JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

DEEPFAKE DETECTION THROUGH DEEP LEARNING USING RESNEXT CNN AND LSTM

[1]Kanchan Warke, [2]Nilam Dalavi
Professor of BVCOEW

[3]Shruti Nahar, [4]Dhanashri Gurule, [5]Unnati Jadhav, [6]Prachi Wadhavane
BE Student at BVCOEW,

Department of Computer Engineering,
Bharati Vidyapeeth College of Engineering for Women, Pune, India

Abstract: With the increasing computational power, the creation of indistinguishable human synthesized videos, known as deepfakes, has become remarkably easy. These realistic face-swapped deepfakes have raised concerns as they can be utilized for malicious purposes such as causing political unrest, fabricating terrorism events, spreading revenge porn, and blackmailing individuals. In this research, we present a novel deep learning-based method that effectively distinguishes AI-generated fake videos from real ones. Our approach focuses specifically on detecting replacement and reenactment deepfakes. We harness the power of Artificial Intelligence (AI) to combat the challenges posed by AI itself. The core of our system lies in a ResNext Convolutional Neural Network, which extracts frame-level features. These features are then used to train a Long Short-Term Memory (LSTM)-based Recurrent Neural Network (RNN) that classifies videos as either manipulated (deepfake) or authentic (real).To ensure real-time applicability and enhance the model's performance on real-world data, we evaluate our method using a large and balanced dataset. This dataset is prepared by blending various available datasets, including FaceForensic++[1], Deepfake Detection Challenge[2], and Celeb-DF[3]. Additionally, we demonstrate how our system achieves competitive results through a simple and robust approach.In summary, our research aims to address the challenges posed by deepfakes by utilizing AI technologies. By leveraging a ResNext CNN and LSTM-based RNN, we successfully detect and classify manipulated videos. Through extensive evaluation on mixed and balanced datasets, we showcase the effectiveness and efficiency of our approach in real-time scenarios.

Keywords: Deepfake Video Detection, convolutional Neural network (CNN), recurrent neural network (RNN), Long short term memory(LSTM).

I. INTRODUCTION:

The widespread use of social media has been greatly facilitated by the advancement in smartphone cameras and the availability of reliable internet connections, enabling easy creation and sharing of digital videos. Deep learning has gained immense power due to increased processing capabilities, surpassing what was once considered unimaginable. However, this progress has also brought about new challenges. The emergence of deep generative adversarial models capable of modifying audio and video samples has led to the creation of "DeepFake" videos. These videos are often spread through social media platforms, leading to issues such as spamming and the dissemination of false information. The existence of such DeepFake videos can have detrimental effects, causing intimidation and deception. Therefore, it is crucial to develop technologies that can effectively detect and identify these fakes, thereby preventing their spread online.

To address this problem, we present a novel deep learning-based technique that successfully distinguishes between AI-generated fake videos (DF Videos) and genuine ones. Understanding the workings of the Generative Adversarial Network (GAN) is pivotal in identifying DeepFake videos. GANs use input videos and images of a target person to replace their faces with those of another person (the "source"). Deep adversarial neural networks are trained on face photos and target videos to automatically map faces and facial expressions, forming the foundation of DeepFake creation. With appropriate post-processing, these generated videos can achieve a high level of realism. The GAN replaces the input image in each frame by dividing the video into frames and then reconstructing the video using techniques like autoencoders.

Our proposed technique is based on the same underlying process employed by GANs to generate DeepFake videos. We focus on

a specific feature of these videos, where face photos synthesized by the DF algorithm undergo an affine warping process to align

with the facial features of the source person while maintaining a fixed size. This warping introduces discernible artifacts in the resulting DeepFake video due to resolution discrepancies between the warped face area and the surrounding context. By comparing these created face areas with their surrounding regions, our technique can identify such artifacts. Furthermore, we capture the temporal inconsistencies between frames introduced by GAN during the video reconstruction process by splitting the video into frames, extracting features, and utilizing a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM). To streamline our approach, we directly train the ResNext CNN model to replicate the resolution inconsistencies observed in affine face wrappings.

III . Literature Survey:

The first method mentioned, "Detecting Face Warping Artifacts in Exposing DF Videos,"[1] focuses on identifying artifacts present in deep fake videos. This technique utilizes a specialized Convolutional Neural Network to compare the generated face regions with their neighboring areas. By examining these areas, the method aims to detect specific visual irregularities that are indicative of deep fakes. The study categorizes these artifacts into different types and takes advantage of the fact that deep fake algorithms often generate low-resolution images. These low-resolution images are then transformed to match the faces being replaced in the original video. By analyzing and comparing these transformations, the method can effectively identify the presence of face warping artifacts, thus aiding in the detection of deep fake videos.

The second method, "Detecting Eye Blinking in Exposing AI Created Fake Videos,"[2] tackles the issue of uncovering fake face videos produced using deep neural network models. The approach relies on the detection of eye blinking, which is a physiological signal that is typically not well represented in synthesized fake videos. By examining the presence or absence of eye blinking in the videos, this method can distinguish between genuine and fake content. The evaluation of this technique involves using eye-blinking detection benchmark datasets, which provide a standardized basis for assessing its performance. The results obtained so far show promise in effectively detecting videos generated by the Deep Neural Network-based software DF. However, it's worth noting that other factors, such as teeth enhancement and wrinkles on the faces, should also be taken into account for a more comprehensive detection approach.

The third method, "Detecting Forged Images and Videos Using Capsule Networks,"[3] introduces the use of capsule networks for identifying manipulated images and videos in various scenarios. This technique employs a network architecture that is specifically designed to capture hierarchical relationships and spatial arrangements of visual features. During the training phase, random noise is introduced as a means to enhance the network's robustness against different forms of manipulation. However, it should be acknowledged that the inclusion of random noise may not be an ideal solution and could potentially affect the method's performance on real-time data. Although the model shows good results on the dataset used for evaluation, its effectiveness in real-world scenarios where noise is present throughout the training phase remains to be thoroughly examined.

The fourth method, "Detecting Synthetic Portrait Videos using Biological Signals,"[4] focuses on extracting and analyzing biological signals from both authentic and fake portrait videos. This approach leverages the concept of spatial coherence and temporal consistency to capture the unique signal characteristics exhibited in genuine videos. By applying specific transformations and processing steps, such as feature set extraction and PPG (Photoplethysmogram) map generation, the method aims to differentiate between real and synthetic content. The system known as "Fake Catcher" demonstrates an accurate detection capability regardless of the video's resolution, quality, generator, or content. However, one of the main challenges associated with this method lies in formulating a differentiable loss function that effectively follows the proposed signal processing steps. Ensuring the preservation and discrimination of biological signals throughout the detection process is crucial to achieve optimal performance

IV . Proposed Architecture:

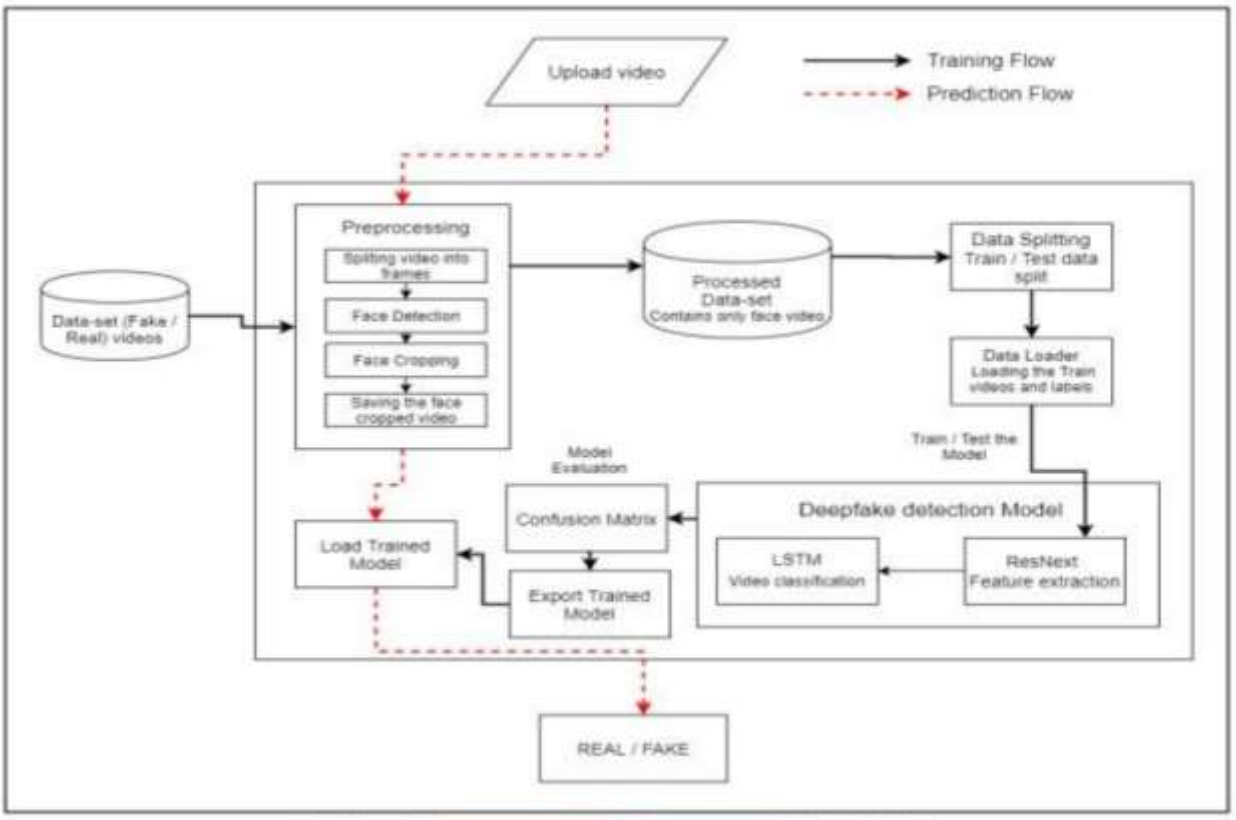

Fig 1. System Architecture

Explanation of System Architecture:

Module 1: Data-set Gathering: To create an efficient real-time prediction model, we gathered data from FaceForensic++ (FF), Deepfake Detection Challenge (DFDC), and Celeb-DF datasets. We combined these datasets with our own collected data, ensuring a balanced mix of 50% real and 50% fake videos. Preprocessing involved removing audio-altered videos from the DFDC dataset. Our final dataset comprised 81 real and 82 fake videos, totaling 163 video.

Module 2: Pre-Processing: In the pre-processing module, videos undergo various steps to remove noise and extract the required content, specifically the face. The initial step involves splitting the videos into frames. Each frame is then analyzed to detect and crop the face. The resulting cropped frames are recombined to form new videos, containing only the face regions. Frames without detected faces are excluded during pre-processing.

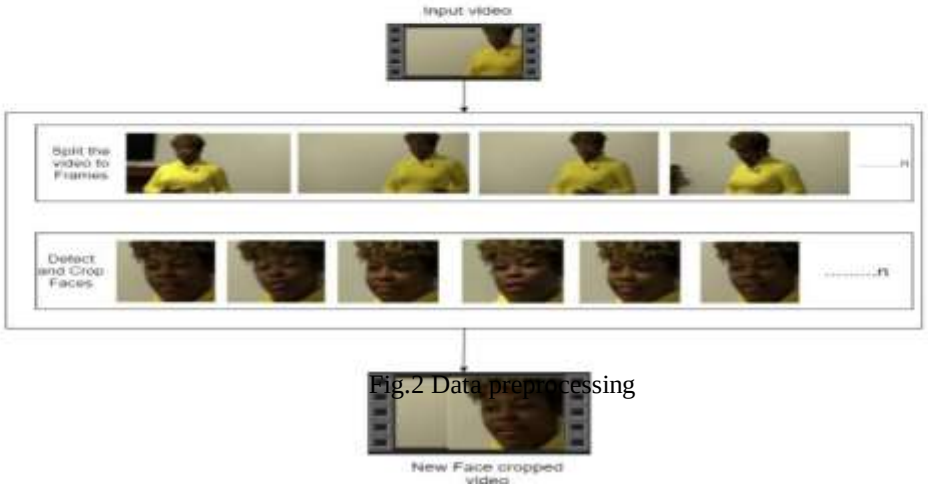

Fig.2 Data preprocessing

To ensure uniformity in the number of frames, a threshold value is determined based on the mean frame count of each video. This threshold value is chosen considering computational limitations, such as the GPU's processing power. For instance, a 10-second video at 30 frames per second (fps) would have 300 frames, which can be computationally challenging to process simultaneously. Thus, a threshold of 150 frames is selected based on the available GPU computational power. When saving frames to the new dataset, only the first 150 frames of each video are retained. The sequential order of frames is maintained to demonstrate the proper

utilization of Long Short-Term Memory (LSTM). The newly created videos are saved at a frame rate of 30 fps and a resolution of 112 x 112 pixels.

Module 3: Data-set split: The dataset is divided into a train and test dataset, with a ratio of 70% train videos and 30% test videos. The split is balanced, ensuring an equal distribution of 50% real and 50% fake videos in both the train and test sets.

Module 4: Model Architecture: Our model architecture combines both CNN and RNN components. We utilize a pre-trained ResNext CNN model for feature extraction at the frame level. These extracted features are then fed into an LSTM network to classify the videos as either deepfake or pristine. During the training process, the labels of the videos from the training split are loaded using a Data Loader and fitted into the model.

ResNext: To avoid starting from scratch, we leverage a pre-trained ResNext model for feature extraction. ResNext is a Residual CNN network specifically optimized for achieving high performance in deeper neural networks. For our experiments, we utilize the resnext50_32x4d model, which consists of 50 layers and dimensions of 32 x 4.

Next, we fine-tune the network by adding additional necessary layers and selecting an appropriate learning rate to ensure proper convergence of the model's gradient descent. The 2048-dimensional feature vectors obtained from the last pooling layers of ResNext serve as the input for the sequential LSTM component.

LSTM for Sequence Processing: For sequence processing, the 2048-dimensional feature vectors are fed into a single LSTM layer. The LSTM layer has 2048 latent dimensions and 2048 hidden layers, with a dropout probability of 0.4, which aids in achieving our objective. The purpose of using LSTM is to process the frames in a sequential manner, allowing for temporal analysis by comparing the frame at time 't' with the frame 't-n' (where 'n' represents the number of frames before 't').

The model also incorporates the Leaky ReLU activation function. A linear layer with 2048 input features and 2 output features is employed to enable the model to learn the average correlation between the input and output. To obtain an output size in the form of H x W (height x width), an adaptive average pooling layer with an output parameter of 1 is included. Sequential layer is used for sequential processing of frames, and a batch size of 4 is employed for batch training. Finally, a SoftMax layer is utilized to obtain the model's confidence during prediction.

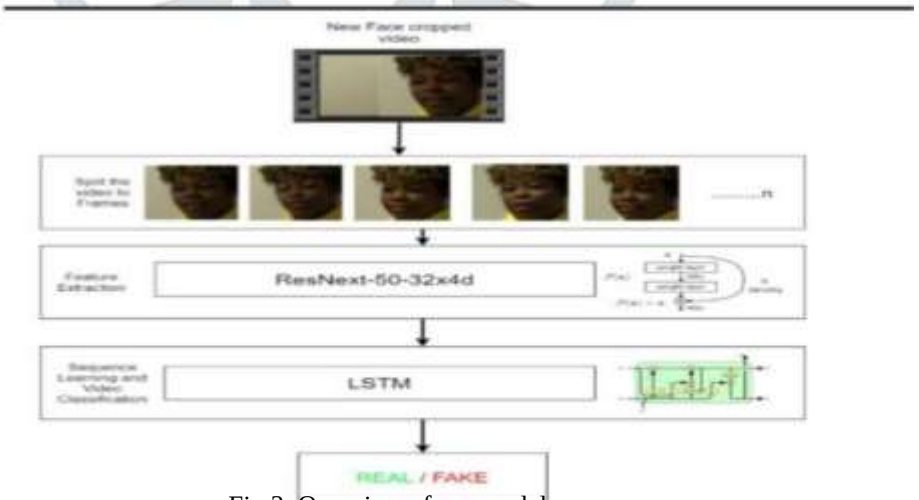

Fig 3. Overview of our model

Module 5: Hyper-parameter Tuning: Hyper-parameter tuning involves selecting the optimal hyper-parameters to maximize accuracy. After multiple iterations on the model, the best hyper-parameters for our dataset are determined. To enable adaptive learning rate, we utilize the Adam optimizer [21] with a learning rate of 1e-5 (0.00001), which helps achieve a better global minimum during gradient descent. A weight decay of 1e-3 is employed. Since this is a classification problem, the cross-entropy loss approach is utilized to calculate the loss. To make efficient use of the available computational power, batch training is implemented with a batch size of 4. This batch size has been determined to be ideal for training in our specific development environment.

For the user interface, we have developed it using the Django framework. Django offers scalability for the application in the future. The initial page of the user interface, index.html, includes a tab for browsing and uploading videos. The uploaded video is then passed to the model, which makes predictions. The model returns the output indicating whether the video is real or fake, along with the confidence of the model. This output is rendered on the predict.html page, overlaying the results on the playing video.

V. Algorithm: ReNext

CNN:

ResNeXt is a deep learning architecture that is based on residual neural networks (ResNet). It is designed to improve the representational power of the network by using a modular and scalable structure. While ResNeXt itself is not specifically designed for deepfake detection, it can be used as a backbone network in a deepfake detection system to extract meaningful features from the input data

ResNeXt introduces a concept called a "cardinality" to ResNet architectures. Cardinality refers to the number of parallel paths within each block of the network. By increasing the cardinality, ResNeXt enables the network to capture more diverse and complementary features, leading to improved performance.

Here's a high-level overview of how ResNeXt can be used in deepfake detection:

1. Input data preprocessing: The input data, such as images or video frames, undergoes preprocessing steps, which may include resizing, cropping, and normalization, to ensure consistency and facilitate efficient processing.

2. ResNeXt architecture: The ResNeXt network is constructed as the backbone of the deepfake detection system. It typically consists of multiple blocks or modules, with each block containing parallel pathways. Each pathway performs convolutional operations to extract features from the input data.

3. Feature extraction: The input data is passed through the ResNeXt network, and at each block, features are extracted using the parallel pathways. These pathways capture different aspects of the input data, allowing the network to learn diverse representations.

4. Classification: The extracted features are then fed into subsequent layers, such as fully connected layers or classifiers, to make predictions regarding the authenticity of the input data. These layers can be trained using labeled data to learn to distinguish between real and fake content.

5. Training and optimization: The entire deepfake detection system, including the ResNeXt backbone, is trained using a large dataset of labeled real and deepfake media. During training, the network's parameters are adjusted iteratively to minimize the difference between predicted labels and ground truth labels. This is typically done using optimization techniques like gradient descent.

6. Evaluation and deployment: The trained deepfake detection system, with the ResNeXt backbone, is evaluated on separate validation and test datasets to assess its performance in accurately detecting deepfakes. Once the system achieves satisfactory performance, it can be deployed for real-world applications, where it takes input data and provides predictions on whether it is real or manipulated.It's important to note that while ResNeXt can be effective for feature extraction in deepfake detection, the overall system's performance may also depend on other factors such as the choice of datasets, data augmentation techniques, and

the specific training and evaluation methodologies employed.

LSTM

Deepfake detection using LSTM (Long Short-Term Memory) networks is an approach that leverages the sequential nature of deepfake videos or temporal information in image sequences to identify inconsistencies or anomalies. LSTMs are a type of recurrent neural network (RNN) that can effectively model long-term dependencies in sequential data.

Here is a general overview of how LSTM networks can be used for deepfake detection:

1. Dataset preparation: A dataset of labeled deepfake videos is collected, along with a corresponding set of real videos for comparison. The videos can be divided into frames, and temporal information can be extracted by considering consecutive frames.

2. Preprocessing: The video frames are preprocessed, which may involve resizing, normalization, and possibly optical flow calculation between consecutive frames to capture motion information.

3. LSTM architecture: The LSTM network is designed as the backbone of the deepfake detection system. The network takes input sequences of frames or feature representations extracted from the frames.

4. Feature extraction: The LSTM network processes the input sequences, capturing temporal dependencies and patterns present in the videos. The LSTM's memory cells allow it to remember relevant information from earlier frames while considering the current frame.

5. Classification: The output of the LSTM network is fed into a classification layer, such as a fully connected layer, which predicts whether the input video is real or a deepfake. The classification layer is typically trained using labeled data and optimization techniques like gradient descent.

6. Training and optimization: The LSTM network is trained using the collected dataset, with the goal of minimizing the difference between predicted labels and ground truth labels. The network's parameters are adjusted iteratively using backpropagation through time (BPTT) to update the weights and optimize the model's performance.

7. Evaluation and deployment: The trained LSTM-based deepfake detection system is evaluated on separate validation and test datasets to assess its performance in correctly identifying deepfakes. Once the system achieves satisfactory performance, it can be deployed for real-world deepfake detection tasks.

It's important to note that the effectiveness of LSTM-based deepfake detection depends on various factors, including the quality and diversity of the training dataset, the choice of network architecture, the preprocessing techniques applied, and the specific training and evaluation methodologies used. Additionally, combining LSTM with other techniques, such as image-based features or ensemble methods, may further enhance the detection performance.

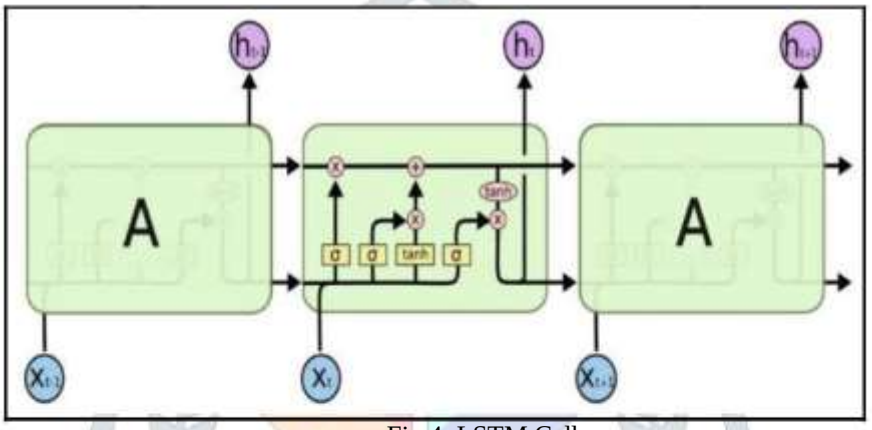

Fig 4: LSTM Cell

VI Result:

Training and Validation Loss:

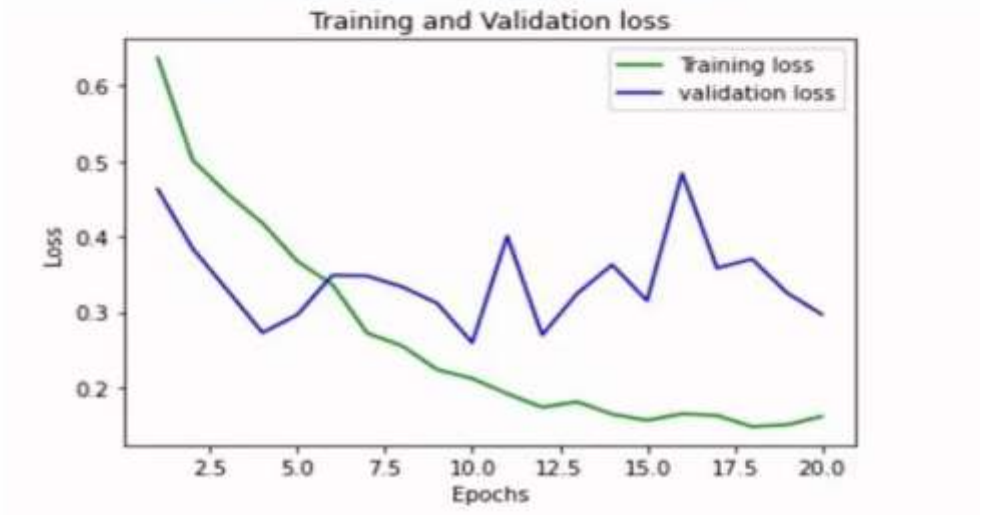

Fig 5: Training and validation loss

Training and Validation Accuracy:

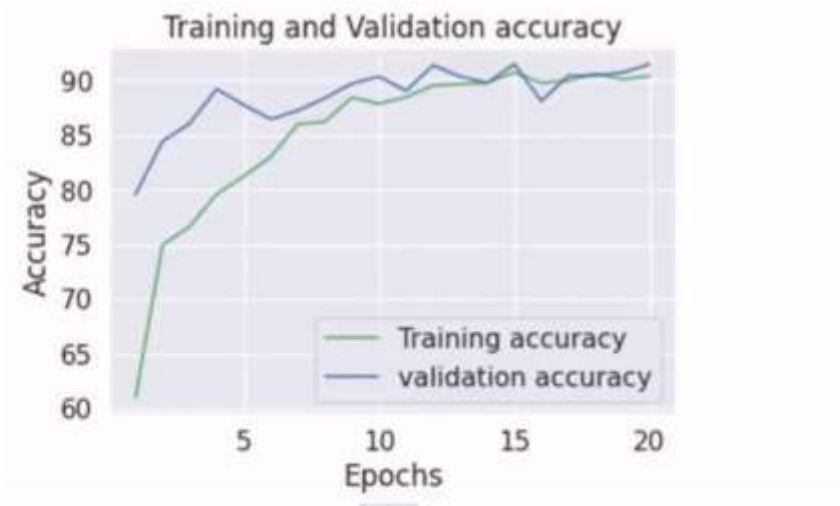

Fig 6: Training and Validation Accuracy

Confusion Matrix:

Fig 7: Confusion Matrix

Trained Model Results :

Model Name	Dataset	No. of videos	Sequence length	Accuracy
model_90_acc _20_frames_ FF_data	DeepFake Detection Challenge	100	20	90.95477
model_95_acc _40_frames_ FF_data	DeepFake Detection Challenge	100	40	95.22613
model_97_acc _60_frames_ FF_data	DeepFake Detection Challenge	100	60	97.48743
model_97_acc _80_frames_ FF_data	DeepFake Detection Challenge	100	80	97.73366

model_97_acc _100_frames_ FF_data	DeepFake Detection Challenge	100	100	97.76180
model_87_acc _20_frames_ final_data	Our Dataset	163	20	87.79160
model_84_acc _10_frames_ final_data	Our Dataset	163	10	84.21461
model_89_acc _40_frames_ final_data	Our Dataset	163	40	89.34681

Table 1. Trained model results

VII Output:

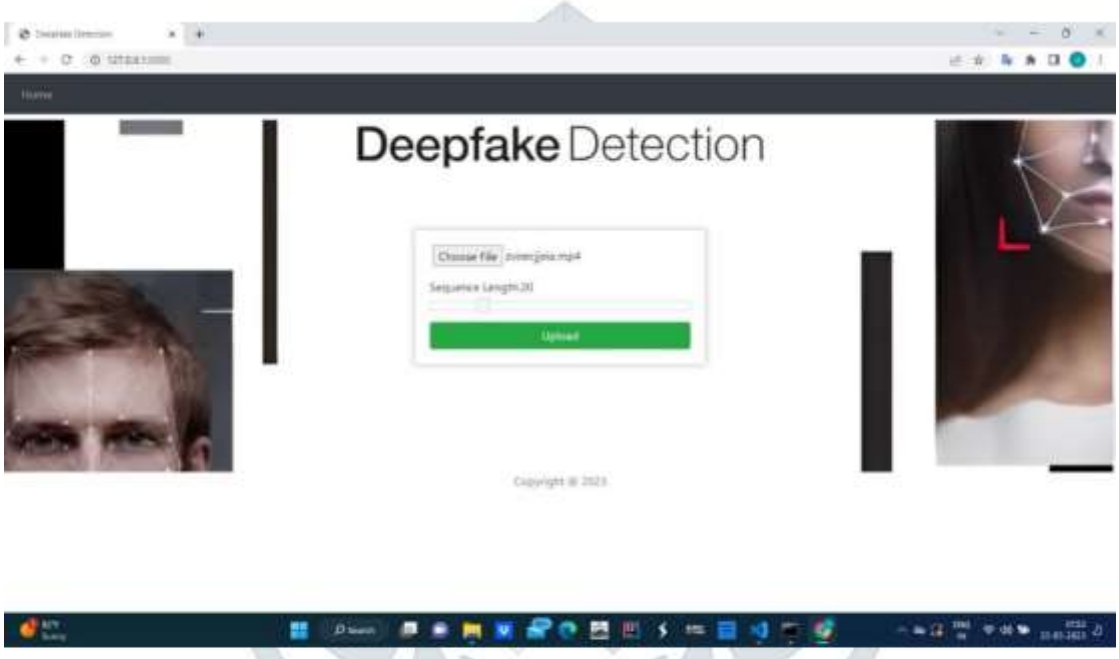

Fig.8: Home Page

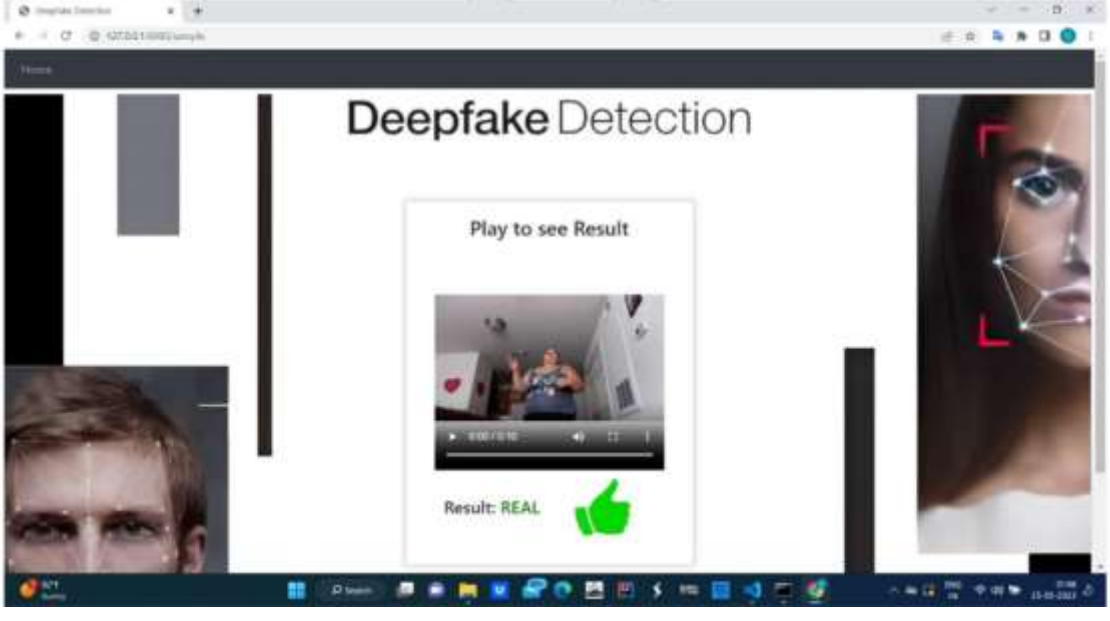

Fig.9 Real Prediction

Fig.9 Fake Prediction

Fig.10 User input validation(i)

Fig.10 User input validation(ii)

VIII. Conclusion:

We present a novel solution that utilizes a neural network architecture for the classification of videos into deepfakes or real, providing a comprehensive measure of confidence in the model's predictions. Our approach stands out for its efficiency, as it achieves accurate results by analyzing only one second of video footage, corresponding to a frame rate of 10 frames per second. To construct the model, we leverage the power of a pre-trained ResNext CNN, which excels at extracting detailed features at the frame level. Furthermore, we incorporate an LSTM component that performs sequential analysis, enabling the identification of temporal changes between consecutive frames. Notably, our model is designed to handle video sequences of varying lengths, including options such as 10, 20, 40, 60, 80, and 100 frames. By considering a diverse range of frame sequences, our solution caters to different video contexts and offers enhanced flexibility in deepfake detection.

IX. References:

[1] Tackhyun Jung, Sangwon Kim, Keecheon Kim - DeepVision:Deepfakes Detection Using Human Eye Blinking Pattern . IEEE 2020.

[2] Md Shohel Rana, Mohammad, Nur Nobi, Andrew H. Sung, Beddhu Murali, Deepfake Detection: A Systematic Literature Review. IEEE 2022.

[3] S Lyu , Deepfake Detection Current Challenges and Next Steps. IEEE 2020

[4] Deng Pan, Lixin Sun, Rui Wang, Richard O. Sinnott, Deepfake Detection through Deep Learning. IEEE 2020.

[5] TensorFlow: https://www.tensorflow.org/ (Accessed on 26 March, 2020)

[6] Yuezun Li, Siwei Lyu, "ExposingDF Videos By Detecting Face Warping Artifacts," in arXiv:1811.00656v3.

[7] PyTorch : https://pytorch.org/ (Accessed on 26 March, 2020)

[8] G. Antipov, M. Baccouche, and J.-L. Dugelay. Face aging with conditional gen- erative adversarial networks. arXiv:1702.01983, Feb. 2017

[9] J. Thies et al. Face2Face: Real-time face capture and reenactment of rgb videos. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2387–2395, June 2016. Las Vegas, NV.

[10] Face app: https://www.faceapp.com/ (Accessed on 26 March, 2020)

[11] Rossler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J. and Nießner, M. (2019) Faceforensics++: Learning to Detect Manipulated Facial Images. Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, 27-28 October 2019, 1-11.
https://doi.org/10.1109/ICCV.2019.00009

[12] de Lima, O., Franklin, S., Basu, S., Karwoski, B. and George, A. (2020) Deepfake Detection Using Spatiotemporal Convolutional Networks.

[13] Sanders, D.A. and Goodrich, S.J. (1971) The Relative Contribution of Visual and Auditory Components of Speech to Speech Intelligibility as a Function of Three Conditions of Frequency Distortion. Journal of Speech and Hearing Research, 14, 154-159.
https://doi.org/10.1044/jshr.1401.154

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

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