Deepfake Video Detection Using CNNs & Long Short-Term Memory Networks

Jeff Byju
jbyju2@illinois.edu

Sai Kuchibhatla
saik2@illinois.edu

1. Introduction

The rapid advancement of deepfake generation techniques—leveraging deep learning models to synthesize highly realistic yet fraudulent videos—has raised serious concerns regarding the authenticity of multimedia content. These manipulated videos have implications for privacy, security, disinformation campaigns, and societal trust in digital media. Detecting deepfakes automatically and accurately is a pressing challenge.

Recently, several competitions and benchmarks have emerged to encourage the development of robust deepfake detection methods. We participated in a Kaggle competition focused on detecting deepfake videos in a large-scale and diverse dataset. (Kaggle Challenge Link)

In this work, we focus on the Celeb-DF v2 dataset [11], a challenging collection of videos that includes both real and manipulated clips from celebrities. The dataset contains a blend of YouTube-real, Celeb-real, and Celeb-synthesis videos. Its complexity and high realism make it a suitable testbed for robust detection models.

Our approach fuses the strengths of a pre-trained CNN—XceptionNet [3]—with a temporal modeling component—a Bi-LSTM with optional attention. The XceptionNet is utilized as a feature extractor to capture discriminative spatial features from each frame. The extracted features are then fed into a sequence model that leverages temporal patterns. The Bi-LSTM can exploit both forward and backward temporal context, and an attention mechanism highlights the most informative frames. We address class imbalance using focal loss [9], ensuring that the model focuses more on challenging, less represented examples. Additionally, we carefully design our data preprocessing pipeline and use LMDB for efficient storage and retrieval of large numbers of frames during training.

We evaluated various sequence lengths (20, 30, and 40 frames) and model configurations (LSTM vs. Bi-LSTM, with and without attention). Our experiments show that incorporating Bi-LSTM and attention with longer sequences improves performance, as measured by AUC and F1 scores.

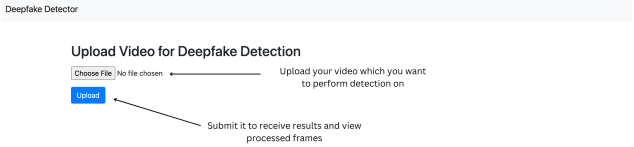

Figure 1. Demo app

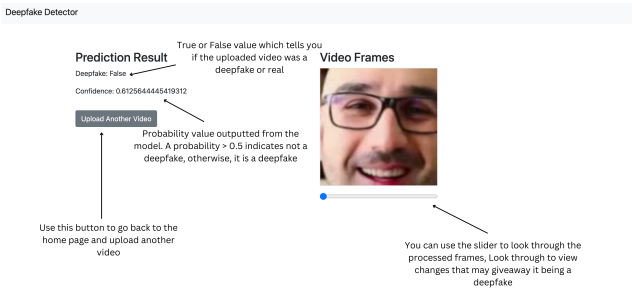

Figure 2. Results for an uploaded video

2. Related Work

- Deepfake Detection: Early deepfake detection methods relied on handcrafted features or inconsistencies in blinking patterns [1,2]. Recently, CNN-based approaches have become popular, with models like Xception [3] and EfficientNet [4] demonstrating strong performance on frame-level classification. Some works have extended detection to spatio-temporal architectures (e.g., 3D CNNs or RNN-based pipelines) [5,6].
- Sequence Modeling: Incorporating temporal information helps distinguish subtle temporal artifacts. LSTM and GRU-based methods have been widely used in video analysis tasks, including action recognition and video classification [7,8]. Bi-directional LSTM (Bi-LSTM)

models are known to capture richer temporal dependencies by processing sequences in both forward and backward directions.

- Attention Mechanisms: Attention enables models to focus on the most discriminative parts of the input. Recent works show that attention can help highlight specific frames or facial regions where deepfake artifacts are most evident [10].
- Loss Functions for Imbalanced Data: Standard BCE loss can be suboptimal for highly imbalanced datasets. Focal loss [9] has been successfully applied in object detection tasks and can help the model pay more attention to hard-to-classify samples in deepfake detection scenarios.

3. Approach

3.1. Data Preprocessing

3.1.1. Data collection and sources

The dataset used for this project is the Celeb-DF v2 dataset, one of the most comprehensive and challenging datasets for deepfake detection. It comprises 5,639 high-quality deepfake videos and 890 real videos. The dataset is constructed from the following sources:

- YouTube Real Videos: Original videos collected from YouTube, featuring various celebrities in diverse settings and lighting conditions.
- Celeb-Real: Real videos curated to match the distribution of the fake videos.
- Celeb-Synthesis: Synthetic videos generated using advanced deepfake synthesis techniques, resulting in high-quality fake videos that are difficult to distinguish from real ones.

We obtained the Celeb-DF v2 dataset following the instructions provided by its creators and used it in accordance with the Kaggle competition rules.

3.1.2. Data labeling

To facilitate training and evaluation, we created a CSV file (final_celeb_df.csv) that maps each video file name to a categorical label indicating whether it is real (1) or fake (0). This labeling process ensured that the model had accurate ground truth for supervised learning and allowed for efficient data handling during the experiments.

3.1.3. Frame Extraction

- Sampling Strategy: We extract every 5th frame from each video to capture different facial expressions and movements, ensuring a diverse temporal representation.
- Face Detection: Using the face_recognition library, we detect and crop the first face in each frame. This focuses the model on the most relevant region altered in deepfakes. We then resized it to 299×299 pixels to match the input size required by XceptionNet.
- Sequence Length: We experimented with sequence lengths of 20, 30, and 40 frames to assess the impact of temporal depth on model performance.

Figure 3. Frames extracted from a video

3.2. Handling Class Imbalance

- Class Distribution: The dataset is imbalanced, containing more fake videos than real ones. Simple BCE loss might bias the classifier towards the majority class.
- Solution: We employ the focal loss function instead of the standard Binary Cross-Entropy (BCE) loss. Focal loss down-weights easy examples and focuses training on hard negatives, mitigating the imbalance effect.

3.3. Data Storage and Optimization

- Parallel Processing: We utilize multiprocessing to accelerate data preprocessing, leveraging available CPU cores effectively.
- LMDB Database: One significant challenge in training deep learning models on large video datasets is efficient input-output (I/O) management. Repeatedly reading large sets of images from the filesystem slows down training and can cause bottlenecks. To address this, we stored all preprocessed frames in an LMDB (Lightning Memory-Mapped Database) environment with compression via lz4.frame. LMDB is a fast key-value store that maps keys to binary blobs. By compressing and storing the tensor

data in LMDB, we achieved:

- Faster Random Access: LMDB's memory-mapped files enable quick retrieval of frames for different videos without repetitive disk seeks.
- Scalability: It handled large amounts of data without complex file handling scripts.
- Reduced Overhead: Preprocessing and transformations were done once, and subsequent training epochs reused the cached data efficiently.

This significantly accelerated training iterations, freeing us to experiment with multiple configurations and longer sequences within the limited computational resources and time allocated by the HPC cluster.

3.4. Model Architecture

3.4.1. Feature Extraction with XceptionNet Network

- Pre-trained CNN: We use the XceptionNet model pre-trained on ImageNet, removing its final classification layer to obtain 2048-dimensional feature embeddings for each frame. We chose XceptionNet because it has consistently demonstrated strong performance on image classification benchmarks. Its depthwise-separable convolutions improve efficiency and representation power, enabling the extraction of rich, discriminative features from faces.
- Frozen Layers: To reduce computational load and overfitting, we freeze the weights of the XceptionNet network during training.

3.4.2. Sequence Modeling with LSTM/Bi-LSTM

- LSTM/Bi-LSTM: A single frame's features, while informative, are insufficient because deepfakes often introduce inconsistencies that emerge over time (e.g., flickering artifacts, unnatural transitions in facial attributes). We feed the sequence of frame embeddings into an LSTM or Bi-LSTM network to capture temporal dependencies across frames.
- Hidden Size and Layers: The LSTM has a hidden size of 512 and consists of two layers. The Bi-LSTM processes the sequence in both forward and backward directions, capturing past and future context. This bidirectionality makes it more robust to variations in how artifacts appear or disappear over time.

3.4.3. Attention Mechanism (Optional)

- Attention Layer: Not all frames are equally informative. Some might be near-identical or lack distinctive manipulations, while others contain pronounced artifacts. An attention layer computes weights over the LSTM outputs, allowing the model to focus on the most informative frames.
- Context Vector: The weighted sum of LSTM outputs forms a context vector used for classification.

3.4.4. Classification Layer

- Fully Connected Layer: A linear layer maps the context vector or final hidden state to a binary output.
- Activation: A sigmoid activation function outputs the probability of a video being fake.

3.4.5. Training Strategy

- Optimizer: We use the Adam optimizer with a learning rate of

$$1 \times 10^{-4}$$

- Batch Size: Due to memory constraints and the size of the model, we set a batch size of 8.
- Epochs: We train for 20 epochs, balancing performance improvement with computational feasibility.
- Validation: The dataset is split into training (60%), validation (20%), and test (20%) sets, ensuring stratification to maintain the class distribution in each subset.
- Evaluation Metrics: Performance is measured using the "Area Under the Receiver Operating Characteristic Curve" (AUC) and F1 score, suitable for imbalanced datasets.

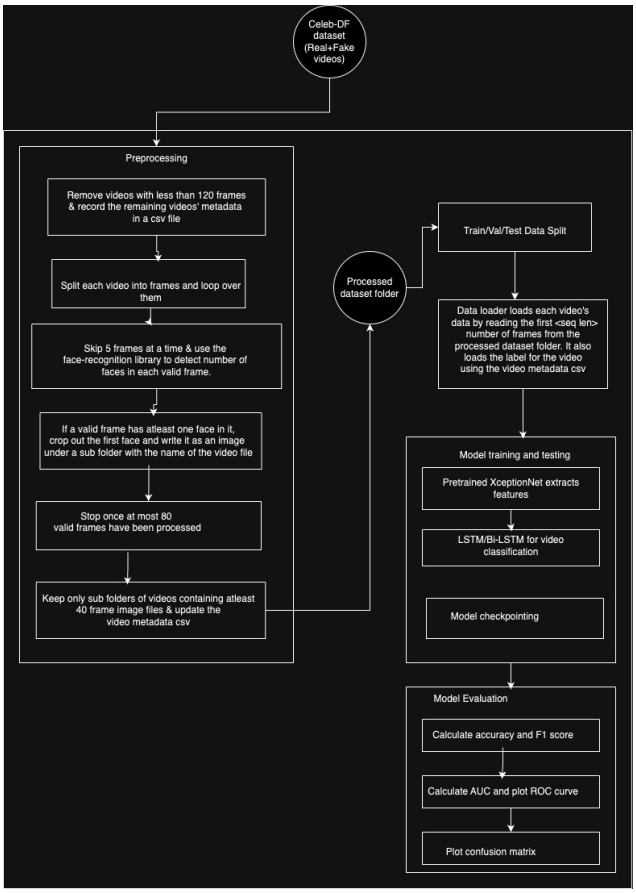

Figure 4. System architecture

4. Results

4.1. Experimental Setup

We conducted experiments under various configurations to evaluate the impact of different components:

- Sequence Lengths: We trained and tested the model with sequence lengths of 20, 30, and 40 frames.
- LSTM Variants: We compared the performance of standard LSTM to that of Bi-LSTM.
- Attention Mechanism: We evaluated models with and without the attention layer.

When training the model using a sequence length of 20, we used a total of 6524 videos (889 real videos and 5635 fake videos) which was split into the following sets :- 3914 videos in the train set, 1305 videos in the validation set and 1305 videos in the test set. When training the model using sequence lengths of 30 and 40 videos, we consider a smaller set of 6329 videos (872 real videos and 5457 fake videos) since not all of the original 6524 videos could provide at least 40 frames given that we are extracting every 4th frame from each video. This set of 6329 videos was split into the following sets :- 3797 videos in the train set, 1266 videos in the validation set and 1266 videos in the test set.

4.2. Quantitative Results

The following tables summarize the performance metrics for different configurations:

For a sequence length of 20 frames, we have -

Model	AUC	F1 score
LSTM without attention	0.9689	0.8444
Bi-LSTM without attention	0.9529	0.8512
LSTM with attention	0.9624	0.8095
Bi-LSTM with attention	0.9746	0.8368

For a sequence length of 30 frames, we have -

Model	AUC	F1 score
LSTM without attention	0.8459	0.4915
Bi-LSTM without attention	0.8961	0.5515
LSTM with attention	0.9536	0.7374
Bi-LSTM with attention	0.9665	0.7557

For a sequence length of 40 frames, we have -

Model	AUC	F1 score
LSTM without attention	0.9150	0.7386
Bi-LSTM without attention	0.8817	0.7048
LSTM with attention	0.9574	0.7604
Bi-LSTM with attention	0.9534	0.7176

5. Discussions and Conclusions

5.1. Observations

- Sequence Lengths:
  - A sequence length of 20 frames resulted in the highest AUC and F1 scores across most configurations, demonstrating that shorter sequences captured sufficient temporal information without introducing noise.
  - Performance declined slightly for sequence lengths of 30 and 40 frames, possibly due to overfitting or redundancy in the sequence information.
- Bi-LSTM vs. LSTM:
  - Bi-LSTM generally outperformed standard LSTM models for F1 score but showed mixed results for AUC. This suggests that bidirectional processing helps capture more contextual information, improving classification confidence for imbalanced data.
- Attention Mechanism:
  - The addition of attention enhanced F1 scores significantly across all sequence lengths, especially with Bi-LSTM models. This validates the hypothesis that attention can help focus on the most relevant frames.
- Optimal Configuration:
  - The Bi-LSTM with attention model achieved the best performance for a sequence length of 20 frames, with an AUC of 0.9746 and an F1 score of 0.8368, making it the optimal configuration for this task.

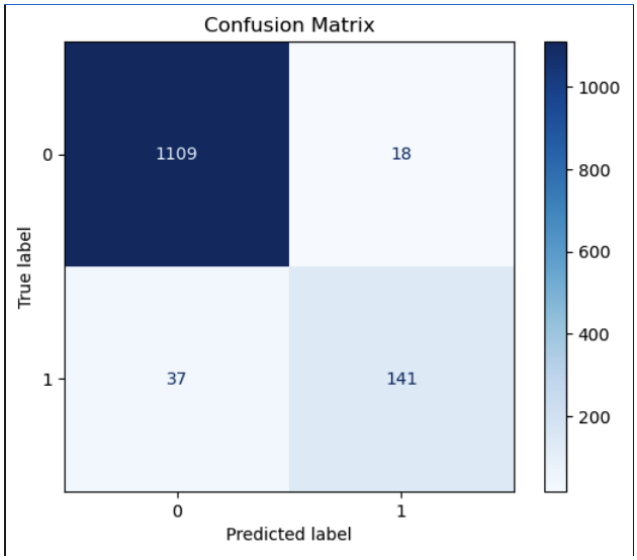

Figure 5. confusion matrix for the optimal configuration

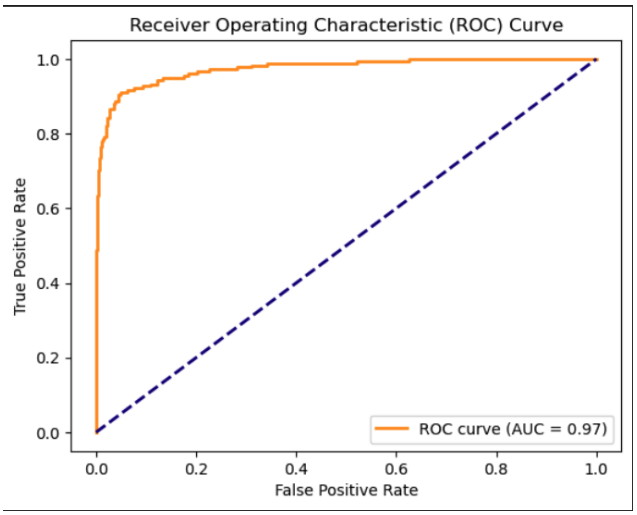

Figure 6. ROC curve for the optimal configuration

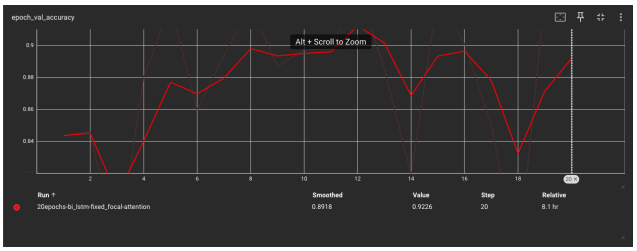

Figure 7. Validation accuracy plot for the optimal configuration

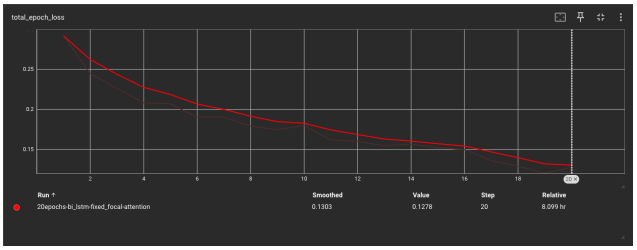

Figure 8. Total loss plot for the optimal configuration

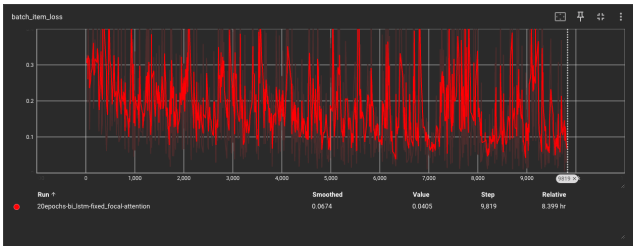

Figure 9. Batch loss plot for the optimal configuration

5.2. Insights

- Sequence Length:
  - The results indicate that shorter sequences (20 frames) are sufficient for detecting deepfakes, likely because most facial anomalies due to deepfake artifacts are localized and do not require long temporal dependencies.
  - Longer sequences (30 and 40 frames) introduced redundancy and did not yield consistent improvements, likely due to overfitting and increased model complexity.
- Attention Mechanism:
  - The attention mechanism played a crucial role in improving F1 scores, particularly for sequences where deepfake artifacts were sparse or subtle. Attention allows the model to dynamically focus on frames that contribute the most to decision-making, leading to more robust classification.
- Bidirectionality:
  - Bi-LSTM models excelled in performance compared to standard LSTM models, underscoring the importance of leveraging both past and future context in sequential data.
- Impact of Class Imbalance:
  - The use of focal loss was instrumental in mitigating the impact of class imbalance, ensuring that the model learned effectively from both real and fake videos.

5.3. Limitations

- Performance for Longer Sequences:
  - The decrease in performance for sequence lengths of 30 and 40 frames highlights the challenge of balancing sequence length and model complexity.
- Computational Constraints:
  - Training longer sequences or experimenting with higher batch sizes was limited by hardware and time constraints.

5.4. Potential solutions

- Transformer Models:
  - Replacing LSTMs with Transformers could improve the model's ability to capture long-range dependencies without the limitations of recurrent architectures.
- Augmentation Techniques:
  - Applying more diverse augmentation techniques (e.g., Gaussian noise, occlusion) could help the model generalize to unseen data.
- Cross-Dataset Evaluation:
  - Testing on datasets like FaceForensics++ or DFDC would assess the model's generalization capabilities across different deepfake generation techniques.
- Multi-modal Features:
  - Incorporating audio and other metadata could provide additional cues for detecting deepfake videos.

- Fine-Tuning the CNN:
  - Unfreezing some layers of XceptionNet could adapt features to deepfake patterns.
- Ensemble Methods: Combining multiple models might yield further gains.

6. Statement of Individual Contributions

- Jeff Byju: Data collection, preprocessing, implementation of the frame extraction pipeline, optimized the pipeline using multiprocessing (yielded a significant speedup), developed the final model integrating both the XceptionNet CNN model and the LSTM/Bi-LSTM model.
- Sai Kuchibhatla: Data Collection, integrated the attention mechanism, optimized the frame extraction pipeline using LMDB, implemented focal loss, conducted model training, designed experiments and analyzed results.

Both team members contributed to the writing of the final report and the preparation of presentation materials.

7. Resource Links

- Public GitHub Repository with Train Code and Flask Application: https://github.com/anandk1999/cs444-final-project
- Google Drive Folder containing all plots for all experiments: https://drive.google.com/drive/folders/19Ca6Eek4DVGnjyYOgzqjUx4XW84xOulE?usp=sharing

8. References

1 Korshunov, P., Marcel, S. "DeepFakes: A New Threat to Face Recognition? Assessment and Detection." arXiv:1812.08685, 2018.

2 Li, Y., Chang, M.C., Lyu, S. "In Ictu Oculi: Exposing AI Created Fake Videos by Detecting Eye Blinking." IEEE Workshop on Information Forensics and Security (WIFS), 2018.

3 Chollet, F. "Xception: Deep Learning with Depthwise Separable Convolutions." CVPR, 2017.

4 Tan, M., Le, Q. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." ICML, 2019.

5 Guera, D., Delp, E.J. "Deepfake Video Detection Using Recurrent Neural Networks." AVSS, 2018.

6 Sabir, E., et al. "Recurrent Convolutional Strategies for Face Manipulation Detection in Videos." CVPR Workshops, 2019.

7 Donahue, J., et al. "Long-term Recurrent Convolutional Networks for Visual Recognition and Description." CVPR, 2015.

8 Yue-Hei Ng, J., et al. "Beyond Short Snippets: Deep Networks for Video Classification." CVPR, 2015.

9 Lin, T.Y., et al. "Focal Loss for Dense Object Detection." ICCV, 2017.

10 Girdhar, R., et al. "Video Action Transformer Network." CVPR, 2019.

11 Li, Y., et al. "Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics." CVPR, 2020.

12 benpflaum, Brian G, djdj, Irina Kofman, JE Tester, JLElliott, Joshua Metherd, Julia Elliott, Mozaic, Phil Culliton, Sohier Dane, and Woo Kim. Deepfake Detection Challenge. https://kaggle.com/competitions/deepfake-detection-challenge, 2019. Kaggle.