Enhancing Robustness Against Manipulated Videos With ResNext50 and LSTM

Lai Le Dinh Duc[1][0009−0009−5584−575X] and Le Minh Khoi[2][0009−0009−3201−9519]

FPT University, Ho Chi Minh , Viet Nam
lailedinhduc@gmail.com
khoile5464@gmail.com

Abstract. Deepfake detection in videos has become a crucial challenge due to the increasing sophistication of synthetic media. In this study, we significantly improved manipulated face detection performance by adopting an integrated approach that leverages ResNext50 and Long Short-Term Memory (LSTM) networks. ResNext50, with its group convolution architecture, captures diverse and discriminative spatial features more effectively than conventional ResNet models. Meanwhile, LSTM leverages sequential frame dependencies, allowing the model to analyze temporal consistency rather than relying solely on static images, as in early methods. The architecture was trained on a well-balanced dataset comprising 6,451 videos, randomly selected and combined from three datasets: DFDC, Celeb, and FaceForensics++. The model finally achieved an accuracy of 89.66% and an F1-score of 87% These results demonstrate the effectiveness of our approach in identifying deepfake content by utilizing both spatial and temporal characteristics.

Keywords: DeepFakeDetection · ResNext50 · LSTM

1 Introduction

1.1 Context

In recent years, deepfake technology has emerged as a double-edged sword in the digital world. Initially developed as a tool for entertainment, film production, and accessibility, its rapid advancement has raised significant ethical and security concerns. Deepfake videos, created using deep learning techniques such as Generative Adversarial Networks (GANs) [3] and, more recently, diffusion models [19], [20] can fabricate highly realistic depictions of individuals, making it increasingly difficult to distinguish between authentic and manipulated content. This has led to a surge in misinformation, identity theft, and political deception, with severe consequences for public trust and online security. Governments, social media platforms, and cybersecurity experts are now racing to develop effective countermeasures to mitigate the risks posed by this technology.

As deepfake algorithms become more sophisticated, traditional detection methods struggle to keep pace. Early detection models primarily relied on analyzing individual frames, treating them as independent images without considering temporal consistency. These methods lacked the ability to detect sequential inconsistencies, making them ineffective against advanced deepfake models that generate highly coherent and temporally consistent videos. The challenge lies in identifying subtle distortions in facial dynamics and motion, which are imperceptible to static image-based analysis but can be detected through sequence-aware learning approaches.

1.2 Problem Statement And Application Potential

To address this growing threat, our solution applied a hybrid deep learning model that combines ResNext50 [1] and Long Short-Term Memory (LSTM) [2] networks to capture both spatial and temporal features. ResNext50 extracts fine-grained facial features from individual frames, identifying subtle pixel-level anomalies that may indicate tampered content. Meanwhile, the LSTM component models temporal dependencies across frames, enabling the detection of inconsistencies in motion and expression. To enhance model robustness, the project integrated data from multiple datasets, including Celeb-DF [17], DFDC [18], and FaceForensics++ [12], where fake videos are generated using techniques rooted in GANs, NeuralTextures, DeepFake AutoEncoder, and Neural Talking Heads (NTH), and related methods.

The potential applications of this deepfake detection system are vast. In cybersecurity and fraud prevention, it can mitigate financial scams and identity theft caused by manipulated videos. In media and politics, it can verify the authenticity of news content, reducing the spread of misinformation. For digital content verification, platforms like Facebook, YouTube, and TikTok can integrate the system to filter deceptive videos, enhancing trust in online media. By combining spatial-temporal learning, this approach enhances the robustness of deepfake detection, making it a vital tool for digital security.

1.3 Challenges

Although the hybrid approach to deepfake detection indicates promising potential, several critical challenges must be overcome to ensure robustness and reliability. As AI-generated content becomes more sophisticated, and diversity of techniques used to create deepfakes, particularly those involving Generative Adversarial Networks (GANs). These techniques include face swapping, expression manipulation, and lip-syncing, each producing highly realistic and varied outputs. The diversity in deepfake generation methods makes it difficult for detection models to generalize across different types of manipulations. Additionally, deepfake videos often exhibit subtle temporal inconsistencies, such as unnatural facial movements or irregular lip-sync patterns, which are harder to detect in static images. Detecting these anomalies requires models to analyze temporal

dynamics and spatial features simultaneously, adding complexity to the task.

Another challenge lies in the computational resources. While leveraging ResNeXt50, a powerful convolutional neural network (CNN), to extract spatial features from input data shows promise, hardware limitations pose a significant barrier. Training ResNeXt50 from scratch demands substantial computational resources, including high-end GPUs and extended training time, which are often inaccessible in resource-constrained environments.

Furthermore, the hybrid architecture's complexity is particularly prone to overfitting. ResNext-50, with a depth of 50 layers, enables the extraction of fine-grained features but risks memorizing training data rather than generalizing if trained on an insufficiently large and diverse dataset. Meanwhile, LSTM, when used with a hidden state size greater than 1024, can model long-term dependencies but is prone to overfitting specific sequences. The combination of both forms a powerful system. However, it may perform well on familiar data while struggling with unseen data if hyperparameter selection and setup are not carefully optimized.

1.4 Summary and Objectives

Our method leverages an advanced approach to deepfake detection by applying a hybrid deep learning model combining ResNext50 [1] and LSTM [2]. The process begins with data preprocessing using the *OpenCV* library to resize and standardize the color of each video frame, reducing noise and optimizing computational efficiency. This step ensures that the model effectively extracts relevant features.

The model architecture, implemented in *PyTorch*, integrates ResNext50—pretrained on ImageNet via transfer learning—with an LSTM layer to capture temporal dependencies. Specifically, the last two layers of ResNext50 are removed and replaced with an average pooling layer, followed by an LSTM and a final linear layer to produce classification outputs. To improve generalization and prevent overfitting, training incorporates dropout, early stopping, and model checkpointing, optimizing resource utilization and facilitating evaluation.

1.5 Results and Achievements

The hybrid ResNext50 and LSTM model demonstrated impressive performance in deepfake detection, achieving an accuracy of 89.66% on a video dataset with an LSTM sequence length of 80. This result highlights the model's capability to effectively identify manipulated video content by leveraging both spatial and temporal features. However, this performance comes with a substantial model

size of 226.6MB and 56.5 million parameters, making it computationally demanding. Although the model can currently run on a CPU without requiring a GPU, a minimum hardware requirement of an RTX 3070ti GPU is necessaryto ensure that the prediction time matches the input video's duration, reflecting the model's resource-intensive nature. Despite these requirements, the training process proved efficient, taking only 77.06 minutes to reach the reported accuracy using an RTX 3090 (24GB) GPU. This balance of high accuracy and reasonable training time underscores the model's potential for practical applications. Nevertheless, its large size limits deployment on resource-constrained devices such as mobile phones or computers without dedicated GPUs. To address this, future research will focus on reducing the model's footprint through optimization techniques like pruning or quantization, aiming to enhance its accessibility and utility across a wider range of platforms in real-world deepfake detection scenarios.

2 Related Work

Over the past few decades, numerous techniques [4], [5], [6] have emerged for generating highly realistic synthetic faces . Early approaches for detection, such as frequency domain analysis via Discrete Fourier Transform (DFT) [10] and shallow CNNs like MesoNet [11], focused on identifying low-level artifacts or mesoscopic facial inconsistencies. However, MesoNet processes each frame independently using CNNs, and DFT analyzes only the frequency components of each single frame independently; neither approach considers temporal relationships between frames. Although these frame-based methods were effective against early deepfakes, they faced challenges with modern GAN-generated forgeries (e.g., StyleGAN) [7], [8], [9], which reduce anomalies in the frequency domain and facial artifacts. This highlights the need for models that balance spatial feature extraction with temporal dynamics.

Subsequent works from 2020 onward emphasized temporal or multimodal analysis. For instance, self-supervised methods [13] leveraged real-world talking faces to extract temporal features, while AV-Lip-Sync+ [14] exploited audio-visual inconsistencies. To address the growing need for capturing the temporal evolution of artifacts over time, we implemented a hybrid CNN-LSTM architecture. Similar architectures in other domains, such as speech emotion recognition [15] and malware detection [16], demonstrate the versatility of CNN-LSTM frameworks in handling sequential data. Specifically, in our deepfake detection solution, ResNext50—a variant of ResNet50 with grouped convolutions—handles spatial feature learning through multi-branch transformations, overcoming the limited depth of earlier CNNs [11], while the LSTM captures subtle temporal artifacts across frames, such as inconsistent eye blinking or unnatural head movements. Finally, the model is trained and evaluated using a combined dataset sampling from Celeb-DF V2 [17], Forensics++ [12] and DFDC [18].

3 Method

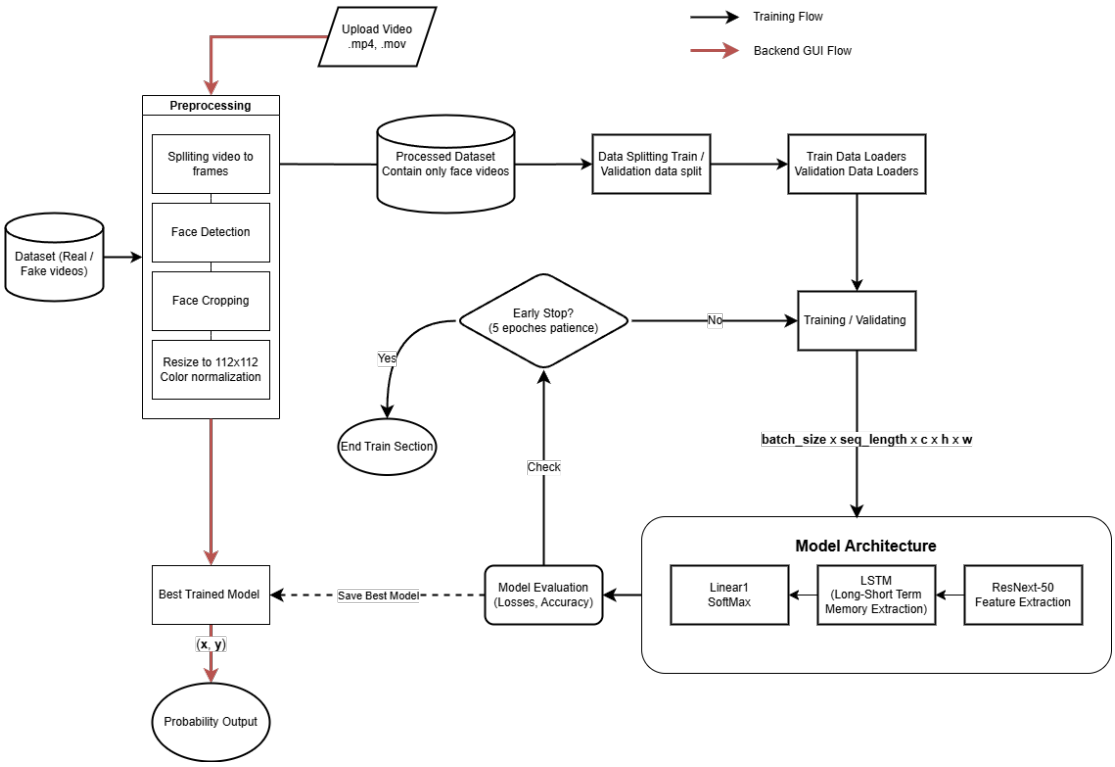

Fig. 1. Overall Flowchart

3.1 Overview

In this study, two models, namely ResNext50 [1] and LSTM [2], were employed. ResNext50 is an improved version of ResNet50, incorporating grouped convolutions to enhance feature extraction efficiency, whereas LSTM is a type of recurrent neural network (RNN) specialized in learning long-term dependencies in sequential data.

3.2 Input and Output

The input of the model is a tensor with the shape:

$$(\text{batch\_size}, \text{seq\_length}, c, h, w)$$

where batch_size is the number of videos processed simultaneously, seq_length is the sequence length, c represents the number of color channels, h is the height, and w is the width. Specifically, the input adopts a shape of

$$(\text{batch\_size}, \text{seq\_length}, 3, 112, 112),$$

where each frame has dimensions of 112×112 in the RGB color space. Subsequently, the input is reshaped to dimensions $(p, 3, 112, 112)$ using the *view* function, where $p = \text{batch\_size} \times \text{seq\_length}$. This transformation is necessary because the ResNext50 architecture exclusively accepts inputs formatted as sequential data.

The output of the model is a probability distribution for two classes: fake and real, represented as:

$$(x, y)$$

where x is the probability of the fake class, and y is the probability of the real class.

3.3 ReLSTM Method (The combination of ResNext-50 and LSTM)

Figure 1 (Overall Flowchart) provides a comprehensive overview of the entire method. The diagram illustrates two main workflows of the project: the training pipeline and the practical prediction pipeline within the web-based GUI. Both workflows follow the same initial steps, involving video input ingestion and preprocessing. However, the preprocessed frames are ultimately assembled into an .mp4 video and a tensor stack, which serves as input for model prediction. Alternatively, these frames can be stored as a preprocessed dataset, which forms the primary data source for training and validation.

In the next step of training pipeline, preprocessed face-cropped videos is first split into training and validation sets before being loaded into *DataLoader* with an appropriate batch size. Subsequently, the frames extracted from the preprocessed videos are fed into the model for both training and testing purposes. The following sections will provide a detailed discussion of two key blocks: preprocessing and model architecture.

Preprocessing Block , in Figure 1 above, plays an essential role in preparing raw video data for efficient and accurate analysis by the detection model. In this block, the processing pipeline is manipulated using *Opencv* [23], and *Tranforms* class of *Torchvision* [24] libraries to ensure high-quality input for model. The initial step involves decomposing the video into individual frames, which provides the fundamental units for subsequent processing. Following this, the face cropping operation is performed by leveraging the *face_recognition* [26] library to identify and extract the region of interest, namely the face, from each frame.

This targeted approach minimizes background interference and focuses computational resources on the most pertinent features. Moreover, each extracted frame is resized to 112×112 pixels, ensuring a consistent spatial resolution across the dataset. The process further incorporates color normalization by applying mean values of $[0.485, 0.456, 0.406]$ and standard deviations of $[0.229, 0.224, 0.225]$ - parameters established by the ImageNet dataset and refined through pretraining on ResNext50 - to standardize the pixel intensity distributions, which is critical for achieving reliable feature extraction and convergence during model training.

Fig. 2. Sample Processsed Frame

After these preprocessing steps, the processed frames are either reassembled into a single .mp4 video file for storage or converted into a tensor with dimensions $(1, \text{seq\_length}, c, h, w)$, assuming a batch size of 1 when predicting a single video. Overall, this well-structured preprocessing pipeline not only standardizes the input data but also enhances the detection model's ability to identify subtle deepfake artifacts by ensuring spatial and color consistency, ultimately leading to more accurate and efficient deepfake detection.

Model Architecture Block , presented in Fig 3, consists a series of convolutional layers of ResNeXt as the initial backbone, followed by a single Long Short-Term Memory (LSTM) layer and a fully connected layer for classification.The convolutional network begins with a 7×7 convolutional layer with 64 filters and a stride of 2, followed by a 3×3 max-pooling layer with a stride of 2 to reduce spatial dimensions.

Subsequently, the model employs four convolutional stages, each consisting of multiple bottleneck blocks. Each block follows the structure of a 1×1 convolution for dimensionality reduction, a 3×3 grouped convolution with cardinality C=32 - 32 paths in each block - to capture representations diversely , and a 1×1 convolution to restore the depth. For instance, in the conv2 stage, the bottle-

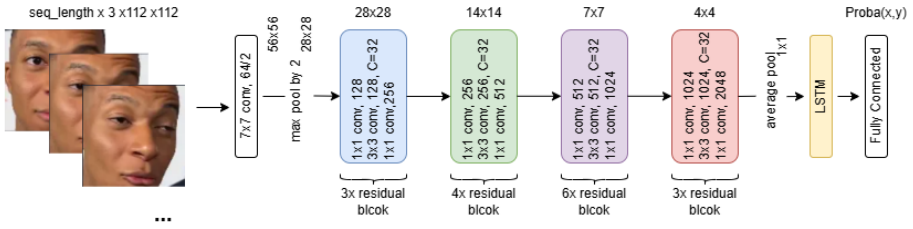

Fig. 3. Visualization From Table 1

neck block is repeated three times with 128 filters in the grouped convolution, and the final output expands to 256 channels. The number of blocks increases progressively in deeper stages, with the fourth stage containing six repetitions and the fifth stage containing three.

After convolutional feature extraction, an adaptive average pooling layer reduces the output to a fixed p × 2048 x 1 x 1 shape. The extracted features are then reshaped back to batch_size x seq_length x 20248 before input to an unidirectional LSTM Layer for capturing sequential dependencies. The final output is obtained by averaging the sequence length and passing it through Linear Layer - a fully connected layer with dropout regularization and a softmax function to compute class probabilities.

3.4 The estimated time and space complexity

The complexity of the deepfake detection task is analyzed in terms of both time and space to assess its computational efficiency.

The **time complexity** is estimated as

$$O(F \times (D \times k^2 \times C + T \times h^2)), \tag{1}$$

where F is the number of video frames, $D = 48$ denotes the number of convolutional layers in the ResNeXt-50 backbone, k is the kernel size (3 or 7), $C = 3$ represents the number of input channels (RGB), T is the sequence length processed by the LSTM, and $h = 2048$ is the number of hidden units in the LSTM. This formulation captures the convolutional operations ($O(D \times k^2 \times C)$ per frame) and the sequential processing of feature vectors by the LSTM ($O(T \times h^2)$), with the total complexity scaling linearly with F.

The **space complexity** is approximated as

$$O(F \times (W' \times H' \times C' + 112^2 + T \times h)), \tag{2}$$

where $(W' \times H' \times C')$ represents the feature map dimensions from ResNeXt-50 ($1 \times 1 \times 2048$ after pooling), 112^2 accounts for the cropped and resized facial

Table 1. ReLSTM Architecture - ResNeXt50 (32×4d) and LSTM.

Stage	Output	Architecture
conv1	$p \times 64 \times 56 \times 56$	7×7, 64, stride 2
maxpool	$p \times 64 \times 28 \times 28$	3×3 max pool, stride 2
conv2	$p \times 256 \times 28 \times 28$	$[1 \times 1, 128]$ $[3 \times 3, 128, C = 32]$ $\times 3$ $[1 \times 1, 256]$
conv3	$p \times 512 \times 14 \times 14$	$[1 \times 1, 256]$ $[3 \times 3, 256, C = 32]$ $\times 4$ $[1 \times 1, 512]$
conv4	$p \times 1024 \times 7 \times 7$	$[1 \times 1, 512]$ $[3 \times 3, 512, C = 32]$ $\times 6$ $[1 \times 1, 1024]$
conv5	$p \times 2048 \times 4 \times 4$	$[1 \times 1, 1024]$ $[3 \times 3, 1024, C = 32]$ $\times 3$ $[1 \times 1, 2048]$
avgpool	$p \times 2048 \times 1 \times 1$ $batch\_size \times seq\_length \times 2048$	AdaptiveAvgPool2d(1)
lstm	$batch\_size \times seq\_length \times 2048$	LSTM(bidirectional = False)
fully connected	$batch\_size \times (x, y)$	Mean of seq_length Linear, Dropout Softmax

region per frame, and $T \times h$ reflects the LSTM's hidden state storage. Although the model leverages a deep architecture, its complexity remains reasonable: the linear dependency on F ensures scalability, while the constrained values of T and reduced feature map sizes keep memory demands moderate.

3.5 Web-based GUI

For practical prediction, we leveraged Gradio to build an interactive web-based interfaces. When a novel video is presented to the trained model for classification, its frames undergo Preprocessing Block to align with the input format required by the trained architecture. Subsequently, the preprocessed frames are fed directly into the model for inference. The website then return results with label, probability, processed time and considered frames.

3.6 Advantages and Disadvantages

After detailing our methodology utilized for deepfake detection in videos, the advantages and limitations of the approach are discussed in this section.

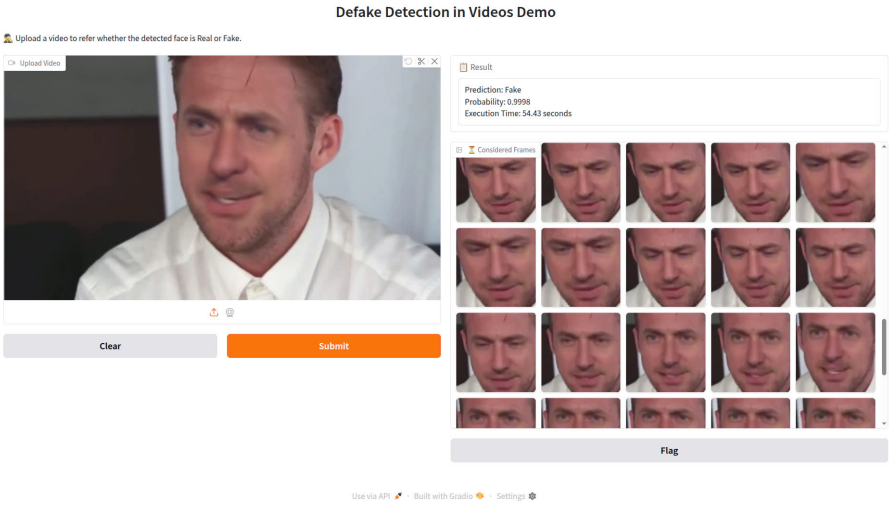

Fig. 4. Sample Test With GUI

One of the primary advantages of the method lies in its utilization of ResNext50 for feature extraction. As an extension of the ResNet architecture, ResNext leverages a grouped convolution strategy within its bottleneck blocks, enhancing representational capability while maintaining computational efficiency compared to conventional ResNet architectures. Furthermore, the integration of LSTM for sequential data processing enables the model to effectively capture long-term dependencies, addressing the vanishing gradient issue prevalent in traditional RNNs. The synergy of ResNext50 and LSTM within this hybrid framework empowers the model to retain and analyze sequential frame dependencies, leading to superior performance over traditional CNN-based methods that rely on static frame classification followed by simple aggregation techniques such as averaging or max pooling. Finally, we ensured the model halts training at the optimal point and maintains generalization ability by adopting Early Stopping with an appropriate patience value and incorporating a Dropout layer before the SoftMax function.

However, certain limitations must be acknowledged. Firstly, the black-box nature of deep learning models limits interpretability, making it difficult to discern the exact features driving classification decisions. Next, with a parameter count of approximately 56.5 million, the model presents deployment challenges on resource-constrained devices such as mobile platforms or environments lacking GPU support, limiting its practical applicability in real-time or edge computing scenarios.

4 Experiments

4.1 Datasets and Experimental Setup

In this study, we leveraged a comprehensive dataset stratified sampling across three widely-used deepfake detection benchmarks: Celeb-DF v2, DFDC, and FaceForensics++ as shown in Fig 5.

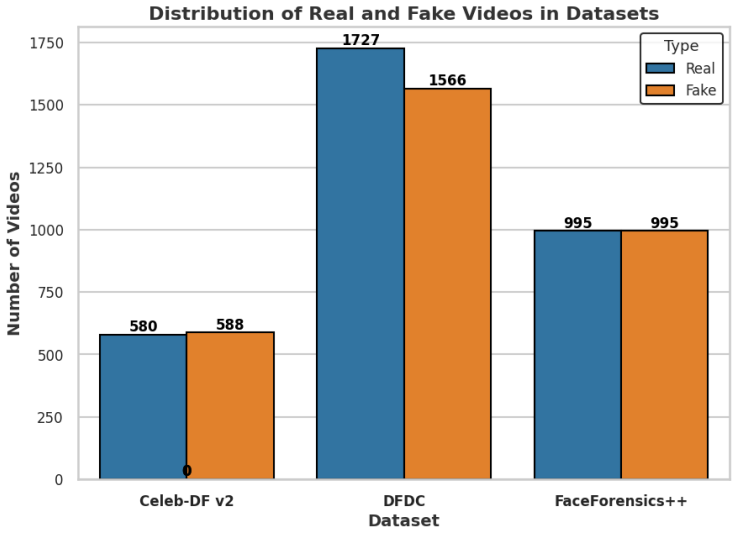

Fig. 5. Distribution of Real and Fake Videos in Datasets

- **Celeb-DF v2 [17]**: Contains 580 real and 588 fake videos created by auto-encoder method. Celeb-DF v2 is chosen for its high-quality deepfakes featuring celebrity faces, which are often the target of malicious deepfake attacks.

- **DFDC [18]**: Comprises 1,566 fake and 1,727 real videos. DFDC is a large-scale dataset released by Facebook AI in collaboration with academic and industry partners to advance deepfake detection research.

- **FaceForensics++ [12]**: Includes balanced 995 real and 995 fake videos. FaceForensics++ have been manipulated with four automated face manipulation methods: Deepfakes, Face2Face, FaceSwap and NeuralTextures, and sourced from 977 youtube videos.

The primary purpose of combining datasets is to enhance the model's robustness and generalization ability by exposing it to a wider range of deepfake generation

techniques and data characteristics. Each dataset (Celeb-DF v2, DFDC, and FaceForensics++) contains unique features and deepfake creation methods, and by integrating them, we aim to create a more comprehensive and diverse training set.

In the next phase, the entire dataset was passed through Preprocessing Block in Figure 1. This step ensured the dataset achieves uniformity across the three color channels (RGB) and standardized the frame dimensions, resulting in a processed dataset that was consistent in both color representation and size. Following this, the dataset was prepared for model training by splitting it into two subsets: training and validation. This division was performed using the train_test_split function from the sklearn library, which guaranteed an accurate distribution of data according to the desired ratio. Specifically, the processed dataset was shuffled and divided into an 80/20 split, with 80% allocated to the training set and 20% to the testing set.

Once the data was split, both the training and testing subsets were passed to the *Dataset* class from the *PyTorch* library [22]. This class extracted the corresponding number of frames based on the specified sequence length and organized the data into a structured format suitable for model input. Finally, the *DataLoader* class was employed to combine the training and testing Dataset objects. The DataLoader facilitated batch-wise data retrieval and enhanced training efficiency by leveraging GPU multi-threading capabilities. This end-to-end process ensured that the data was optimally prepared for training and evaluation, enabling robust model performance.

4.2 Training Configuration

The model architecture leveraged transfer learning with the ResNeXt-50 weights pretrained on ImageNet. We applied a `dropout` regularization (with a probability of 0.4) to prevent overfitting. The experiments (sequence length = 80) were conducted on an NVIDIA RTX 3090 GPU, which has 24GB of VRAM, using PyTorch version `1.12.1`. To balance CPU utilization and memory efficiency, the implementation set `num_workers` to 2 for data loading. Furthermore, different training hyperparameters are presented in Table 2.

Training/Testing Details: Prior to initiating the training process, a checkpoint variable was initialized to systematically store the optimal model weights based on validation performance. During training, the model underwent iterative epochs, each followed by an evaluation phase to prevent test-time learning. Specifically, upon completing each epoch, the model was switched to evaluation mode, and predicted on unseen data. Predictions were evaluated using the Cross Entropy loss function, with the computed loss serving dual purposes. First, it monitored potential overfitting by tracking validation loss trends; if the loss failed to improve for a pre-defined number of epochs (patience), early stopping

Table 2. Training Hyperparameters

Parameter	Value
Learning Rate	1e-5
Batch Size	4
Epochs	20
Optimizer	Adam
Loss Function	CrossEntropyLoss
Early Stopping Patience	5 epochs
Dropout Rate	0.4

was triggered to halt training prematurely. Second, the loss determined whether to update the checkpoint: if the current epoch's validation loss was the lowest observed, the model's weights were saved, ensuring retention of the best-performing iteration. This dual mechanism balanced optimization and generalization, dynamically adapting to training dynamics.

Post-training, the finalized model with the optimal checkpoint was subjected to a comprehensive evaluation protocol. A custom evaluation function, developed by the research team, quantified both primary metrics and secondary metrics to holistically assess performance. Additionally, critical visualizations, including confusion matrices and loss/accuracy plots, were saved. This trainin and testing pipeline ensured rigorous validation, mitigated overfitting risks, and provided actionable insights into model efficacy.

4.3 Evaluation Metrics

The proposed framework employs multiple evaluation metrics, systematically categorized into primary and secondary groups, to holistically assess model performance. The primary metrics—Average Accuracy and Cross-Entropy Loss—are monitored across training epochs to guide optimization decisions. These metrics directly inform early stopping criteria and checkpoint selection, ensuring the model retains optimal generalization. Secondary metrics, including F1-score and AUC-ROC, provide complementary perspectives: the F1-score balances precision and recall to address class-specific performance nuances, and the AUC-ROC evaluates probabilistic discrimination capability across thresholds. This tiered prioritization aligns with the dataset's inherent balanced class distribution, where primary metrics sufficiently reflect global performance trends, while secondary metrics validate robustness against subtle biases.

Primary Metrics

Average Accuracy measures the overall correctness of the model's predictions. It is calculated as the ratio of correctly classified videos to the total number of

videos.

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(\hat{y}_i = y_i) \tag{3}$$

where:

- N - Total number of videos.
- $\hat{y}_i$ - Predicted label of video i.
- y_i - Ground truth class for video i.
- $\mathbb{I}(x)$ - Indicator function, which equals 1 if x is true and 0 otherwise.

Cross-Entropy Loss quantifies the difference between the predicted probability distribution and the true distribution of the classes. A lower cross-entropy loss indicates better model performance.

$$-\frac{1}{N}\sum_{i=1}^{N}[y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)] \tag{4}$$

where:

- N - Total number of videos.
- y_i - True label (1 for fake, 0 for real) for video i.
- $\hat{y}_i$ - Predicted probability of video i being fake.

Secondary Metrics

F1 Score is the harmonic mean of precision and recall, providing a balanced measure of the model's ability to correctly identify both real and fake videos.

$$F1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{5}$$

AUC-ROC - The Area Under the Receiver Operating Characteristic Curve measures the model's ability to distinguish between classes across different thresholds. A higher AUC-ROC indicates better discrimination capability.

4.4 Results and Analysis

Model Performance After training phase shown in Fig 6, the trained architecture contains 56.5M parameters (226.6MB weights), achieving:

- **Loss and Accuracy**: The model achieved a low cross-entropy loss of 0.25 on the validation set and 0.28 on the test set. This indicates its ability to accurately distinguish between real and fake videos. The training process showed a consistent improvement in accuracy across epochs, reaching a final accuracy of 89.66%.
- **F1-Score**: With a precision of 88.75% and recall of 90.82%, the model effectively balances true positives while minimizing false positives and achieved 89.78% with F1-Score.
- **AUC Score:** The AUC score of 89.66% reflects that the model has a good ability to distinguish between classes.

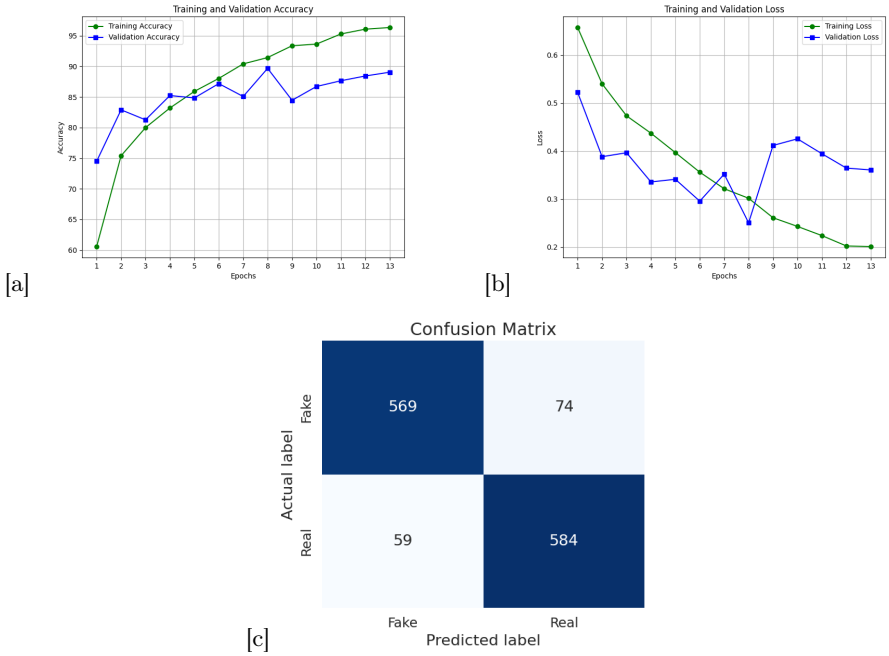

Fig. 6. Training accuracy, loss, and confusion matrix of the model.
(a) Accuracy (b) Loss (c) Confusion matrix

Computational Efficiency The total training time for the model was 77.06 minutes on an RTX 3090 (24GB) GPU.

The training time in Table 3 exhibits a linear growth with increasing sequence length, as shown in Table 3. This linear relationship reflects the efficient processing of sequential data by the LSTM network. The consistent GPU utilization between 98-100% throughout the training process further confirms the efficient use of hardware resources.

Table 3. Training Time vs Sequence Length

Sequence Length	Time (min)	GPU Configuration	Accuracy
10	36.05	RTX 3070 Ti (8GB)	83.44%
20	44.75	RTX 3070 Ti (8GB)	83.05%
40	77.02	RTX 3070 Ti (8GB)	86.16%
80	77.06	RTX 3090 (24GB)	89.66%

Hardware utilization analysis shows:

- Consistent GPU utilization (98-100%) for RTX 3070 Ti and 50% for RTX3090.
- Linear VRAM consumption growth with sequence length.
- Critical VRAM overflow at sequence length 80 on 8GB GPUs with batch size = 4.

Model Comparison To further assess the effectiveness of our model, we conducted a comparative analysis with two other deepfake detection models trained on the same dataset.

- **ResNext-50 without LSTM:** This model replaces the LSTM layer with a simple max pooling layer to aggregate the most important information from the ResNeXt-50 output. It is designed to assess whether the sequential frame learning capability of the LSTM layer significantly enhances deepfake detection performance.
- **ResNet50 +LSTM [25]:** In this model, ResNeXt-50 is replaced with ResNet-50, its predecessor, to evaluate improvements in feature extraction capabilities and their impact on the overall model performance.

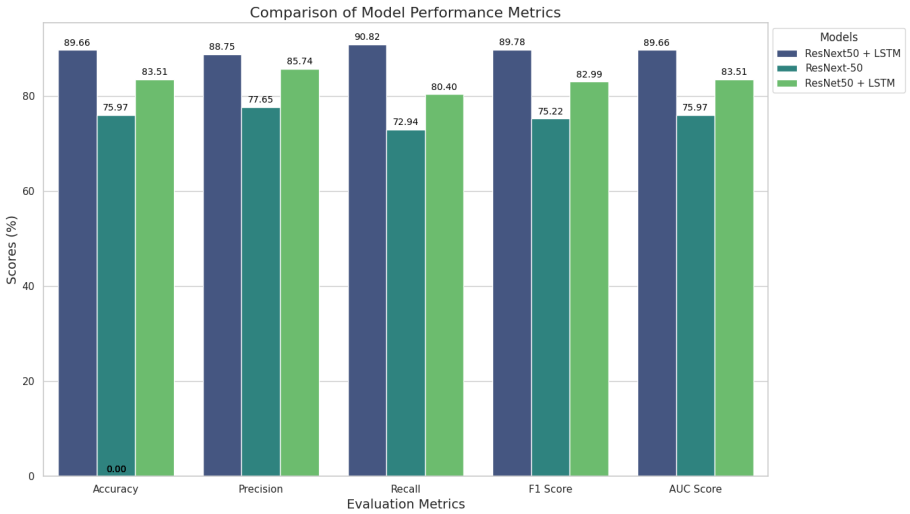

Fig. 7. Bar Chart Comparing Accuracy, F1-score, and AUC for three models

The experimental results, as visualized through Fig 8, reveal significant performance disparities among the three evaluated architectures. As observed in the loss trajectories, ResNext50 + LSTM demonstrates the most stable and consistent reduction in loss, achieving superior training and validation accuracy. In contrast, ResNext-50 without LSTM exhibits pronounced fluctuations in validation loss, indicating instability due to its reliance on static feature retention

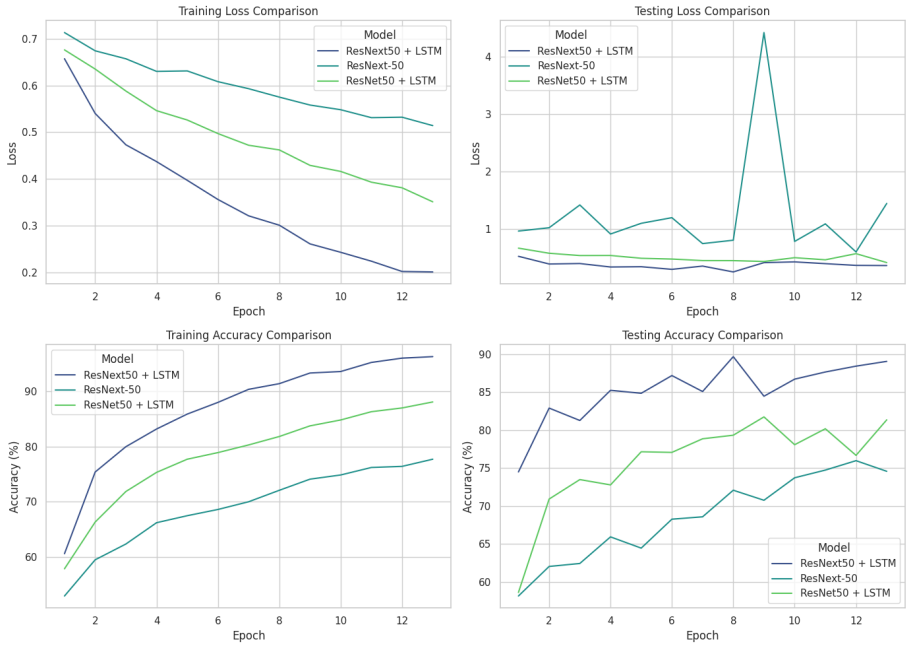

Fig. 8. Plot Chart Comparing Accuracy, Precision for three models

rather than sequential learning. Meanwhile, ResNet50 + LSTM delivers intermediate performance, with steady but suboptimal loss reduction, attributed to its less diverse feature extraction compared to ResNext50's 32-group convolution structure.

Quantitative metrics in Fig 7 corroborate these trends: ResNext50 + LSTM outperforms counterparts across all critical benchmarks, including accuracy, precision, recall, F1-score, and AUC-ROC. Notably, ResNext-50 without LSTM registers the lowest recall and AUC scores, reflecting inferior generalization capabilities.

These findings highlight two key advantages of the ResNext50 + LSTM architecture: (1) its ability to leverage temporal dependencies for sequential feature extraction, and (2) its enhanced representational diversity from grouped convolutions, enabling superior anomaly detection in time-series data compared to non-sequential or less structurally optimized models.

4.5 Limitations and Error Analysis

Despite its strong overall performance, the model exhibits limitations in detecting deepfakes generated using diffusion-based methods. The model achieved a

lower accuracy on diffusion-based deepfakes compared to GAN-based samples. This discrepancy is attributed to the training data composition, which primarily consists of GAN-generated forgeries, leading to a distribution shift that challenges generalization.

Another limitation of the current framework is its inability to process videos containing multiple faces effectively. When presented with multi-face inputs, the model arbitrarily selects a single face for deepfake detection while neglecting the remaining facial data. This approach introduces significant information loss, particularly in scenarios involving multiple subjects.

5 Conclusion

Our approach successfully addressed the challenge of deepfake video detection by the hybrid architecture integrating ResNext50 and LSTM architectures. This combination offers two key advantages: the ability to extract diverse features from deepfake videos with thirty two grouped convolution and the capability to retain temporal dependencies across frames. Despite certain limitations existing, our above experimental results demonstrate the model's potential in tackling the increasingly complex deepfake detection problem in the digital era.

For future work, we plan to leverage knowledge distillation techniques to reduce the model's computational complexity, enabling efficient real-time deployment on mobile devices. This optimization will also support parallel processing of multiple faces and extend the framework to multimodal applications, especially audio analysis. Furthermore, to address the rapidly evolving landscape of synthetic media, we will enhance the model's robustness by integrating advanced unsupervised learning methodologies. Specifically, instead of solely relying on training with labeled datasets produced by state-of-the-art deepfake techniques, we will adopt anomaly detection to enable the model to delve deeper into uncovering subtle irregularities within video content. This approach allows us to move beyond basic supervised classification, providing nuanced insights into abnormal patterns, thereby ensuring greater adaptability to emerging forgery methods.

References

1. Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). *Aggregated Residual Transformations for Deep Neural Networks*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 1492–1500).
2. Hochreiter, S., & Schmidhuber, J. (1997). *Long Short-Term Memory*. Neural Computation, 9(8), 1735–1780.
3. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). *Generative adversarial nets*. In Advances in Neural Information Processing Systems (NeurIPS) (pp. 2672–2680).

4. Dale, K., Sunkavalli, K., Johnson, M. K., Vlasic, D., Matusik, W., & Pfister, H. (2011). *Video face replacement.* In Proceedings of the SIGGRAPH Asia Conference (pp. 1–10).
5. Korshunova, I., Shi, W., Dambre, J., & Theis, L. (2017). *Fast face-swap using convolutional neural networks.* In Proceedings of the IEEE International Conference on Computer Vision (pp. 3677–3685).
6. Thies, J., Zollhöfer, M., Nießner, M., Valgaerts, L., Stamminger, M., & Theobalt, C. (2015). *Real time expression transfer for facial reenactment.* ACM Trans. Graph., 34(6):183–1.
7. shaoanlu. (n.d.). *faceswap-gan: A denoising autoencoder + adversarial losses and attention mechanisms for face swapping.* Retrieved from https://github.com/shaoanlu/faceswap-GAN.
8. Fakeapp. (n.d.). *Fakeapp 2.2.0.* Retrieved from https://www.malavida.com/en/soft/fakeapp/.
9. iperov. (n.d.). *DeepFaceLab: Deepfacelab is the leading software for creating deepfakes.* Retrieved from https://github.com/iperov/DeepFaceLab.
10. Durall, R., Keuper, M., & Pfreundt, F.-J. (2019). *Unmasking DeepFakes with Simple Features.* Journal of Artificial Intelligence Research.
11. Afchar, D., Nozick, V., & Yamagishi, J. (2018). *MesoNet: A Compact Facial Video Forgery Detection Network.* Proceedings of the ACM Conference on Multimedia.
12. Rössler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., Nießner, M. (2019) *FaceForensics++: Learning to Detect Manipulated Facial Images.* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.
13. Haliassos, A., Mira, R., Petridis, S., & Pantic, M. (2023). *Leveraging Real Talking Faces via Self-Supervision for Robust Forgery Detection.* Proceedings of the ACM Conference on Multimedia
14. Shahzad, S. A., Islam, M. J., & Banna, M. H. (2023). *AV-Lip-Sync+: Leveraging AV-HuBERT to Exploit Multimodal Inconsistency for Video Deepfake Detection.* Proceedings of the IEEE International Conference on Computer Vision.
15. Hatami, N., Boudinet, M., & Petrovskii, A. (2020). *CNN+LSTM Architecture for Speech Emotion Recognition with Data Augmentation.* Proceedings of the International Conference on Acoustics, Speech and Signal Processing.
16. Alshehri, M., Alqahtani, A., & Alshehri, M. (2023). *CNN-LSTM and Transfer Learning Models for Malware Classification based on Opcodes and API Calls.* Proceedings of the International Conference on Cybersecurity and Cryptography.
17. Li, Y., Yang, X., Sun, P., Qi, H., & Lyu, S. (2020). *Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics.* Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.
18. Dolhansky, B., Bitton, J., Pflaum, B., Lu, J., Howes, R., Wang, M., & Ferrer, C. C. (2020). *The DeepFake Detection Challenge (DFDC) Dataset.*
19. Stypułkowski, M., Vougioukas, K., He, S., Zięba, M., Petridis, S., & Pantic, M. (2023). *Diffused Heads: Diffusion Models Beat GANs on Talking-Face Generation.* arXiv preprint arXiv:2301.03396.
20. Luo, Z., Chen, D., Zhang, Y., Huang, Y., Wang, L., Shen, Y., Zhao, D., Zhou, J., & Tan, T. (2023). *VideoFusion: Decomposed Diffusion Models for High-Quality Video Generation.* arXiv preprint arXiv:2303.08320.
21. Abid, A., Abdelaal, M., Muhammad, N., Choi, J. H., Mustahsan, Z., & Alfozan, A. (2019). *Gradio: Hassle-free sharing and testing of ML models in the wild.* Retrieved from https://github.com/gradio-app/gradio.

22. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. In Advances in Neural Information Processing Systems 32 (pp. 8024–8035). Retrieved from `https://pytorch.org`.
23. OpenCV Team. (n.d.). *OpenCV: Open Source Computer Vision Library*. Retrieved from https://opencv.org/.
24. Torchvision. (n.d.). *Torchvision: Datasets, Transforms and Models for PyTorch*. Retrieved from https://github.com/pytorch/vision.
25. Karandikar, A., Thakare, Y., Sah, O., Sah, R. K., Nafde, S., & Kumar, S. (2023). *Detection of Deepfake Video Using Residual Neural Network and Long Short-Term Memory*. International Journal of Next-Generation Computing. https://doi.org/10.47164/ijngc.v14i1.1046.
26. Geitgey, A. (n.d.). *face_recognition: The world's simplest facial recognition API for Python and the command line*. Retrieved from https://github.com/ageitgey/face$_r$*ecognition*.