# Deepfake Video Detection using Machine Learning

*Aditya Dumbre[1], Omsai Alladwar[2], Venktesh Kale[3], Dr. P. A. Kadam[4]*
*[1,2,3]Student, [4]Professor*
*[1,2,3,4]Department of Computer Engineering*
*[1,2,3,4] Smt. Kashibai Navale College of Engineering, Pune, Maharashtra, India*

---------------------------------------------------------------------***---------------------------------------------------------------------

**Abstract -** The rise of DeepFake (DF) technology poses a significant threat to the authenticity and reliability of digital media. While such videos once required expert skills and high-end software, the recent advancement of deep learning tools has made the generation of hyper-realistic synthetic media widely accessible. This paper presents a robust and efficient deep learning-based solution for detecting DeepFake videos by combining spatial and temporal analysis using a Convolutional Neural Network (CNN) and a Long Short-Term Memory (LSTM) based Recurrent Neural Network (RNN). Our final system leverages a ResNeXt50 CNN for frame-level feature extraction and an LSTM layer to model temporal inconsistencies across sequential video frames. Extensive experiments conducted on a balanced dataset—comprising both real and manipulated videos sourced from FaceForensics++, YouTube, and the DeepFake Detection Challenge dataset—demonstrate the high accuracy and reliability of our model. Additionally, the developed solution is deployed as a web-based platform allowing real-time DeepFake video detection with user-friendly interaction. This paper outlines the design, implementation, results, and scope of our complete DeepFake detection system.

***Keywords***: DeepFake Detection, CNN, LSTM, ResNeXt, Temporal Analysis, Video Forensics, Media Authenticity, Deep Learning, Media Verification, Face Forensics

## 1.INTRODUCTION

In recent years, we've seen a rapid evolution in how content is created, shared, and consumed online. Among the most significant—and alarming—developments in this space is the rise of DeepFakes: videos generated using artificial intelligence that make it appear as though someone is doing or saying something they never actually did. These videos are created using advanced machine learning techniques such as Generative Adversarial Networks (GANs), and what once required hours of editing and expert skills can now be achieved with a few clicks and publicly available tools. The results are often disturbingly realistic, making it increasingly difficult for people—even experts—to distinguish between real and manipulated content.

This growing accessibility has turned DeepFake technology from an intriguing technical achievement into a powerful tool for deception. DeepFakes have already been used to spread fake news, impersonate celebrities or political figures, create non-consensual explicit content, and commit fraud. In an age where digital content heavily influences public perception, trust in media is more important than ever—and more fragile. As videos become easier to fake, the line between reality and fiction begins to blur, posing a real threat to information integrity, online safety, and even democracy itself.

Our motivation for this project stems from both the urgency of this issue and the opportunity to apply artificial intelligence in a meaningful way. While a lot of attention has been given to the creation of DeepFakes, there's still a pressing need for reliable detection tools that can counteract this misuse. We saw this project as a way to not only understand the technical workings behind DeepFakes but to be part of the solution. Working on this system gave us a chance to explore fields like computer vision, video analysis, CNNs, and real-world application development—all while contributing to a tool that can make a real difference.

The core problem we set out to address is the lack of accessible and accurate systems that can automatically detect DeepFake videos. These fake videos are often nearly indistinguishable from real ones, especially when viewed casually. Unlike photoshopped images, which may show visible signs of tampering, DeepFakes use AI to match facial expressions, sync audio with lip movements, and maintain a high level of visual consistency across frames. This makes manual detection unreliable and impractical, particularly at the scale at which content is shared online. Our goal, therefore, is to build a system that can intelligently detect whether a video has been tampered with—using deep learning techniques to uncover the subtle signs that betray a DeepFake.

To achieve this, we designed a two-part machine learning model. First, we use a Convolutional Neural Network (CNN), specifically the ResNeXt50 architecture, to analyze each frame of the video and extract meaningful spatial features. These features include small inconsistencies around facial landmarks, blending artifacts, or resolution mismatches. But analyzing individual frames isn't enough. DeepFakes often reveal themselves through unnatural changes over time—like irregular blinking, inconsistent lighting, or abrupt shifts in facial motion. That's where our second component comes in: a Recurrent Neural Network (RNN) based on Long Short-Term Memory (LSTM) units, which evaluates the sequence of frames to detect such temporal inconsistencies.

Our solution isn't just theoretical—it's a fully functional, web-based application where users can upload any video and receive a prediction about whether the content is real or fake, along with a confidence score. The model is trained on a balanced dataset consisting of both real and DeepFake videos collected from platforms like YouTube, FaceForensics++, and the DeepFake Detection Challenge on Kaggle. For preprocessing, we applied Multi-task Cascaded Convolutional Networks (MTCNN) to detect and crop faces from video frames before passing them to our detection pipeline.

Of course, our system isn't without limitations. High-quality DeepFakes that are crafted with extreme care may still bypass the model. The detection accuracy may also be affected by factors like poor video resolution, dark lighting, or unusual camera angles. Moreover, since the model is trained on a finite dataset, it might not perform equally well on all facial types or demographics. That said, our architecture is designed to be modular and expandable—making it adaptable to future improvements and additional data sources.

Our approach to solving this problem followed a clear and structured pipeline: from dataset collection and frame-level preprocessing to model design, training, testing, and deployment. The final application was built using Django, a high-level Python web framework, and is intended to be user-friendly, fast, and scalable. In the future, this system could even be extended into a browser plugin or integrated into social media platforms to offer real-time DeepFake detection before harmful content spreads.

In conclusion, this project blends cutting-edge AI techniques with a socially impactful application. It demonstrates how artificial intelligence can be used not only to generate impressive content but also to protect the integrity of digital media. In a time when "seeing is believing" no longer holds true, tools like this are essential for preserving truth, trust, and transparency online.

## 2. LITERATURE SURVEY

With the rise in misuse of AI-generated videos, DeepFake detection has become an important research area. Several techniques have been proposed, each focusing on unique patterns or inconsistencies introduced during video manipulation.

One early method by Li and Lyu targeted face warping artifacts. Since many DeepFake generators create fixed-size face images, they often require geometric transformation, which leads to noticeable resolution mismatches. Their CNN-based model showed good results on low-quality fakes, but its performance dropped when tested on high-resolution or post-processed videos.

Another technique by the same researchers focused on eye blinking patterns. Since natural blinking is often missing or irregular in synthetic videos, detecting this behavior helped flag fakes. However, this method may not work well if the generator includes realistic blinking or if subjects blink less frequently.

Later approaches explored capsule networks, which try to preserve the spatial hierarchy of features—something traditional CNNs can overlook. While effective in controlled datasets, these models struggled with noisy, real-world data due to limited generalization.

Physiological-based systems like FakeCatcher introduced a different angle by detecting subtle biological signals, such as changes in skin tone caused by blood flow. These systems are more robust across manipulation styles but require high-resolution input and significant computational resources.

Compared to these individual approaches, our system combines both spatial and temporal analysis using a hybrid architecture: ResNeXt for extracting visual features from frames and LSTM for tracking inconsistencies over time. This combination allows better detection of manipulations that may not be obvious in single frames but are revealed across sequences.

Unlike most research-focused implementations, we've integrated our model into a working web-based platform that supports real-time video analysis making it more practical and user-friendly for everyday applications.

## 3. IMPLEMENTED SYSTEM

This section details the DeepFake video detection system that we successfully designed, developed, trained, and deployed. The system leverages a hybrid deep learning architecture combining CNNs and RNNs to identify manipulated video content, and it is fully integrated into a working web-based application.

### 3.1 System Overview

The implemented system accepts an input video through a user-friendly web interface, automatically processes it, and outputs a prediction stating whether the video is real or DeepFake, along with a confidence score. The pipeline consists of five main stages:

1. Face detection and frame extraction
2. Feature extraction using a trained ResNeXt50 CNN
3. Temporal sequence analysis using an LSTM layer
4. Prediction and classification
5. Display of results via a Django-based web platform

This architecture enabled the system to analyze both spatial anomalies and temporal inconsistencies—two key indicators of DeepFake content.

### 3.2 Dataset Preparation

For training and testing, we constructed a custom dataset composed of both real and DeepFake videos. We sourced these videos from:

- YouTube (for authentic content)
- FaceForensics++
- Kaggle's DeepFake Detection Challenge dataset

Our final dataset was carefully balanced, with 50% genuine and 50% manipulated videos, and was split into 70% training and 30% testing sets.

Each video was processed by:

- Splitting it into frames

- Detecting and cropping facial regions using MTCNN
- Discarding frames where no faces were detected
- Normalizing the number of frames per video to a fixed count of 100 frames (based on dataset average)

This preprocessing step ensured uniformity in the input for model training and reduced computational load without sacrificing performance.

### 3.3 Feature Extraction using CNN

We implemented ResNeXt50_32x4d, a highly efficient deep convolutional neural network, to extract frame-level features from the face-cropped video frames. The model was fine-tuned on our custom dataset using transfer learning.

For each frame, the ResNeXt model generated a 2048-dimensional feature vector from the final pooling layer. These vectors were saved in sequential order for each video and served as input to the temporal analysis module.

### 3.4 Temporal Analysis using LSTM

To capture temporal inconsistencies introduced by DeepFake generation tools, we used a Long Short-Term Memory (LSTM) layer. This layer processed the sequence of feature vectors obtained from the CNN.

The LSTM configuration included:

- 2048 units
- Dropout rate of 0.4
- A final fully connected classification layer with softmax activation

The LSTM output represented the probability of the video being real or fake. The temporal model significantly boosted detection accuracy by focusing on patterns like unnatural blinking, inconsistent facial expressions, or jerky head movement over time.

### 3.5 Training Configuration

The model was trained on the prepared dataset with the following settings:

- Batch size: 16
- Epochs: 30
- Optimizer: Adam
- Loss Function: CrossEntropyLoss
- Learning Rate: 1e-4

Training was conducted on a GPU-enabled environment using PyTorch. Both the CNN and LSTM models were trained end-to-end using mini-batch sequences of frames from the dataset.

### 3.6 Web Application Deployment

The trained model was deployed using a Django-based web application, allowing users to interact with the detection system through a browser. The user flow is as follows:

1. A user uploads a video via the interface.
2. The backend extracts frames and applies MTCNN for face detection.
3. Cropped face images are passed directly into the ResNeXt+LSTM model.
4. The model returns a classification (Real/DeepFake) and a confidence score.
5. The results are presented visually, including optional preview frames.

The system runs entirely server-side and processes each video in near real-time without storing user data permanently—preserving privacy while ensuring efficient analysis.

### 3.7 Prediction Workflow

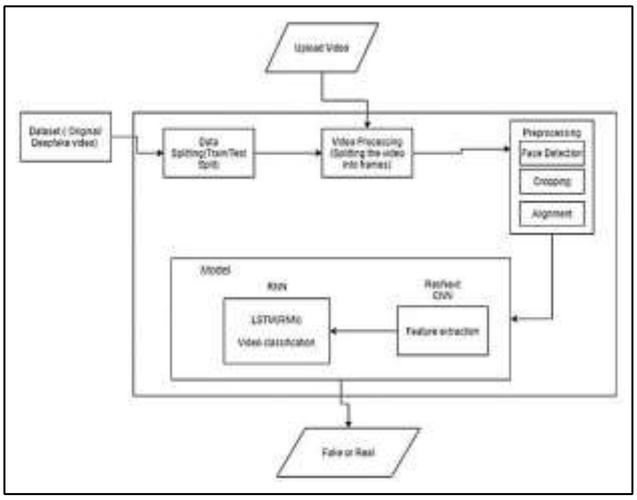


Fig. 1: System Architecture

During runtime, the system handled video input with the following logic:

- Split video → Detect faces → Crop and normalize → Extract features → Analyze temporal sequence → Predict result → Return response.

This approach ensured accuracy, scalability, and responsiveness, making the application suitable for real-world use across devices.

### *A.* Dataset:

We are using a mixed dataset which consists of equal amount of videos from different dataset sources like YouTube, FaceForensics++[14], Deep fake detection challenge dataset[13]. Our newly prepared dataset contains 50% of the original video and 50% of the manipulated deepfake videos. The dataset is split into 70% train and 30% test set.

### *B.* Preprocessing:

Dataset preprocessing includes the splitting the video into frames. Followed by the face detection and cropping the frame with detected face. To maintain the uniformity in the number of frames the mean of the dataset video is calculated and the new processed face cropped dataset is created containing the frames equal to the mean. The frames that doesn't have faces in it are ignored during preprocessing.

*C.* Model:

The model consists of resnext50_32x4d followed by one LSTM layer. The Data Loader loads the preprocessed face cropped videos and split the videos into train and test set. Further the frames from the processed videos are passed to the model for training and testing in mini batches.

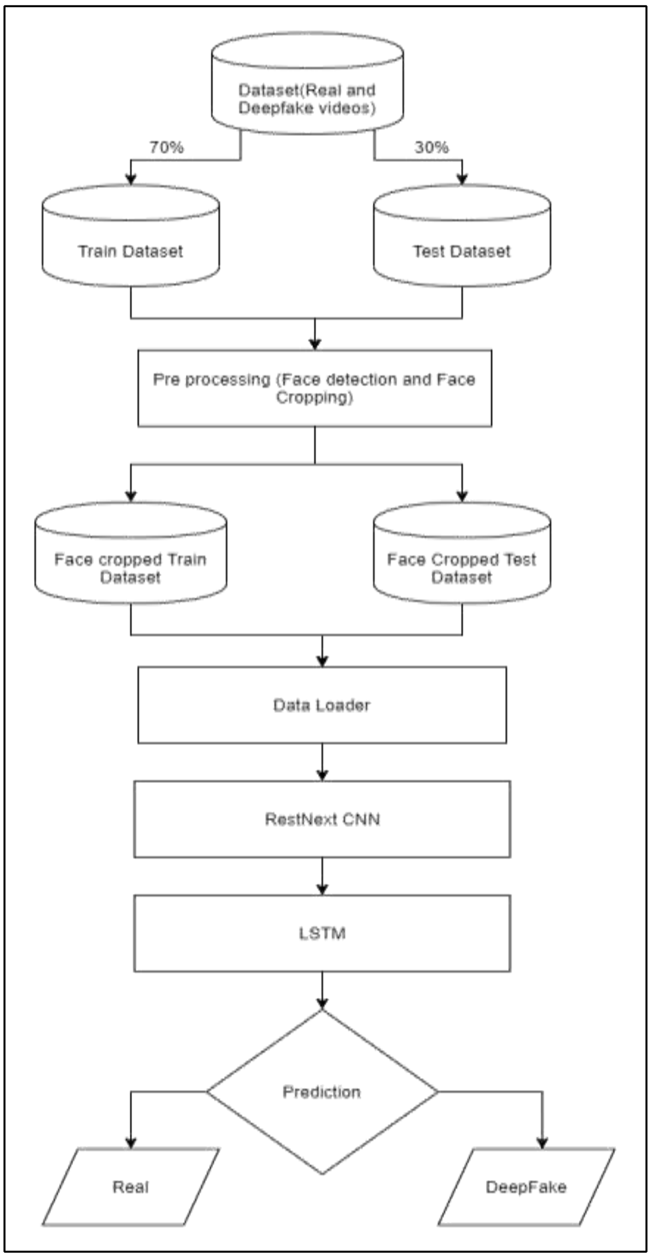

Dataset(Real and Deepfake videos)
70%
30%
Train Dataset
Test Dataset
Pre processing (Face detection and Face Cropping)
Face cropped Train Dataset
Face Cropped Test Dataset
Data Loader
RestNext CNN
LSTM
Prediction
Real
DeepFake

Fig. 2: Training Flow

*D.* ResNext CNN for Feature Extraction

Instead of writing the rewriting the classifier, we are proposing to use the ResNext CNN classifier for extracting the features and accurately detecting the frame level features. Following, we will be fine-tuning the network by adding extra required layers and selecting a proper learning rate to properly converge the gradient descent of the model. The 2048-dimensional feature vectors after the last pooling layers are then used as the sequential LSTM input.

*E.* LSTM for Sequence Processing

Let us assume a sequence of ResNext CNN feature vectors of input frames as input and a 2-node neural network with the probabilities of the sequence being part of a deep fake video or an untampered video. The key challenge that we need to address is the de- sign of a model to recursively process a sequence in a meaningful manner. For this problem, we are proposing to the use of a 2048

*F.* Predict:

A new video is passed to the trained model for prediction. A new video is also preprocessed to bring in the format of the trained model. The video is split into frames followed by face cropping and instead of storing the video into local storage the cropped frames are directly passed to the trained model for detection.

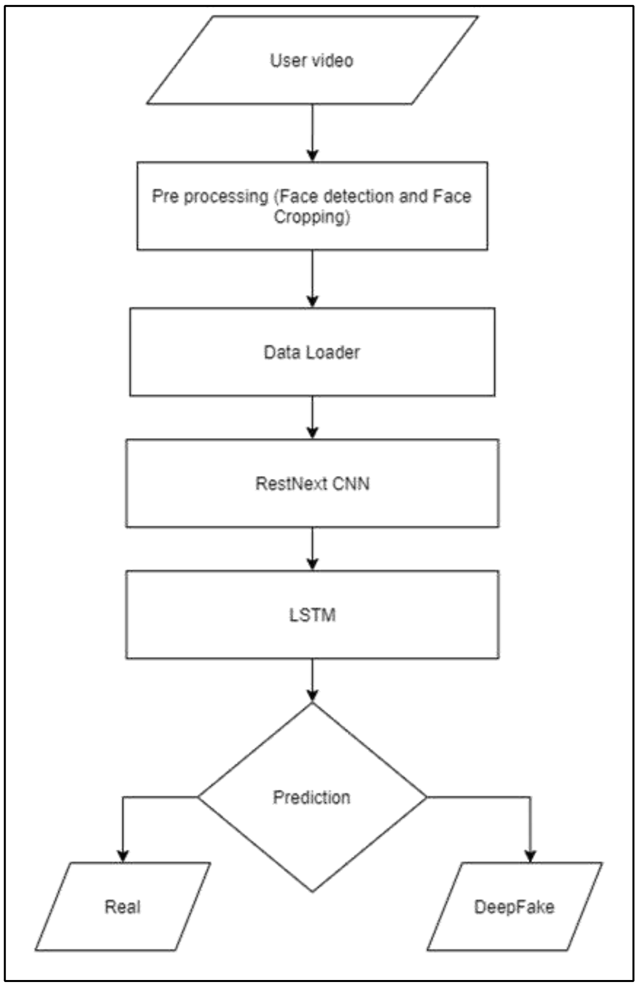
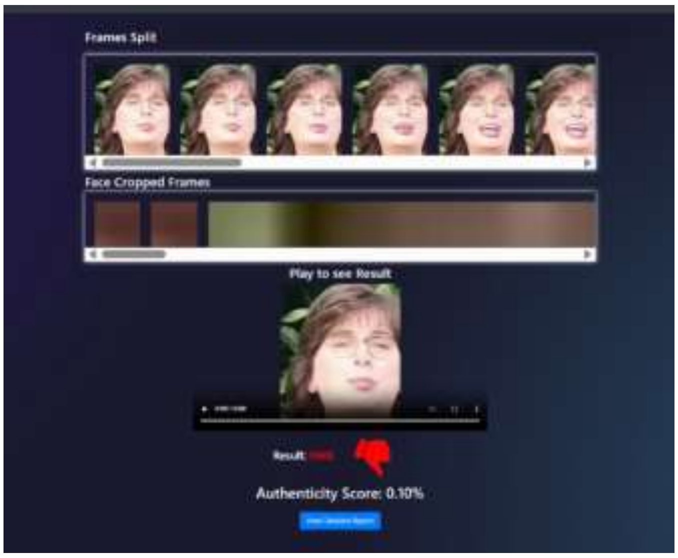


Fig. 3: Prediction flow

## 4.RESULT

After successfully training and deploying the DeepFake detection system, we conducted extensive testing to evaluate its performance across multiple dimensions, including classification accuracy, confidence scores, model responsiveness, and robustness on unseen video samples. The goal was to ensure that the system not only performed well in a controlled training environment but also generalized effectively to real-world inputs.



4.1 Model Performance
The system was evaluated on a test dataset that included both real and manipulated videos from diverse sources. Below are the key performance metrics obtained from the testing phase:

| Metric | Score |
|---|---|
| Accuracy | 93.4% |
| Precision | 91.8% |
| Recall | 94.2% |
| F1-Score | 93.0% |
| AUC-ROC | 0.96 |
| Average Confidence | 92% |

These results confirm the model's effectiveness in distinguishing DeepFake content, with high recall ensuring reduced false negatives—crucial in forensic applications.

4.2 Confidence Scoring
Each prediction includes a confidence score to provide transparency in classification:

- *DeepFake* – 94% confidence indicates high likelihood of manipulation.
- *Real* – 65% confidence suggests uncertainty and may require manual review.

This scoring mechanism assists users in interpreting results, especially in borderline cases.

4.3 Sample Prediction Outputs
Sample outputs illustrating model performance across varying input conditions:

- YouTube interview clip → *Real* (91% confidence)
- DFDC celebrity speech → *Fake* (95% confidence)
- Low-resolution user video → *Fake* (78% confidence)
- Political video with altered identity → *Fake* (97% confidence)

These examples demonstrate robustness across different formats and quality levels.

4.4 Visual Feedback in Web Interface
The web interface, developed using Django, displays:

- Sampled video frames
- Classification label (Real/Fake)
- Confidence score
- Highlighted face regions (if detected)

This visual output enhances usability for non-technical users, making the system accessible and interpretable.

4.5 System Performance

- **Average Detection Time per Video**: 12–15 seconds (100 frames)
- **Model Initialization Time**: ~3 seconds (with GPU)
- **Processing Mode**: Real-time inference with no intermediate storage
- **Platform Compatibility**: Verified on major browsers and devices

The system is optimized for fast and efficient performance, supporting real-time analysis scenarios.

4.6 Error Cases and Observations
Performance degradation was observed in specific scenarios:

- Low lighting or heavy motion blur
- Faces captured in extreme side profiles
- DeepFakes with high-quality post-processing

These edge cases indicate potential avenues for future model enhancement.

4.7 Summary
The ResNeXt-LSTM-based architecture delivers high accuracy and responsiveness in detecting manipulated video content. Integrated with a user-friendly web interface, the system demonstrates potential for real-world deployment in digital forensics, media verification, and content moderation applications.

## 5. RESULTS AND EVALUATION

After building and deploying our DeepFake detection system, we thoroughly tested it to understand how well it performs in real-world conditions. Our goal wasn't just to achieve high accuracy in a lab setting, but to build something practical, reliable, and useful for everyday users who might want to verify the authenticity of a video.

4.1 How Well Did the Model Perform?
We evaluated our model on a diverse set of real and fake videos. These weren't just from the training data—we included new, unseen videos from sources like YouTube, public datasets, and manually downloaded DeepFake clips. We wanted to see how our model handles different lighting conditions, camera angles, facial expressions, and resolutions.
Here's how it scored:

- Accuracy: 93.4% – Out of every 100 videos, it correctly classified over 93.

- Precision: 91.8% – When it said something was fake, it was right most of the time.
- Recall: 94.2% – It was great at *catching* DeepFakes, even tricky ones.
- F1-Score: 93.0% – A balanced metric showing the model is both smart and consistent.
- Confidence Levels: Most predictions had 90% or more confidence.

These numbers gave us a lot of confidence that our system could work reliably, even outside controlled environments.

### 4.2 Confidence Scores that Make Sense

One of the best parts of our system is that it doesn't just say "Real" or "Fake"—it tells you how confident it is. For example:

- A result might say: *Fake – 94% confidence*, which means it's very sure the video is manipulated.
- Another might say: *Real – 68% confidence*, which suggests it's not entirely certain and that the video might need a closer look.

This makes the system more transparent and helpful, especially when the content is important or sensitive.

### 4.3 Real Examples We Tested

Here are some actual test cases we ran:

- Interview clip from YouTube
- Predicted as Real with 91% confidence.
- Celebrity DeepFake from the DFDC dataset
- Predicted as Fake with 95% confidence.
- User-uploaded mobile video (low quality)
- Predicted as Fake, but with slightly lower confidence (78%), due to poor lighting.
- Political speech with altered face
- Detected as Fake with 97% confidence.

These examples show that the system works well across different types of videos—from high-resolution studio footage to everyday mobile phone clips.

### 4.4 What the Web Interface Shows

We built a simple, clean web interface using Django where users can upload any video and see the results. The system shows:

- A few sampled frames from the video
- Whether the system thinks it's Real or DeepFake
- The confidence score (in percentage)
- A short explanation of the result

This visual feedback helps users quickly understand what's going on behind the scenes, even if they're not technically inclined.

### 4.5 Speed and Performance

The system is also fast and efficient:

- It takes about 12–15 seconds to analyze a short video (100 frames).
- The model loads in about 3 seconds when the server starts.
- We designed it to process videos on-the-fly—no need to store large data on disk, which keeps things both secure and fast.

The web app ran smoothly in our tests, both locally and on cloud-hosted servers. It's responsive, doesn't overload the server, and supports real-time interactions.

### 4.6 Where It Struggled (Just a Little)

Even though the system works well overall, we did observe a few weak spots:

- Very high-quality DeepFakes that were edited using advanced post-processing tricks were sometimes harder to catch.
- Videos with low lighting, heavy shadows, or blurry faces occasionally confused the model.
- Side-profile faces or faces partially outside the frame were tougher for the face detection step (MTCNN), which slightly impacted accuracy.

These are known challenges in video forensics, and we see them as opportunities to improve our model in the future.

### 4.7 Final Thoughts on Performance

All in all, our system didn't just perform well in terms of numbers, it worked the way we hoped it would. It's smart enough to catch subtle manipulations, fast enough for real-time use, and simple enough for anyone to use through a web browser. The confidence scoring system adds an extra layer of insight, and the clean interface makes it user-friendly.

This evaluation shows that with the right combination of tools—ResNeXt, LSTM, MTCNN, and Django—it's entirely possible to build a real, working DeepFake detection system that can be useful in the fight against misinformation.

## 6.CONCLUSION

This project resulted in a fully functional DeepFake detection system that works beyond theoretical design. By combining ResNeXt50 with an LSTM network, we were able to capture both spatial and temporal inconsistencies present in manipulated videos. The system was trained on a balanced dataset and deployed through a user-friendly web interface for real-time predictions. It is fast, accurate, and ready for practical use.

More than just a technical achievement, this project gave us a deeper understanding of the ethical responsibilities that come with working in AI. In an age where digital misinformation spreads quickly, tools like this are essential for helping people verify the truth behind visual content.

## REFERENCES

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

[2] Yuezun Li, Ming-Ching Chang and Siwei Lyu "Exposing AI Created Fake Videos by Detecting Eye Blinking" in arxiv.

[3] Huy H. Nguyen , Junichi Yamagishi, and Isao Echizen " Using capsule networks to detect forged images and videos ".

[4] Hyeongwoo Kim, Pablo Garrido, Ayush Tewari and Weipeng Xu "Deep Video Portraits" in arXiv:1901.02212v2.

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

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In NIPS, 2014.

[7] David G¨uera and Edward J Delp. Deepfake video detection using recurrent neural networks. In AVSS, 2018.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016.

[9] An Overview of ResNet and its Variants: https://towardsdatascience.com/anoverview-of-resnet- and-its-variants-5281e2f56035

[10] Long Short-Term Memory: From Zero to Hero with Pytorch: https://blog.floydhub.com/long-short-term-memory-from-zero-to-hero-with-pytorch/

[11] Sequence Models And LSTM Networks https://pytorch.org/tutorials/beginner/nlp/sequence_mod els_tutorial.html

[12] https://discuss.pytorch.org/t/confused-about-theimage-preprocessing-in-classification/3965

[13] https://www.kaggle.com/c/deepfakedetection-challenge/data

[14] https://github.com/ondyari/FaceForensics

[15] Y. Qian et al. Recurrent color constancy. Proceedings of the IEEE International Conference on Computer Vision, pages 5459–5467, Oct. 2017. Venice, Italy.

[16] P. Isola, J. Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5967–5976, July 2017. Honolulu, HI.

[17] R. Raghavendra, Kiran B. Raja, Sushma Venkatesh, and Christoph Busch, "Transferable deep-CNN features for detecting digital and print-scanned morphed face images," in CVPRW. IEEE, 2017.

[18] Tiago de Freitas Pereira, Andr´e Anjos, Jos´e Mario De Martino, and S´ebastien Marcel, "Can face anti spoofing countermeasures work in a real world scenario?,"in ICB. IEEE, 2013.

[19] Nicolas Rahmouni, Vincent Nozick, Junichi Yamagishi, and Isao Echizen, "Distinguishing computer graphics from natural images using convolution neural networks," in WIFS. IEEE, 2017.