ABSTRACT

The increased computational capacity has rendered deep learning algorithms incredibly potent, to the point where generating highly realistic human-synthesized videos, commonly known as deep fakes, has become remarkably effortless. These highly realistic face-swapped deepfakes can be deployed in various scenarios, such as causing political turmoil, fabricating terrorism events, engaging in revenge porn, or blackmailing individuals. This work describes a new deep learning-based method that can effectively distinguish AI-generated fake videos from real videos. Our method is capable of automatically detecting the replacement and reenactment of deep fakes. The research tries to use Artificial Intelligence (AI) to fight Artificial Intelligence (AI). Our system employs a Res-Next Convolutional Neural Network to extract features at the frame level. These extracted features are then utilized to train a Recurrent Neural Network (RNN) based on Long Short-Term Memory (LSTM). The purpose of this training is to classify videos, determining whether they have undergone any form of manipulation, thereby discerning between deep fake videos and genuine ones. To emulate the real time scenarios and make the model perform better on real time data, the system evaluates our method on large amount of balanced and mixed dataset prepared by mixing the various available dataset like Face-Forensic++, Deepfake detection challenge, and Celeb-DF. It also shows how our system can achieve competitive results using a very simple and robust approach.

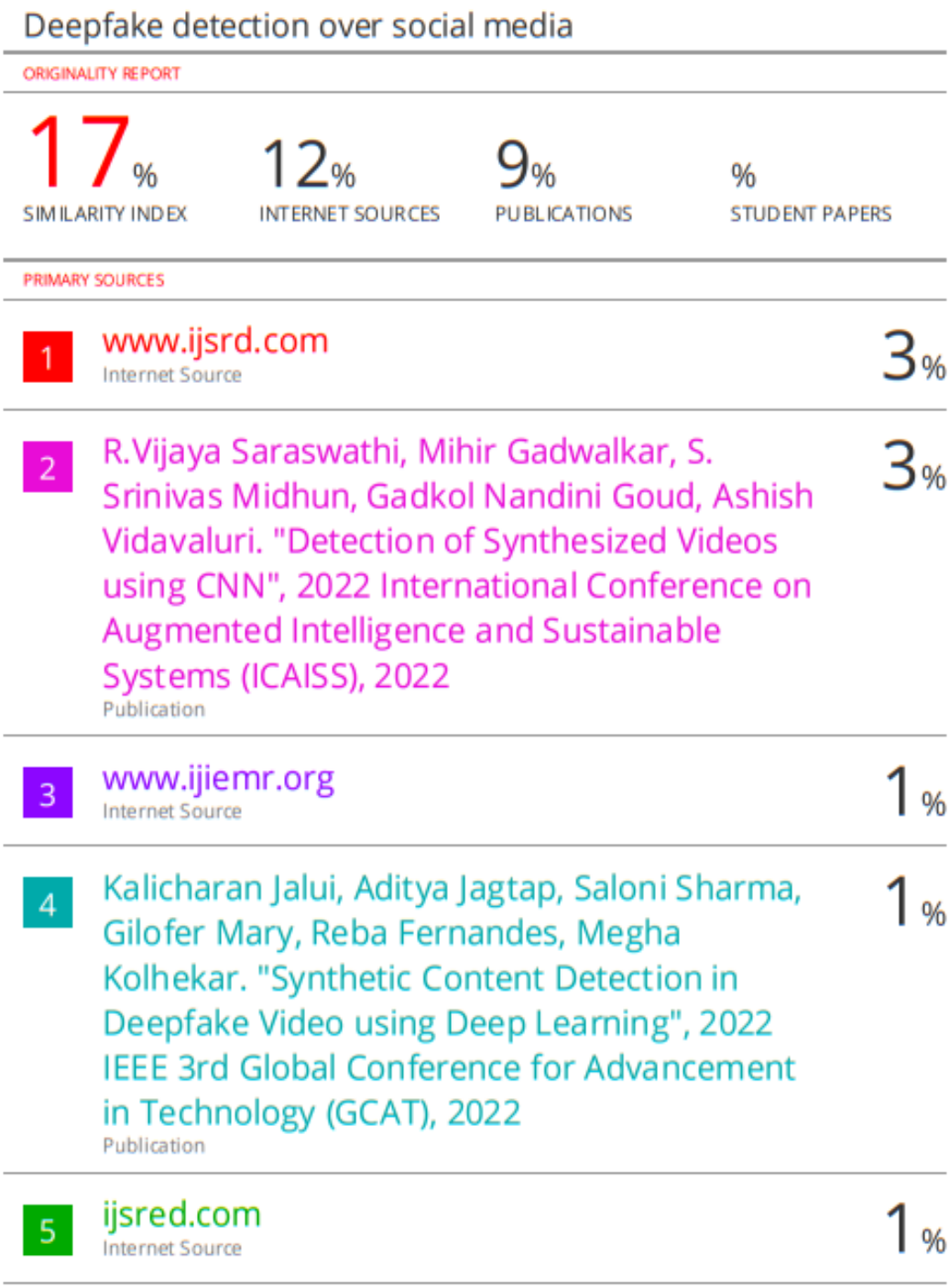
Keywords: *Res-Next Convolution neural network, Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Computer vision.*

PLAGIARISM REPORT

Deepfake detection over social media

by Ramavatar Yadav

Submission date: 29-May-2023 11:43AM (UTC+0500)
Submission ID: 2104370392
File name: 206121027_thesis_check_plag.pdf (1.19M)
Word count: 5912
Character count: 30768



Ramavatar Yadav
Student

Dr. M. Sridevi
Project Guide

TABLE OF CONTENTS

Title **Page No**

Contents

LIST OF FIGURES

Fig No **Title** **Page No**

LIST OF TABLES

Table No **Title** **Page No**

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

In the world of ever growing Social media platforms, Deepfakes are considered as the major threat of the AI. These highly realistic face-swapped deepfakes can be deployed in various scenarios, such as causing political turmoil, fabricating terrorism events, engaging in revenge porn, or blackmailing individuals. Some of the examples are Brad Pitt, Angelina Jolie nude videos. It becomes very important to spot the difference between the deepfake and pristine video. The system uses AI to fight AI Deepfakes are created using tools like FaceApp and Face Swap, which use pre-trained neural networks like GAN or Auto encoders for these deepfakes creation. Our method uses a LSTM based artificial neural network to process the sequential temporal analysis of the video frames and pre-trained Res-Next CNN to extract the frame level features. ResNext The frame-level features are extracted by a Convolutional Neural Network (CNN), which are then utilized to train an artificial Recurrent Neural Network (RNN) based on Long Short-Term Memory (LSTM). This training aims to classify videos as either Deepfake or real. In order to enhance the model's performance on real-time data and emulate real-world scenarios, our method was trained using a substantial amount of balanced data, including a combination of various available datasets such as Face Forensic, Deepfake Detection Challenge, and Celeb-DF. Additionally, to provide a user-friendly experience, a front-end application is developed where users can easily upload their videos. The video will be processed by the model and the output will be rendered back to the user with the classification of the video as deepfake or real and confidence of the model.

1.2 MOTIVATION

Advancements in mobile camera technology and the widespread adoption of social media platforms have made it incredibly easy to create and share digital videos. With the emergence of deep learning, previously unimaginable technologies have become a reality. Modern generative models, for instance, have the remarkable ability to produce highly realistic images, speech, music, and video. These models have found diverse applications, such as enhancing accessibility through text-to-speech technology and aiding in the generation of training data for medical imaging. Overall, the convergence of mobile cameras, social media, and deep learning has revolutionized the creation and dissemination of digital videos, enabling a wide range of

innovative applications in various fields.

The prevalence of fake videos circulating on social media platforms is on the rise, resulting in issues such as spamming and the dissemination of false information. Consider the alarming scenario of a deepfake video depicting our prime minister announcing an unjust war against neighboring nations or a fabricated video showing a renowned celebrity verbally abusing their fans. The consequences of such deepfakes would be disastrous, as they have the potential to threaten and deceive the public.

To address this issue, the detection of Deepfake videos becomes paramount. Hence, a novel deep learning-based approach that proficiently discriminates between AI-generated fake videos (Deep Fake Videos) and genuine ones is presented. The development of technology capable of detecting fakes is of utmost significance to identify and prevent the dissemination of deep fakes across the internet.

The motivation for deepfake video detection lies in protecting individuals' privacy, preserving trust in media, combating misinformation, safeguarding democracy, enhancing security efforts, and promoting responsible and ethical AI practices.

1.3 SCOPE

While numerous tools exist for generating deep fakes, the availability of tools for deep fake detection is scarce. Our approach to detecting deep fakes will serve as a significant contribution in preventing the widespread dissemination of deep fakes across the World Wide Web. This Proposed system safeguard the integrity of online information, protect user trust, and mitigate the potential harms caused by the dissemination of deepfake content within the dynamic and influential realm of social media.

1.4 OBJECTIVES

The main objective of this proposed system is

- To discover the distorted truth of the deep fakes.
- To reduce the Abuses' and misleading of the common people on the world wide web.
- To distinguish and classify the video as deepfake or pristine.
- To provide an easy to use system for uploading the video and distinguish whether that video is real or fake.

CHAPTER 2
LITERATURE REVIEW

In Meta Deepfake Detection (MDD) [1], a meta-learning-based method for generalized deepfake detection. MDD utilizes meta-weight learning to transfer information between different domains, enhancing model generalization. Pair-attention loss and average-center alignment loss are introduced to improve detection capabilities. Evaluation on popular deepfake datasets confirms the effectiveness of our approach.

The proliferation of AI and deep learning tools has led to the emergence of Deepfake (DF) [2] videos, posing a significant challenge in determining their authenticity. Developing algorithms capable of detecting DF is a complex endeavor. In this research, a framework is used that leverages Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) [12]. CNN extracts frame-level features, subsequently employed by RNN to classify videos based on temporal inconsistencies. Through comparisons with established datasets, our straightforward architecture showcases the effectiveness and accuracy of our proposed method in identifying DF videos.

In this introduce a unique method for detecting Deepfakes by extracting a Deepfake fingerprint using the Expectation-Maximization algorithm [3]. This fingerprint captures Convolutional Traces (CT) left by GANs [4] during image generation and exhibits excellent discriminatory capabilities. With an impressive overall accuracy of over 86%, our CT-based approach surpasses existing techniques, demonstrating robustness against different attacks and independence from image semantics. Moreover, real-case tests conducted on Deepfakes generated by FACEAPP achieve a remarkable 84% accuracy in detecting fake images, further validating the effectiveness of our proposed technique.

The availability of extensive facial databases and advancements in deep learning, particularly Generative Adversarial Networks (GANs) [4], have resulted in the creation of highly realistic fake facial content. This raises concerns about potential misuse and has prompted research on manipulation detection methods. In this study, we introduce a novel approach using autoencoders to remove GAN "fingerprints" from synthetic fake images, while maintaining visual quality. We conduct a comprehensive analysis of

facial manipulation detection techniques and evaluate the challenging task of detecting facial manipulation in unconstrained scenarios. Additionally, we present iFakeFaceDB, a new public database generated using our proposed GANprintR approach. Our empirical evaluation highlights the need for robust detection systems against unseen conditions and spoof techniques.

In Face Warping Artifacts [9], there is a special type of computer program called a Convolutional Neural Network to detect problems in fake faces. They compared the fake face to the real face around it and looked for differences. This method found two types of problems in fake faces. The basis of their method lies in the observation that existing deepfake algorithms can solely generate images with constrained resolutions. Subsequently, these images must undergo further transformations to align with the faces to be replaced within the source video.

In this method the deepfake technology can only create images that are not very detailed. To make the deepfake match the person in the original video, the image must be changed in some way. Their method has not considered the temporal analysis of the frames.

Detection by Eye Blinking [10] describes a new method for detecting the deep- fakes by the eye blinking as a crucial parameter leading to classification of the videos as deepfake or pristine. The Long-term Recurrent Convolution Network (LRCN) was used for temporal analysis of the cropped frames of eye blinking. As today the deepfake generation algorithms have become so powerful that lack of eye blinking cannot be the only clue for detection of the deepfakes. There must be certain other parameters considered for the detection of deep- fakes like teeth enchantment, wrinkles on faces, wrong placement of eyebrows etc.

The utilization of capsule networks [11] in the detection of forged images and videos employs a methodology that leverages a capsule network to identify manipulated and forged content in various scenarios, including replay attack detection and the detection of computer-generated videos.

Their approach incorporates random noise during the training phase, which may not be an ideal choice. Despite this, their model exhibited favorable performance on their dataset. However, it may encounter difficulties when applied to real-time data due to the presence of noise during training. In contrast, our proposed method is designed to

be trained on noiseless, real-time datasets, aiming to enhance its effectiveness in real-world scenarios.

Recurrent Neural Network [12] (RNN) for deepfake detection used the approach of using RNN for sequential processing of the frames along with ImageNet pre-trained model. Their process used the dataset consisting of just 600 videos.

Their dataset consists of a small number of videos and the same type of videos, which may not perform very well on real time data. The model will be trained on large number of Realtime data.

The Synthetic Portrait Videos using Biological Signals [14] approach described involves extracting biological signals from facial regions in pairs of pristine and deepfake portrait videos. Transformations are applied to calculate spatial coherence and temporal consistency, capturing signal characteristics in feature vectors and photoplethysmography (PPG) maps. A probabilistic Support Vector Machine (SVM) and Convolutional Neural Network (CNN) [12] are then trained using these signals. The average authenticity probabilities are used for classification, determining whether the video is a deepfake or a pristine recording.

Fake Catcher demonstrates high accuracy in detecting fake content, regardless of the generator, content, resolution, or quality of the video. However, the lack of a discriminator in their findings hinders the preservation of biological signals.

CHAPTER 3
PROPOSED WORK AND IMPLEMENTATION

A robust deepfake detection system can be developed by combining advanced machine learning algorithms with a comprehensive dataset of both real and deepfake videos, enabling the system to accurately identify and differentiate between genuine and manipulated content.

3.1 PROPOSED SOLUTION

An unsupervised deep learning model for detection of deepfake image/video over the social media is proposed. The video frames are being converted into frames and each frame is being given to preprocessing phase. All the unrequired and noise is removed from videos. After splitting the video into frames, the face is detected in each of the frames and the frame is cropped along the face. In the preprocessing phase the face is detected and cropped.

The feature extraction is being done, in feature extraction specific features of images like shape, size, color is being extracted. Later the cropped frame is again converted to a new video by combining each frame of the video. After the preprocessing phase the features are sent to the classifier. The pre-trained model of ResNext is used for feature extraction. ResNext is Residual CNN network optimized for high performance on deeper neural networks. For the experimental purpose there resnext50_32x4d model is used having 50 layers and 32 x 4 dimensions. This facilitates the sequential passing of the feature vector to the LSTM, enabling temporal analysis of the data.

The 2048-dimensional feature vectors are utilized as input to the LSTM. Our model includes one LSTM layer with 2048 latent dimensions and 2048 hidden layers. Additionally, a dropout rate of 0.4 is applied to enhance the model's performance. By sequentially analyzing the frames, LSTM enables us to assess temporal variations in the video. This involves comparing the frame at time 't' with the frame 't-n', where 'n' can represent any number of frames preceding 't'. For sequential processing of the frames a Sequential Layer is used. The batch size of 4 is used to perform batch training. A SoftMax layer is used to get the confidence of the model during prediction.

3.2 BLOCK DIAGRAM

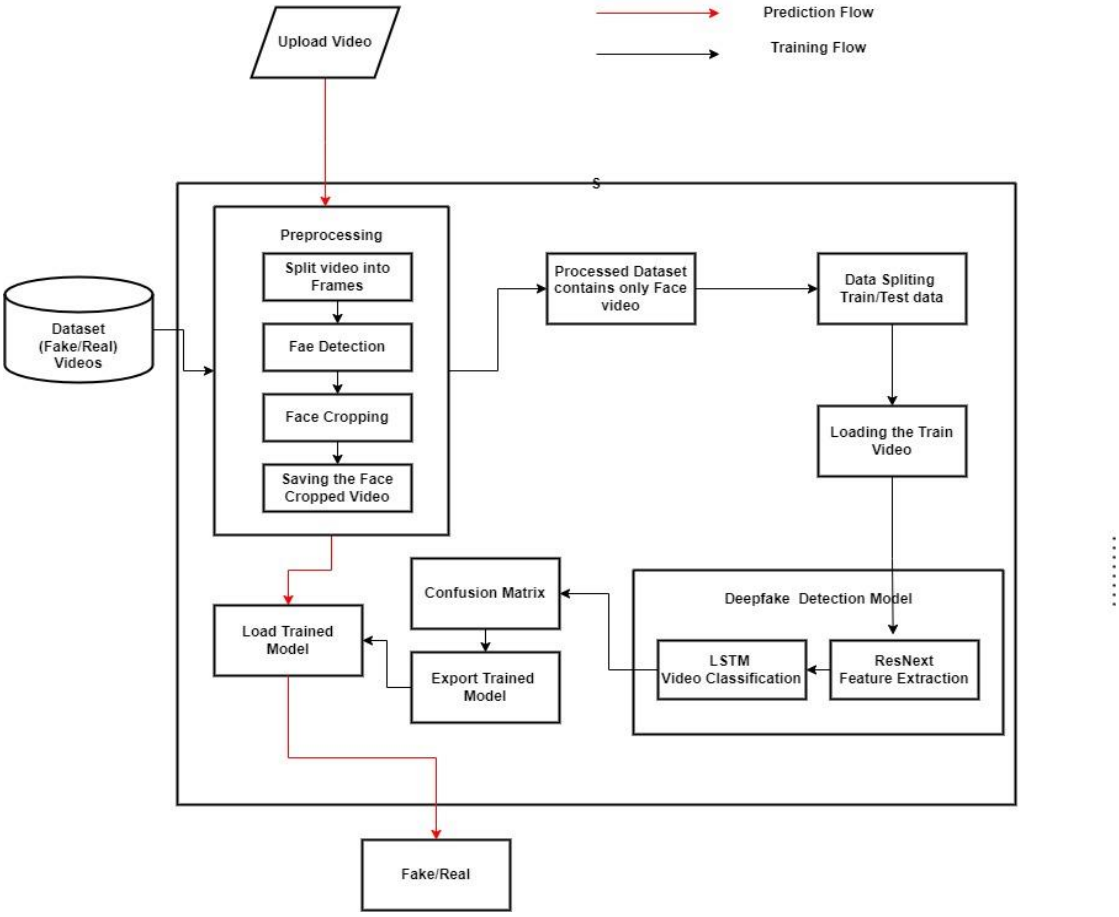


Fig. 3.1 System Architecture

In our system, a deepfake detection model is trained using PyTorch. To prevent bias in the model, it is ensured that an equal number of real and fake videos were used for training. The model's architecture is depicted in the Figure 3.1. During the development phase, a dataset is chosen and performed preprocessing on it. Specifically, a new processed dataset is created that exclusively consists of videos with cropped faces.

• Creating deepfake videos

To effectively detect deepfake videos, it is crucial to understand the underlying creation process. Most tools, such as GANs and autoencoders, utilize a source image and target video to generate deepfakes. These tools divide the video into frames, identify faces, and replace the source face with the target face on each frame. The modified frames are then combined using various pretrained models, which enhance the video's quality by eliminating residual traces left by the deepfake creation process. Although these deepfakes can appear highly realistic to the human eye, subtle traces and artifacts

often remain undetectable. The objective of this paper is to identify these imperceptible traces and distinguishable artifacts, enabling the classification of videos as either deepfakes or real based on these indicators.

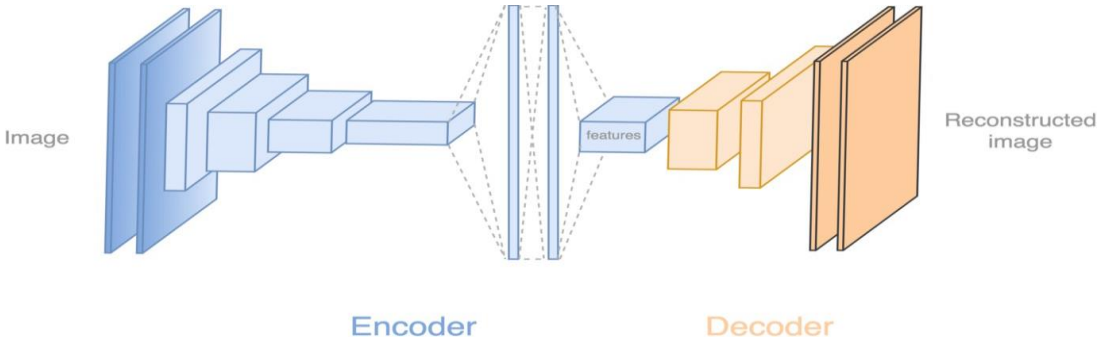


Fig. 3.2 Deepfake generation

Fig. 3.3 Face swapped deepfake generation

3.3 ARCHITECTURE DESIGN

A typical deepfake detection architecture consists of a pre-processing module for video input, followed by a feature extraction module that analyzes visual. These features are then fed into a deep learning model, such as a convolutional neural network (CNN) or a recurrent neural network (RNN), which classifies the input as either genuine or deepfake based on learned patterns and anomalies.

3.3.1 DATASET GATHERING

For making the model efficient for real time prediction. The data from different available datasets like FaceForensic++ (FF), Deepfake detection challenge (DFDC),

and Celeb-DF is gathered. Further the dataset with the collected datasets is mixed and created our own new data set, to accurate and real time detection on different kinds of videos. To avoid the training bias of the model, 50% Real and 50% fake videos are considered. Deep fake detection challenge (DFDC) dataset consists of certain audio alerted video, as audio deepfake are out of scope for this paper. The system preprocessed the DFDC dataset and removed the audio altered videos from the dataset by running a python script. After preprocessing the DFDC dataset, 1500 Real and 1500 Fake videos from the DFDC dataset are taken. 1000 Real and 1000 Fake videos from the FaceForensic++(FF) dataset and 500 Real and 500 Fake videos from the CelebDF dataset. Which makes our total dataset consisting of 3000 Real, 3000 fake videos and 6000 videos in total.

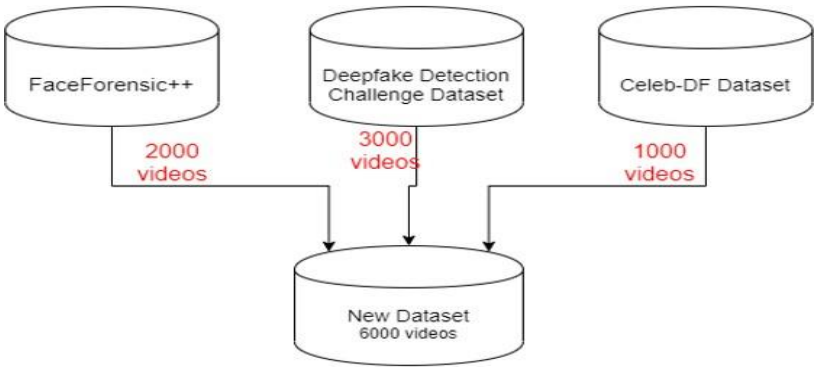


Fig. 3.4 Distribution of the datasets Dataset

3.3.2 PRE-PROCESSING

In this step, the videos are preprocessed and all the unrequired and noise is removed from videos. Only the required portion of the video i.e., face is detected and cropped. The first steps in the preprocessing of the video are to split the video into frames. After splitting the video into frames, the face is detected in each of the frames and the frame is cropped along the face. Later the cropped frame is again converted to a new video by combining each frame of the video.

The process is followed for each video which leads to creation of processed dataset containing face only videos. The frame that does not contain the face is ignored while preprocessing. To maintain the uniformity of number of frames, a threshold value is selected based on the mean of total frames count of each video. Another reason for selecting a threshold value is limited computation power.

As a video of 10 second at 30 frames per second(fps) will have total 300 frames and it is computationally very difficult to process the 300 frames at a single time in the experimental environment. So, based on our Graphic Processing Unit (GPU) computational power in an experimental environment 150 frames are selected as the threshold value. While saving the frames to the new dataset only the first 150 frames of the video are saved to the new video. In our demonstration of the effective utilization of Long Short-Term Memory (LSTM), the sequential arrangement of frames is specifically focused. Rather than selecting frames randomly, the system organizes them in a consecutive manner, specifically considering the initial 150 frames. The resulting video is saved at a frame rate of 30 frames per second (fps) and a resolution of 112 x 112 pixels.

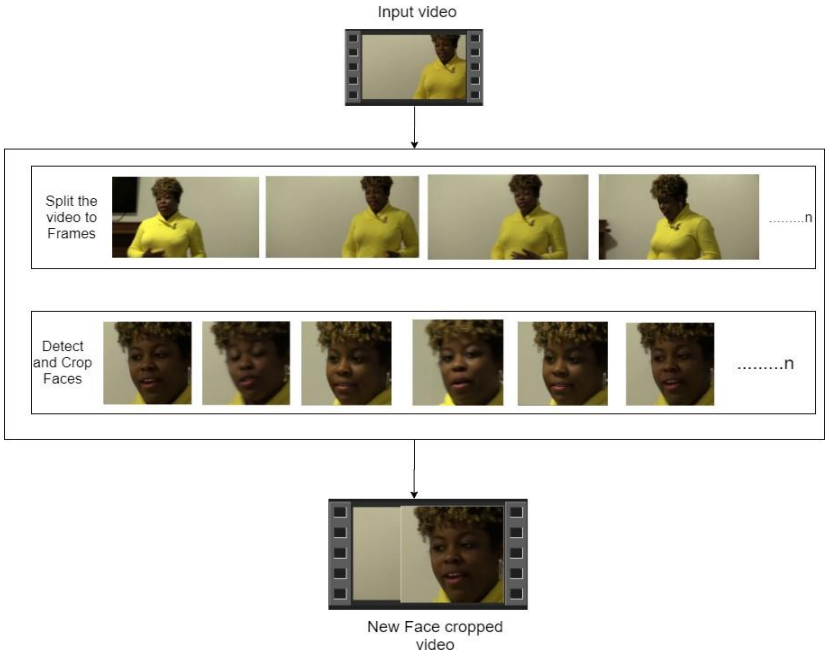


Fig.3.5 Pre-processing of video

3.3.3 DATASET SPLIT

The entire collection of data is divided into two parts: train dataset and test dataset with a ratio of 70% train videos (4,200) and 30% (1,800) test videos. The train and test split are a balanced split i.e. 50% of the real and 50% of fake videos in each split.

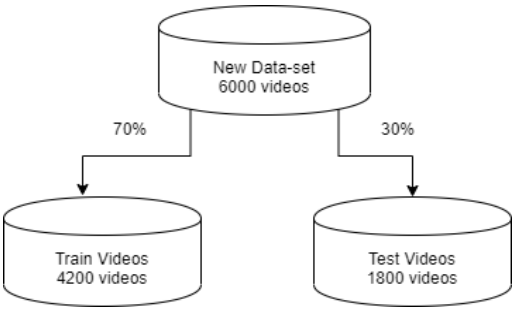


Fig. 3.6 Train test split

3.3.4 MODEL ARCHITECTURE

Our model is a combination of CNN and RNN. In this proposed system there used a pretrained ResNext CNN model to extract features from individual frames of the videos. These extracted features were then used to train a LSTM network, which aimed to classify the videos as either deepfake or authentic (pristine). By analyzing the distinctive features captured by the model, the system could determine whether a video was a deepfake or an original recording. Using the Data Loader on training split of videos the labels of the videos are loaded and fitted into the model for training.

3.3.4.1 RESNEXT50 MODEL

Instead of writing the code from scratch, the pre-trained model of ResNext for feature extraction was used. ResNext is Residual CNN network optimized for high performance on deeper neural networks. For the experimental purpose resnext50_32x4d model is used. A ResNext of 50 layers and 32 x 4 dimensions is used.

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

Fig. 3.7 ResNext Architecture

Following, the network is fine-tuned by incorporating additional necessary layers and carefully selecting an appropriate learning rate to ensure the smooth convergence of the model's gradient descent. Specifically, the 2048-dimensional feature vectors obtained from the last pooling layers of ResNext will be utilized as the input for the sequential LSTM.

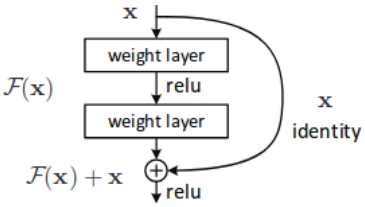


Fig. 3.8 ResNext Working

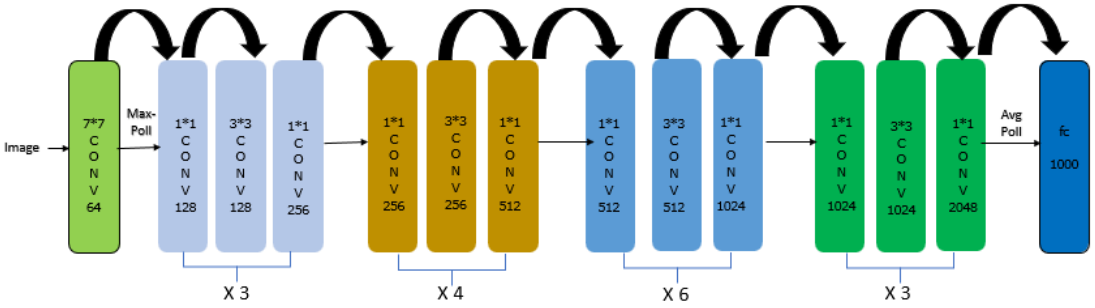


Fig. 3.9 Overview of ResNext Architecture

3.3.4.2 SEQUENTIAL LAYER

The Sequential layer serves as a container for Modules, allowing them to be stacked and executed simultaneously. In this case, the Sequential layer is employed to store the feature vector obtained from the ResNext model in a structured manner. This facilitates the sequential passing of the feature vector to the LSTM, enabling temporal analysis of the data.

3.3.4.3 LSTM LAYER

LSTM is employed to process the video frames in a sequential manner, allowing for the identification of temporal changes. The 2048-dimensional feature vectors are utilized as input to the LSTM. Our model includes one LSTM layer with 2048 latent dimensions and 2048 hidden layers.

Additionally, a dropout rate of 0.4 is applied to enhance the model's performance. By sequentially analyzing the frames, LSTM enables us to assess temporal variations in the video. This involves comparing the frame at time 't' with the frame 't-n', where 'n' can represent any number of frames preceding 't'.

The model also consists of Leaky Relu activation function. A linear layer of 2048 input features and 2 output features are used to make the model capable of learning the average rate of correlation between eh input and output. An adaptive average polling layer with the output parameter 1 is used in the model. Which gives the the target output size of the image of the form H x W. For sequential processing of the frames a Sequential Layer is used. The batch size of 4 is used to perform batch training. A SoftMax layer is used to get the confidence of the model during prediction.

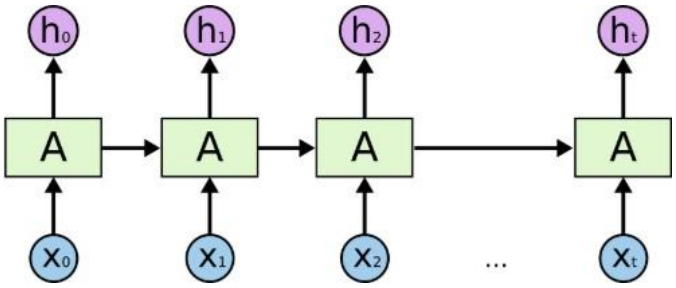


Fig. 3.10 Overview of LSTM Architecture

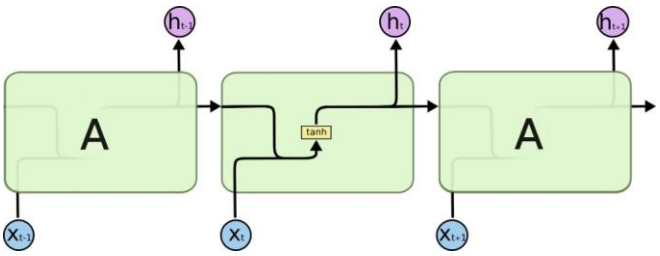


Fig. 3.11 Internal LSTM Architecture

3.3.4.4 RELU LAYER

Rectified Linear Unit (ReLU) is an activation function those outputs 0 when the input is negative, and the input itself when it is positive. This function closely resembles the behavior of our biological neurons. Unlike the sigmoid function, ReLU is non-linear and does not suffer from backpropagation errors. Additionally, when constructing larger Neural Networks, utilizing ReLU leads to faster model building due to its computational efficiency.

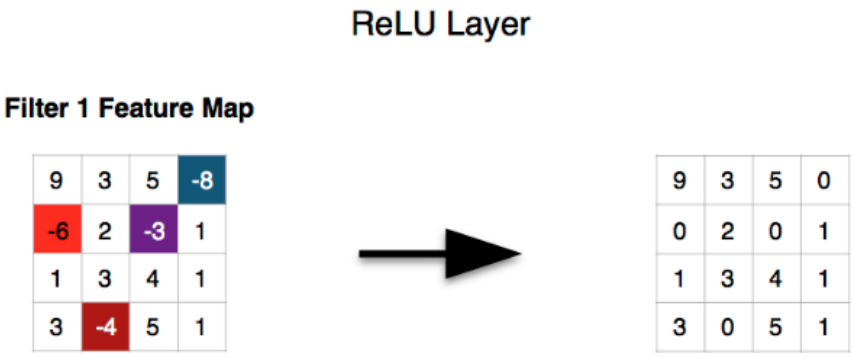


Fig. 3.12 ReLU Layer Working

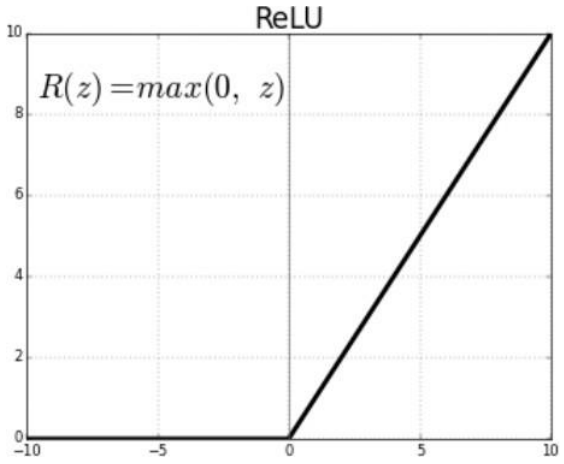


Fig. 3.13 Relu Activation function

3.3.4.5 DROPOUT LAYER

To mitigate overfitting in the model, a Dropout layer with a dropout rate of 0.4 is incorporated. This layer aids in generalization by randomly setting the output of specific neurons to 0 during training. By introducing this randomness, the cost function becomes

more sensitive to the activation of neighboring neurons, which alters the weight updates during the backpropagation process. As a result, the model becomes less likely to rely excessively on individual neurons, thus improving its ability to generalize to unseen data.

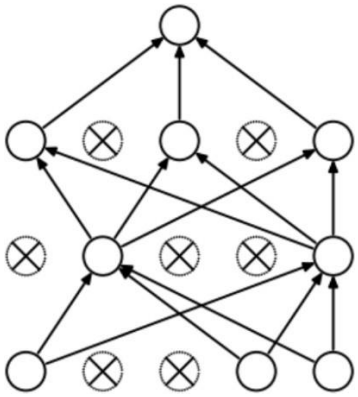


Fig. 3.14 Dropout layer overview

3.3.4.6 ADAPTIVE AVERAGE POOLING LAYER

It is used to reduce variance, reduce computation complexity, and extract low level features from neighborhood. 2-dimensional Adaptive Average Pooling Layer is used in the model.

3.3.5 HYPER-PARAMETER TUNING

It is the process of choosing the perfect hyper-parameters for achieving the maximum accuracy. After reiterating many times on the model. The best hyper-parameters for our dataset are chosen. To enable the adaptive learning rate Adam optimizer with the model parameters is used. The learning rate is tuned to 1e-5 (0.00001) to achieve a better global minimum of gradient descent. The weight decay used is 1e-3. As this is a classification problem, to calculate the loss a cross entropy approach is used. To use the available computation power properly batch training is used. The batch size is taken of 4. A batch size of 4 is tested to be ideal size for training in our development environment. The User Interface for the application is developed using Django framework. Django is used to enable the scalability of the application in the future. The first page of the User interface i.e index.html contains a tab to browse and upload the video. The uploaded video is then passed to the model and prediction is made by the model. The model returns the output whether the video is real or fake along with the confidence of the model.

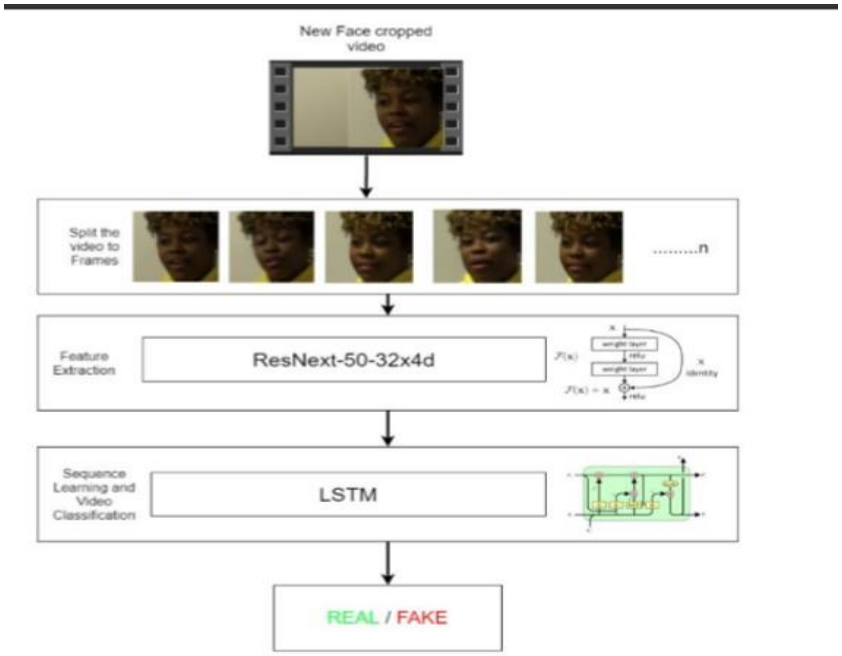


Fig. 3.15 Overview of Proposed model

3.4 MODEL TRAINING DETAILS

The training process involves optimizing the model's parameters through techniques like supervised learning, adversarial training, or self-supervised learning. The dataset is then divided into training and validation sets for model development. The deep learning model is trained using techniques like transfer learning, fine-tuning, and data augmentation, optimizing for high accuracy and robustness in detecting deepfake content.

3.4.1 DATA SPLIT

The dataset is split into train and test dataset with a ratio of 70% train videos (4,200) and 30% (1,800) test videos. The train and test split is a balanced split i.e. 50% of the real and 50% of fake videos in each split.

```
header_list = ["file","label"]
labels = pd.read_csv('/content/drive/MyDrive/deepfake/labels/Gobal_metadata.csv',names=header_list)
print(labels)
train_videos = video_files[:int(0.7*len(video_files))]
valid_videos = video_files[int(0.7*len(video_files)):]
print("train : " , len(train_videos))
print("test : " , len(valid_videos))
```

Fig. 3.16 Snapshot of dataset split

3.4.2 DATA LOADER

It is used to load the videos and their labels with a batch size of 4.

```
train_data = video_dataset(train_videos,labels,sequence_length = 10,transform = train_transforms)
# print(train_data)
val_data = video_dataset(valid_videos,labels,sequence_length = 10,transform = train_transforms)
train_loader = DataLoader(train_data,batch_size = 4,shuffle = True,num_workers = 4)
valid_loader = DataLoader(val_data,batch_size = 4,shuffle = True,num_workers = 4)
image,label = train_data[1]
im_plot(image[0,:,:,:])
```

Fig. 3.17 Snapshot of data loader

3.4.3 TRAINING

The training is done for 50 epochs with a learning rate of 1e-5 (0.00001), weight decay of 1e-3 (0.001) using the Adam optimizer.

```
for epoch in range(1,num_epochs+1):
    l, acc = train_epoch(epoch,num_epochs,train_loader,model,criterion,optimizer)
    train_loss_avg.append(l)
    train_accuracy.append(acc)
    true,pred,tl,t_acc = test(epoch,model,valid_loader,criterion)
    test_loss_avg.append(tl)
    test_accuracy.append(t_acc)
plot_loss(train_loss_avg,test_loss_avg,len(train_loss_avg))
plot_accuracy(train_accuracy,test_accuracy,len(train_accuracy))
print(confusion_matrix(true,pred))
print_confusion_matrix(true,pred)
```

```
[Epoch 1/50] [Batch 233 / 234] [Loss: 0.691465, Acc: 53.21%]Testing
[Batch 58 / 59]  [Loss: 0.629502, Acc: 73.50%]
Accuracy 73.5042735042735
[Epoch 2/50] [Batch 233 / 234] [Loss: 0.530540, Acc: 75.91%]Testing
[Batch 58 / 59]  [Loss: 0.398072, Acc: 83.76%]
Accuracy 83.76068376068376
```

Fig. 3.18 Snapshot of training model

3.4.4 ADAM OPTIMIZER

To enable the adaptive learning rate Adam optimizer with the model parameters is used.

```
optimizer = torch.optim.Adam(model.parameters(), lr= lr,weight_decay = 1e-5)
criterion = nn.CrossEntropyLoss().cuda()
```

Fig. 3.19 Snapshot of Adam optimizer

3.4.5 CROSS ENTROPY

To calculate the loss function Cross Entropy approach is used because a classification problem is being trained.

3.4.6 SOFTMAX LAYER

The SoftMax function is a type of squashing function that confines the output within the range of 0 to 1. This characteristic allows the output to be directly interpreted as probability. SoftMax functions are specifically designed for multi-class classification tasks, where they determine the probabilities of multiple classes simultaneously. Due to the interpretability of the outputs as probabilities (which sum up to 1), a SoftMax layer is typically employed as the final layer in neural network architectures.

It's important to note that the number of nodes in the SoftMax layer must match the number of classes in the output. In our case, the SoftMax layer consists of two output nodes representing the classes "REAL" or "FAKE". Additionally, the SoftMax layer provides us with the confidence or probability associated with each prediction.

3.4.7 CONFUSION MATRIX

A confusion matrix is a valuable tool for analyzing the predictions made by a classification model. It offers a concise summary of the model's correct and incorrect predictions for each class. The primary objective of the confusion matrix is to highlight the areas where the classification model encounters difficulties in making accurate predictions. It enables us to not only assess the overall errors made by the model but also identify the specific types of errors made. By utilizing the confusion matrix, the performance of our model can be effectively evaluated and its accuracy can be calculated.

```
import seaborn as sn
#Output confusion matrix
def print_confusion_matrix(y_true, y_pred):
    cm = confusion_matrix(y_true, y_pred)
    print('True positive = ', cm[0][0])
    print('False positive = ', cm[0][1])
    print('False negative = ', cm[1][0])
    print('True negative = ', cm[1][1])
    print('\n')
    df_cm = pd.DataFrame(cm, range(2), range(2))
    sn.set(font_scale=1.4) # for label size
    sn.heatmap(df_cm, annot=True, annot_kws={"size": 16}) # font size
    plt.ylabel('Actual label', size = 20)
    plt.xlabel('Predicted label', size = 20)
    plt.xticks(np.arange(2), ['Fake', 'Real'], size = 16)
    plt.yticks(np.arange(2), ['Fake', 'Real'], size = 16)
    plt.ylim([2, 0])
    plt.show()
    calculated_acc = (cm[0][0]+cm[1][1])/(cm[0][0]+cm[0][1]+cm[1][0]+ cm[1][1])
    print("Calculated Accuracy",calculated_acc*100)
```

Fig. 3.20 Snapshot of building confusion matrix

3.4.8 EXPORT MODEL

After the model is trained it is exported. So that it can be used for prediction on real time data.

```
                num_epochs,
                i,
                len(data_loader),
                losses.avg,
                accuracies.avg))
torch.save(model.state_dict(),'/content/drive/MyDrive/model_84_acc_10_frames_final_data.pt')
return losses.avg,accuracies.avg
```

Fig. 3.21 Snapshot of model export

3.4.9 MODEL PREDICTION DETAILS

• The model is loaded in the application

• The new video for prediction is preprocessed 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.

```
video_dataset = validation_dataset(path_to_videos,sequence_length = 20,transform = train_transforms)
model = Model(2).cuda()
path_to_model = '/content/drive/MyDrive/model_84_acc_10_frames_final_data.pt'
model.load_state_dict(torch.load(path_to_model))
model.eval()
for i in range(0,len(path_to_videos)):
  print(path_to_videos[i])
  prediction = predict(model,video_dataset[i],'./')
  if prediction[0] == 1:
    print("REAL")
  else:
    print("FAKE")
```

Fig. 3.22 Snapshot of result prediction

3.5 HARDWARE AND SOFTWARE REQUIREMENT

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. After analysis PyTorch framework is used along with python3 language for programming. PyTorch was chosen as it has good support to CUDA i.e., Graphic Processing Unit (GPU) and it is customize-able. Google Cloud Platform for training the final model on large amount of dataset.

- **Hardware Resources Required**

Table 3.1: Hardware Requirements

Sr. No.	Parameter	Minimum Requirement
1	Intel Xeon E5 2637	3.5 GHz
2	RAM	16 GB
3	Hard Disk	100 GB
4	Graphic card	NVIDIA GeForce GTX Titan (12 GB RAM)

- **Software Resources Required**

  Platform:

  1. Operating System: Windows 7+
  2. Programming Language: Python 3.0
  3. Framework: PyTorch 1.4, Google colab
  4. Cloud platform: Google Cloud Platform
  5. Libraries: OpenCV, Face-recognition, matplotlib, Cuda

CHAPTER 4
RESULTS AND DISCUSSION

4.1 DISCUSSION

This chapter includes all the performance analysis and results of deepfake detection of images over social media.

The metrics used for evaluation of deepfake detection are Accuracy, Precision, Recall and F1-score. The confusion matrix shown in Figure 4.1 is used to calculate the different parameters.

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

Fig. 4.1 Confusion Matrix

True Positive: When predicted "yes," the actual result was also "yes."

True Negative: When predicted "no," the actual result was also "no."

False Positive: When predicted "yes," the actual result was also "no."

False Negative: When predicted "no," the actual result was also "yes."

The various parameters are calculated using equations 4.1:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \qquad (4.1)$$

4.2 RESULTS AND ANALYSIS

The proposed model can perform better than traditional models for detection of deepfake videos. The model Inception_v3 gives an accuracy of 87% compared to the proposed method which gives an accuracy of 89%. The proposed model uses ResNext50 model which is proven better than other traditional models.

CNN in deep learning is being proven more accurate in classifying whether a particular video is deepfake or pristine.

Training the model:

Fig 4.2 shows the model is trained with 50 iterations, the accuracy and loss corresponding to each epoch. The accuracy of our model is improving slowly with each epoch and the loss is being slowly decreased.

```
Accuracy 89.31623931623932
[Epoch 46/50] [Batch 233 / 234] [Loss: 0.113480, Acc: 99.68%]Testing
[Batch 58 / 59]  [Loss: 0.457600, Acc: 88.03%]
Accuracy 88.03418803418803
[Epoch 47/50] [Batch 233 / 234] [Loss: 0.123276, Acc: 99.36%]Testing
[Batch 58 / 59]  [Loss: 0.348028, Acc: 88.46%]
Accuracy 88.46153846153847
[Epoch 48/50] [Batch 233 / 234] [Loss: 0.113431, Acc: 99.68%]Testing
[Batch 58 / 59]  [Loss: 0.400117, Acc: 88.89%]
Accuracy 88.88888888888889
[Epoch 49/50] [Batch 233 / 234] [Loss: 0.149066, Acc: 98.07%]Testing
[Batch 58 / 59]  [Loss: 0.713489, Acc: 82.91%]
Accuracy 82.90598290598291
[Epoch 50/50] [Batch 233 / 234] [Loss: 0.140824, Acc: 98.39%]Testing
[Batch 58 / 59]  [Loss: 0.359227, Acc: 89.32%]
Accuracy 89.31623931623932
```

Fig. 4.2 Snapshot of Model-1 ResNext50 Epochs

```
[Batch 58 / 59]  [Loss: 0.240683, Acc: 87.61%]
Accuracy 87.6068376068376
[Epoch 16/20] [Batch 233 / 234] [Loss: 0.293158, Acc: 92.29%]Testing
[Batch 58 / 59]  [Loss: 0.354630, Acc: 85.90%]
Accuracy 85.8974358974359
[Epoch 17/20] [Batch 233 / 234] [Loss: 0.309036, Acc: 91.33%]Testing
[Batch 58 / 59]  [Loss: 0.258396, Acc: 89.32%]
Accuracy 89.31623931623932
[Epoch 18/20] [Batch 233 / 234] [Loss: 0.245624, Acc: 93.36%]Testing
[Batch 58 / 59]  [Loss: 0.286915, Acc: 87.18%]
Accuracy 87.17948717948718
[Epoch 19/20] [Batch 233 / 234] [Loss: 0.259802, Acc: 93.90%]Testing
[Batch 58 / 59]  [Loss: 0.270999, Acc: 88.89%]
Accuracy 88.88888888888889
[Epoch 20/20] [Batch 233 / 234] [Loss: 0.230171, Acc: 95.29%]Testing
[Batch 58 / 59]  [Loss: 0.302903, Acc: 87.18%]
Accuracy 87.17948717948718
```

Fig. 4.3 Snapshot of Model-2 Inception_v3 Epochs

Fig 4.3 shows the model is trained with 20 iterations, training loss is not constant after every iteration, and training accuracy is increased after every iteration as seen in figure 4.3.

4.2.1 LOSS GRAPHS

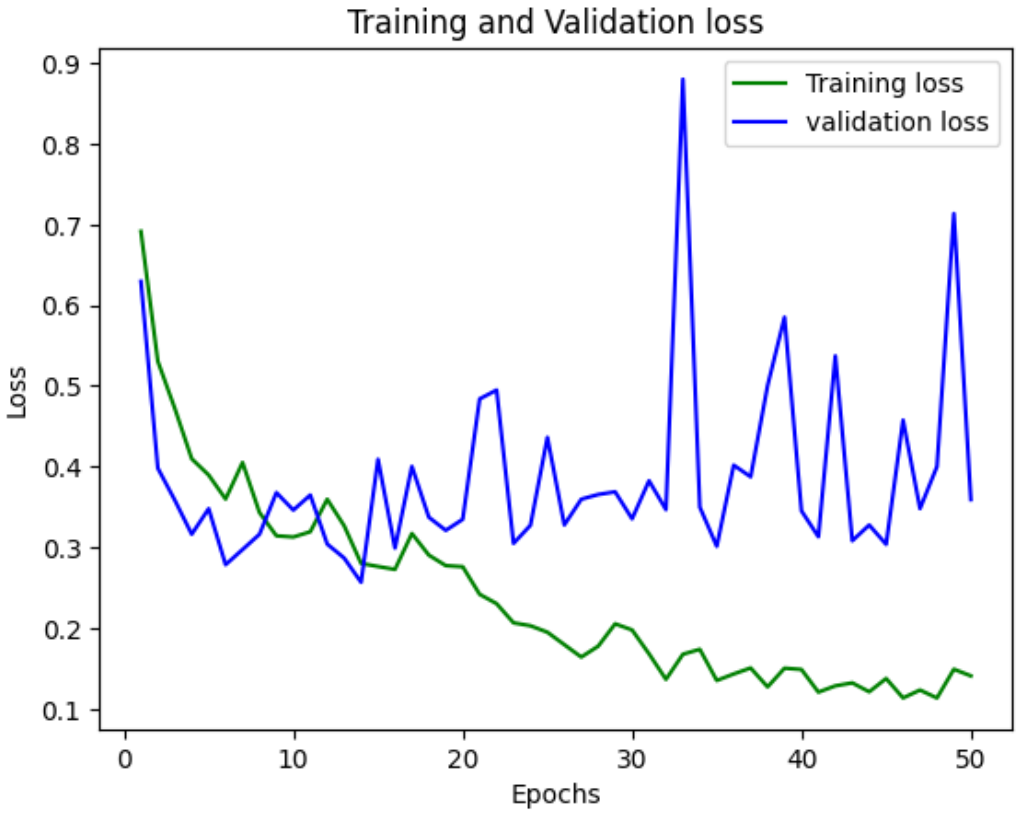


Fig. 4.4 Epoch vs loss graph for Model -1 ResNext50

Fig 4.4 shows the training as well as testing loss values for model-1 decreases after every iteration. It can be observed that training loss is gradually decreased.

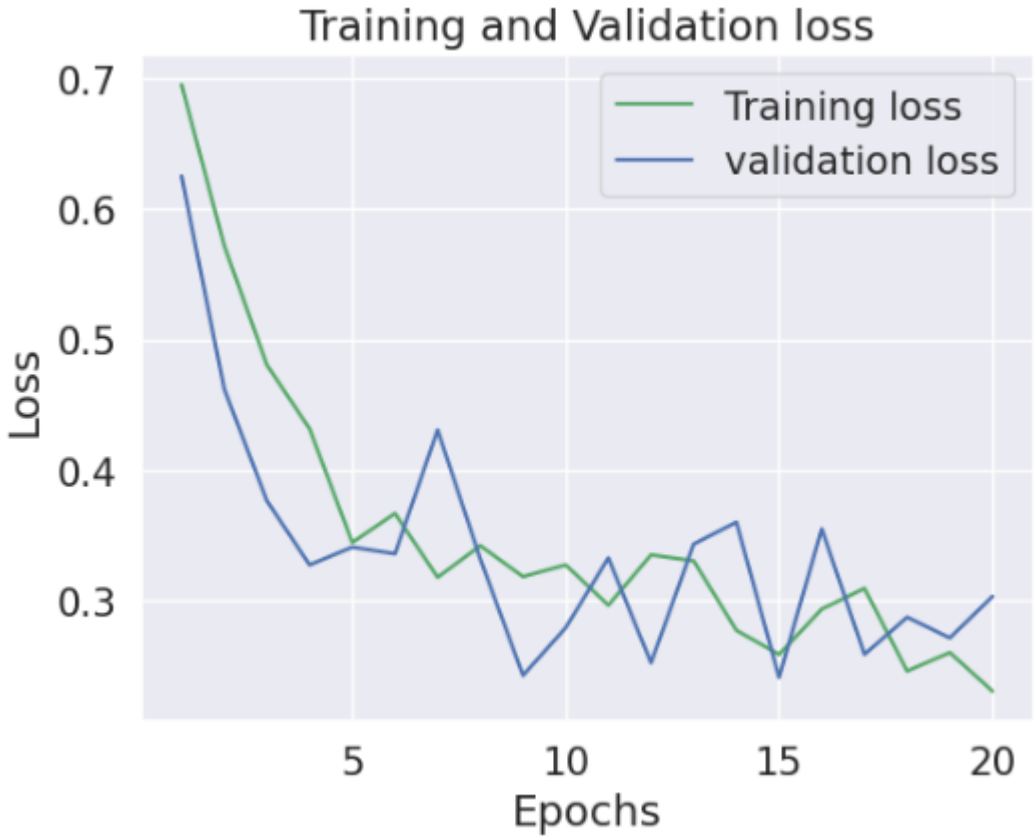


Fig. 4.5 Epoch vs loss graph for Model-2 Inception_v3

Fig 4.5 shows the training as well as testing loss values for model 2. It can be observed that training loss is constant over the period and testing loss decreases instantly and then becomes constant over a period.

4.2.2 ACCURACY GRAPHS

Fig 4.6 indicates the proposed model in fig gives training accuracy of 98% and validation accuracy of 89% as seen from above graph. The model outperforms other model consideration the amount of computation required for testing and training of data.

Fig. 4.6 Epoch vs Model Accuracy graph for Model-1 ResNext50

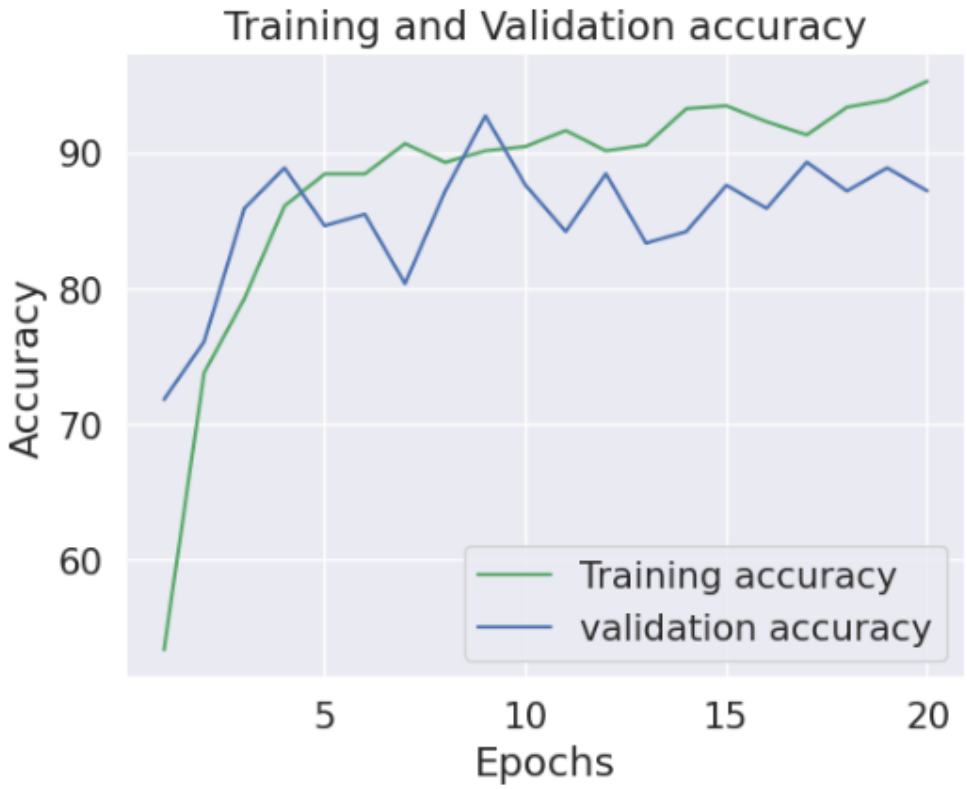


Fig. 4.7 Epoch vs Model Accuracy graph for Model-2 Inception_v3

Fig 4.7 indicates the proposed model in fig gives training accuracy of 95% and validation accuracy of 87% as seen from above graph.

The test accuracy, Loss and Validation Accuracy for each model are below:

Results: The Training Accuracy, Loss and Validation Accuracy of Model-1 and 2 are showed in table 4.1.

Table 4.1 Model 1 and Model 2 Comparison

	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
Model 1(ResNext50)	0.9839	0.1408	0.8932	0.3592
Model 2(Inception_v3)	0.9529	0.2301	0.8718	0.3029

In Table 4.1 indicates the proposed ResNext50 model has training accuracy of 98% and testing accuracy of 89% and loss of 14% which is good in comparison to traditional Inception_v3 model.

Confusion Matrix

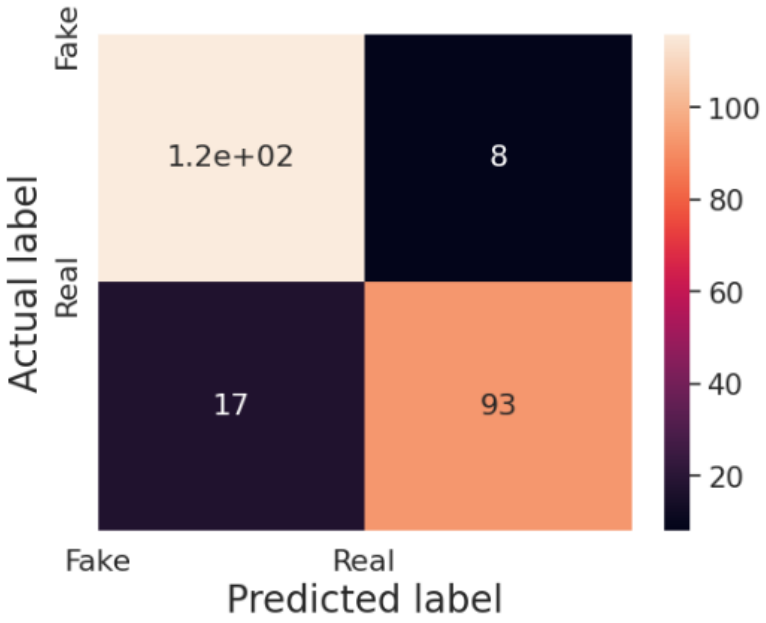


Fig 4.8: Confusion Matrix for ResNext50

True positive = 116
False positive = 8
False negative = 17
True negative = 93

Fig 4.8 indicates the true and predicted label on testing data set,116 true positive and 93 true negative, 8 false positive and 17 false negative.

The model gives precision of 0.93, recall of 0.87, accuracy of 0.89 and F1 score of 0.905

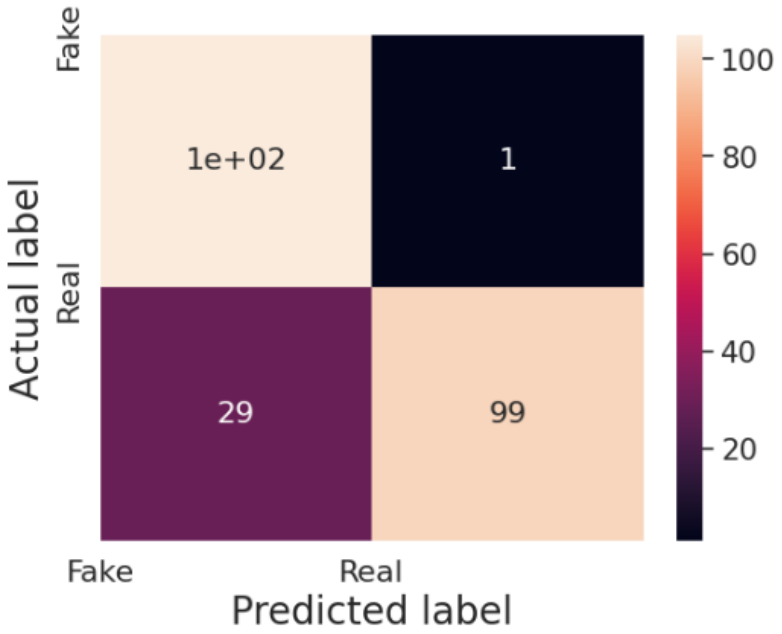


Fig 4.9: Confusion Matrix for Inception_v3

True positive = 105
False positive = 1
False negative = 29
True negative = 99

Fig 4.9 indicates the true and predicted label on testing data set,105 true positive and 99 true negative, 1 false positive and 29 false negative.

The model gives precision of 0.99, recall of 0.78, accuracy of 0.87 and F1 score of 0.87

4.2.3 COMPARISON GRAPH

The comparison graphs for CNN Model – ResNext50 and Inception_v3.

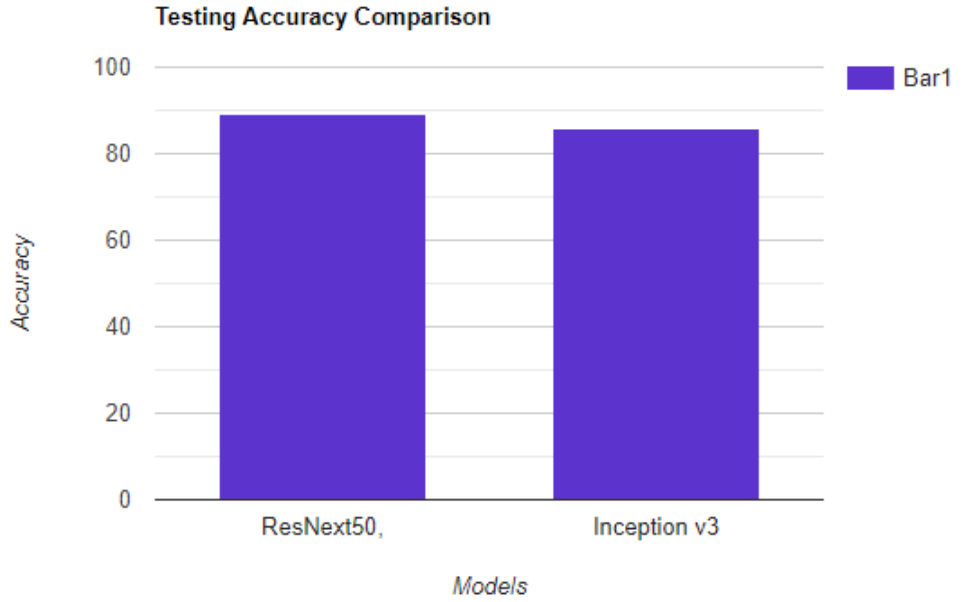


Fig.4.10: Testing Accuracy Comparison

Fig 4.10 indicates the proposed ResNext50 model gives better testing accuracy as it consists of 50 layer, as compared to Inception_v3 which has an accuracy of 87%.

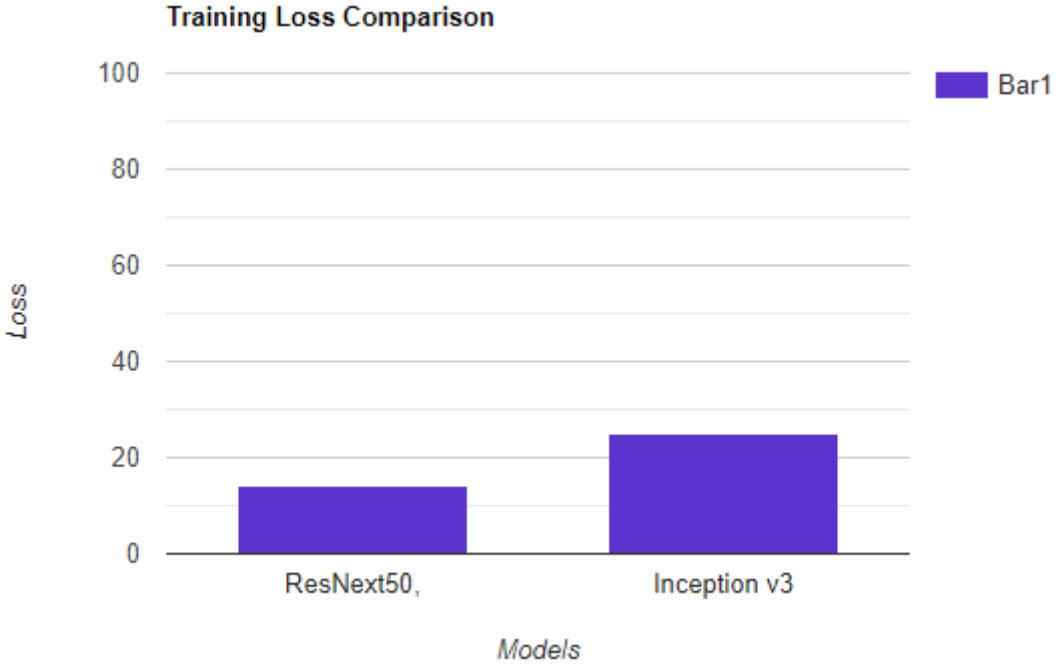


Fig.4.11: Training Loss Comparison

Fig 4.11 indicates The proposed ResNext50 model has less training loss compared to traditional Inception_v3 which has training loss of 24% means less the loss and more accuracy.

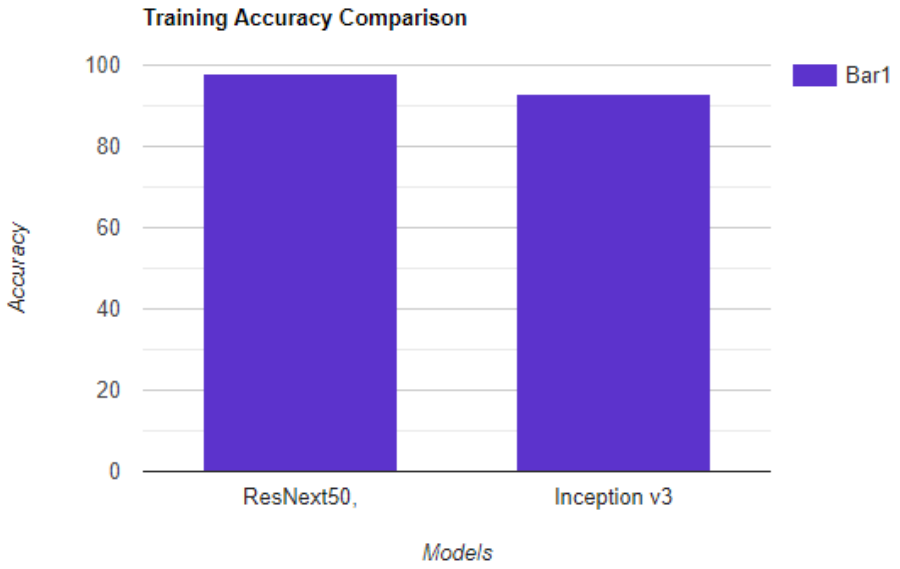


Fig.4.12: Training Accuracy Comparison

Fig 4.12 indicates the proposed ResNext50 model has better training accuracy which is 98% compared to Inception_v3 which is 95%. The proposed ResNext50 consists of 50 layers which gives better results compared to traditional model.

4.2.4 PREDICTION

Deepfake detection models make predictions by feeding video inputs into the trained neural network, which computes a probability score indicating the likelihood of the input being a deepfake.

Fig. 4.13.a REAL video Frame

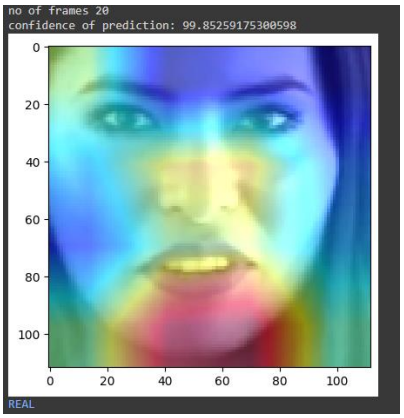


Fig. 4.13.b Predict as REAL video.

Confidence of prediction: 99%

Result: REAL

A real video frame as shown in Fig 4.13.a was given as input to the model. From Fig 4.13.b it can be observed that the model predicted it as real. Also, the confidence of prediction achieved was 99%.

Fig. 4.14.a Deepfake video Frame.

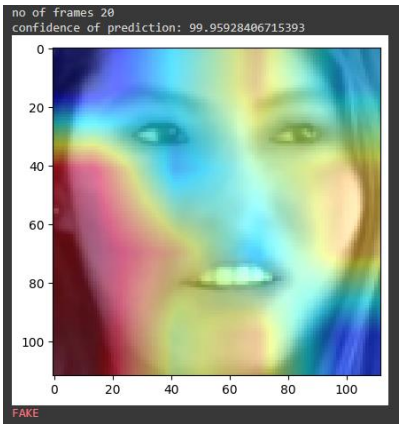


Fig. 4.14.b Predict as FAKE video.

Confidence of prediction: 99%

Result: FAKE

A real video frame as shown in Fig 4.14.a was given as input to the model. From Fig 4.14.b it can be observed that the model predicted it as fake. Also, the confidence of prediction achieved was 99%.

Fig. 4.15.a REAL video Frame

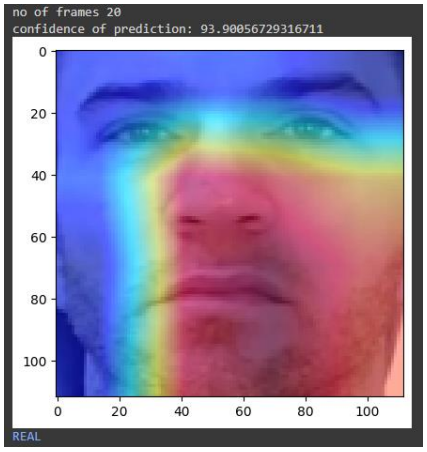


Fig. 4.15.b Predict as REAL video.

Confidence of prediction: 93%

Result: REAL

A real video frame as shown in Fig 4.15.a was given as input to the model. From Fig 4.15.b it can be observed that the model predicted it as real. Also, the confidence of prediction achieved was 93%.

Fig. 4.16.a Deepfake video Frame

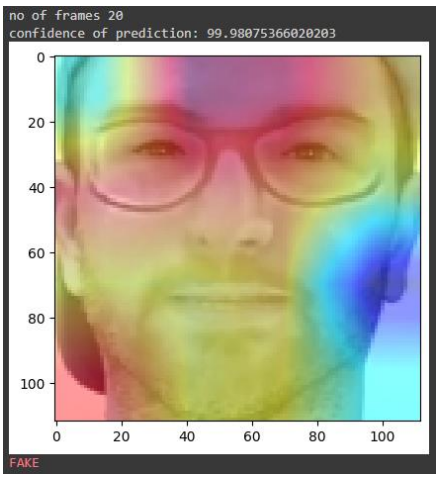


Fig. 4.16.b Predict as FAKE video.

Confidence of prediction: 99%

Result: FAKE

A real video frame as shown in Fig 4.16.a was given as input to the model. From Fig 4.16.b it can be observed that the model predicted it as fake. Also, the confidence of prediction achieved was 99%.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

A neural network-based approach to classify the video as deep fake or real, along with the confidence of proposed model is presented. Our method can predict the output by processing 1 second of video (10 frames per second) with a good accuracy. The model is implemented by using pre-trained ResNext CNN model to extract the frame level features and LSTM for temporal sequence processing to spot the changes between the t and t-1 frame. Our model can process the video in the frame sequence of 10,20,40,60,80,100.

5.2 FUTURE WORK

In future, the system is deployed as a web based platform for ease and real time usage. Currently only Face Deep Fakes are being detected by the algorithm, but the algorithm can be enhanced in detecting full body deep fakes.

REFERENCES

[1] V. -N. Tran, S. -G. Kwon, S. -H. Lee, H. -S. Le and K. -R. Kwon, "Generalization of Forgery Detection With Meta Deepfake Detection Model," in IEEE Access, vol. 11, pp. 535-546, 2023, doi: 10.1109/ACCESS.2022.3232290.

[2] M. Jiwtode, A. Asati, S. Kamble and L. Damahe, "Deepfake Video Detection using Neural Networks," 2022 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS), Pune, India, 2022, pp. 1-5, doi: 10.1109/ICBDS53701.2022.9935984.

[3] L. Guarnera, O. Giudice and S. Battiato, "Fighting Deepfake by Exposing the Convolutional Traces on Images," in IEEE Access, vol. 8, pp. 165085-165098, 2020, doi: 10.1109/ACCESS.2020.3023037.

[4] J. C. Neves, R. Tolosana, R. Vera-Rodriguez, V. Lopes, H. Proença and J. Fierrez, "GANprintR: Improved Fakes and Evaluation of the State of the Art in Face Manipulation Detection," in IEEE Journal of Selected Topics in Signal Processing, vol. 14, no. 5, pp. 1038-1048, Aug. 2020, doi: 10.1109/JSTSP.2020.3007250.

[5] D. Pan, L. Sun, R. Wang, X. Zhang and R. O. Sinnott, "Deepfake Detection through Deep Learning," 2020 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT), Leicester, UK, 2020, pp. 134-143, doi: 10.1109/BDCAT50828.2020.00001.

[6] X. Wu, Z. Xie, Y. Gao and Y. Xiao, "SSTNet: Detecting Manipulated Faces Through Spatial, Steganalysis and Temporal Features," ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 2020, pp. 2952-2956, doi: 10.1109/ICASSP40776.2020.9053969.

[7] Deepfake detection challenge dataset : https://www.kaggle.com/c/deepfake-detection- challenge.

[8] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies,Matthias Nießner, “FaceForensics++: Learning to Detect Manipulated Facial Images” in arXiv:1901.08971

[9] Yuezun Li, Siwei Lyu, “ExposingDF Videos By Detecting Face Warping Arti-facts,” in arXiv:1811.00656v3.

[10] Yuezun Li, Ming-Ching Chang and Siwei Lyu “Exposing AI Created Fake Videos by Detecting Eye Blinking” in arXiv:1806.02877v2.

[11] Huy H. Nguyen , Junichi Yamagishi, and Isao Echizen “ Using capsule net- works to detect forged images and videos ” in arXiv:1810.11215.

[12] D. Güera and E. J. Delp, "Deepfake Video Detection Using Recurrent Neural Networks," 2018 15th IEEE International Conference on Advanced Video and Sig- nal Based Surveillance (AVSS), Auckland, New Zealand, 2018, pp. 1-6.

[13] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic hu- man actions from movies. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8, June 2008. Anchorage, AK.

[14] Umur Aybars Ciftci, ˙Ilke Demir, Lijun Yin “Detection of Synthetic Portrait Videos using Biological Signals” in arXiv:1901.02212v2.

[15] ResNext Model : https://pytorch.org/hub/pytorch_vision_resnext/ accessed on 06 April 2020.

[16] TensorFlow: https://www.tensorflow.org/ .

[17] Keras: https://keras.io/ .

[18] PyTorch : https://pytorch.org/ .