., Shin, Y. and Woo, S.S. (2018) Detecting Both Machine and Human Created Fake Face Images in the Wild. Proceedings of the 2nd International Workshop on Multimedia Privacy and Security, Toronto, 15 October 2018, 81-87. https://doi.org/10.1145/3267357.3267367

[25] Li, H., Li, B., Tan, S. and Huang, J. (2018) Detection of Deep Network Generated Images Using Disparities in Color Components.

[26] Do, N.-T., Na, I.-S. and Kim, S.-H. (2018) Forensics Face Detection from GANS Using Convolutional Neural Network. ISITC.