Video Forgery Detection

Deep Learning DATS 6303 - Fall 2024

Group 6

Individual report

Author: Bharat Khandelwal

Table of Contents:

Introduction

The exponential growth of multimedia content has revolutionized how we communicate and share information. Approximately 3.2 billion images and 720,000 hours of videos are shared daily (source). However, this surge in content has also given rise to challenges such as misinformation and deepfake manipulation.

Deepfakes—synthetically generated or altered videos and images—pose a significant threat to digital trust. These manipulations have been used for:

- Political Propaganda: Fake videos of politicians making inflammatory statements.
- Financial Fraud: Identity theft through morphed images.
- Misinformation Campaigns: Spreading false narratives with convincing fake visuals.

The societal impact of deepfakes is profound, ranging from eroding public trust to causing financial losses and reputational damage. Detecting these manipulations is crucial to mitigating their harmful effects.

Our project aims to address this issue by developing robust deep-learning models for forgery detection. By leveraging state-of-the-art datasets like CelebDF and implementing advanced architectures such as GRU, VGG, and 3dCNN, we strive to create a reliable system for identifying manipulated multimedia content.

Repository Structure

The repository is structured into different folders based on the experiments conducted:

1. cnn_basic/:
    - Contains code for implementing a basic CNN model.
    - Focused on lightweight architectures for efficient image-based forgery detection.
2. streamlit/:
    - Includes the Streamlit app for user interaction.
    - Features dropdown menus for model selection, file upload functionality, and preprocessing steps.
3. vgg/:
    - Contains code for adapting the VGG model for forgery detection.
    - Includes auxiliary feature integration (e.g., mean RGB values).
4. video_mask/:
    - Focuses on preprocessing video data by extracting frames and applying masks.
    - Implements different approaches for data loading and auxiliary feature extraction.
5. eda.py:
    - Performs exploratory data analysis (EDA) on the dataset.
    - Calculates mean pixel values for real and fake images.

Each folder represents a milestone in our journey toward building an effective forgery detection system.

Dataset

We used the CelebDF dataset, a large-scale dataset specifically designed for deepfake detection (source). The dataset is divided into three categories:

1. Real Videos:
    - Authentic videos of celebrities.
2. Fake Videos:
    - Deepfake-generated videos using advanced synthesis techniques.
3. YouTube Videos:
    - Real-world examples of multimedia content.

The CelebDF dataset contains approximately 6,000 videos, with around 1,100 real videos and the remaining being forged videos. The forged videos are generated using advanced synthesis techniques, making them highly realistic and difficult to detect. This diversity and quality make CelebDF a challenging benchmark for forgery detection models.

Proof of Concept with Image Dataset

Before transitioning to the video dataset, we used an image dataset to conduct a proof of concept (POC). This dataset contained approximately 70,000 images, with around 10,000 real images and the remaining being fake images generated through various methods such as:

- GANs (Generative Adversarial Networks): Used to create highly realistic fake images.
- FaceLabs: A tool for generating manipulated facial images.
- FaceApp: A popular app for facial transformations.

The image dataset allowed us to experiment with preprocessing techniques, auxiliary feature extraction (e.g., mean RGB values), and initial model training. These experiments provided valuable insights into handling real vs. fake data and informed our approach to working with the more complex video datasets.

(a) (b)Fig 1a,b : Real and fake image.

Figures 1a and 1b depict the real and distorted images through the faceapp method.

Models Tried and Architecture:

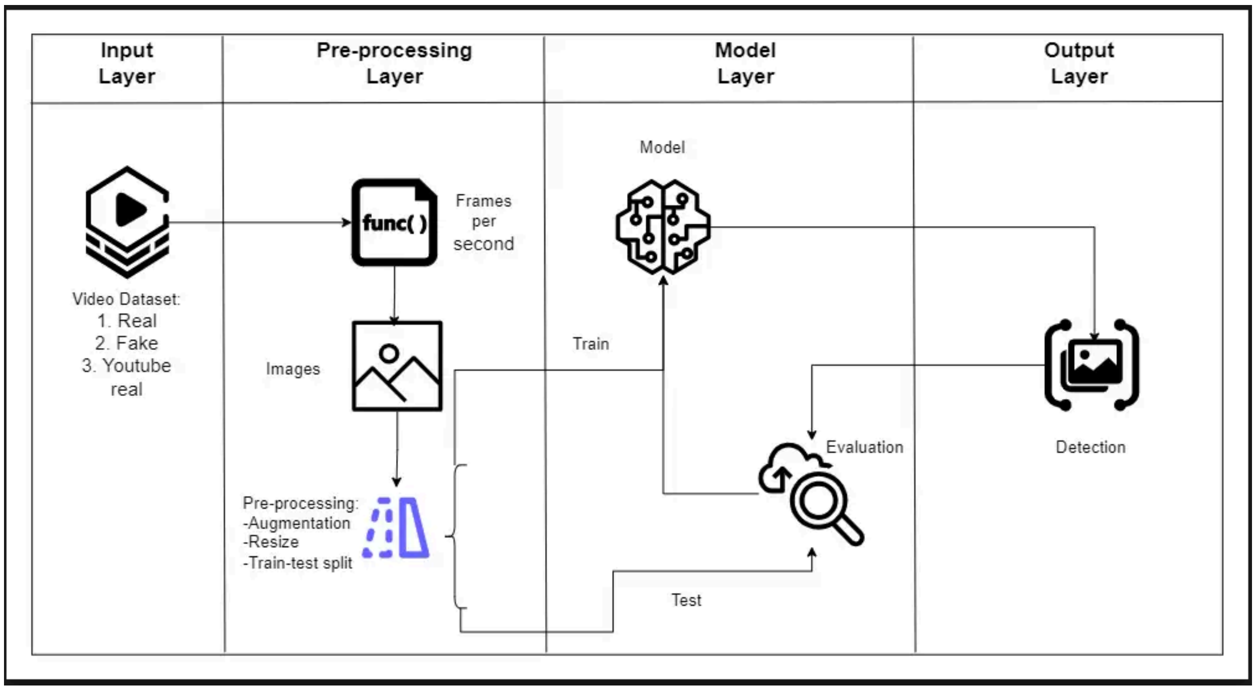


Figure 2: Flow Architecture

Figure 2 represents the comprehensive workflow of the forgery detection system, highlighting the sequential steps involved in detecting manipulated multimedia content. The Input Layer consists of a video dataset categorized into real videos, fake videos, and YouTube real videos. These datasets serve as the foundation for training and testing the models. The Preprocessing Layer involves converting videos into frames at specific rates (frames per second), followed by augmentation, resizing, and splitting the data into training and testing sets. This ensures that the data is well-prepared for effective model training.

The Model Layer is where the core of the system resides. It includes training deep learning models on preprocessed data and evaluating their performance using unseen test data to ensure generalization. The trained model learns to distinguish between real and fake content based on spatial and temporal features extracted from the frames. Finally, the Output Layer is responsible for detecting forgery in multimedia content based on the predictions made by the model. This layer outputs whether a given video is real or fake, providing a robust solution to combat misinformation caused by deepfakes.

This structured pipeline ensures that each layer contributes to building an efficient and accurate forgery detection system, capable of handling diverse datasets with high-quality synthesis techniques.

1. CNN with GRU

In the project, we combined a Convolutional Neural Network (CNN) with a Gated Recurrent Unit (GRU) to process video data and classify it as "Real" or "Fake." This approach leveraged the strengths of CNNs for feature extraction and GRUs for temporal sequence modeling.
Key Features:

- Video frames were resized to `(299x299)` dimensions to match the input requirements of the InceptionV3 feature extractor.
- A pretrained **InceptionV3** CNN was used to extract rich spatial features (2048-dimensional vectors) from each frame.
- GRU efficiently processed sequential data, identifying motion and temporal inconsistencies across frames.
- Dense layers following the GRU reduced the temporal representation and performed binary classification.
- **Focal loss** was used to address the class imbalance, ensuring the model focused on hard-to-classify examples.

Challenges:

- Uniform frame sampling from videos might miss critical frames containing temporal artifacts, impacting the model's ability to generalize.
- Overfitting due to some target leakage.

- Data loading was one of the biggest challenges as each video frame being extracted and then loaded to the model took a significant amount of time.
- Focal loss was not correctly used to handle the huge imbalance.

2. CNN with Bidirectional GRU

We extended the CNN-GRU architecture by replacing the standard GRU with a Bidirectional GRU (BiGRU). This enabled the model to capture dependencies in both forward and backward directions in the temporal sequence, improving its ability to detect subtle patterns in the video.
Key Features:

- I added another video dataset DFDC to increase the number of real videos and fake video for processing.
- BiGRU captured temporal dependencies from both past and future frames, providing a more comprehensive understanding of video dynamics.
- A temporal attention layer was added after the BiGRU to focus on the most relevant frames in the sequence.
- Dense layers following the BiGRU + attention reduced the temporal representation and performed binary classification.
- The model was trained using focal loss with right values of gamma and alpha, which helped improve performance on imbalanced datasets.

Challenges:

- While attention improved interpretability and performance, it added additional parameters, increasing the risk of overfitting on a smaller dataset.
- The bidirectional nature of GRU increased the model's complexity, resulting in higher computational requirements during training and inference.
- Uniform sampling of frames from videos sometimes missed subtle artifacts critical for classification, reducing generalization on certain test cases.

Results:

1. CNN with GRU

	Precision	Recall	f1-score

Real	0.6	0.9	0.72
Fake	0.99	0.94	0.96
accuracy			0.93
macro avg	0.8	0.92	0.84
weighted avg	0.95	0.93	0.94
F1-Macro score Test accuracy	0.84 0.93		

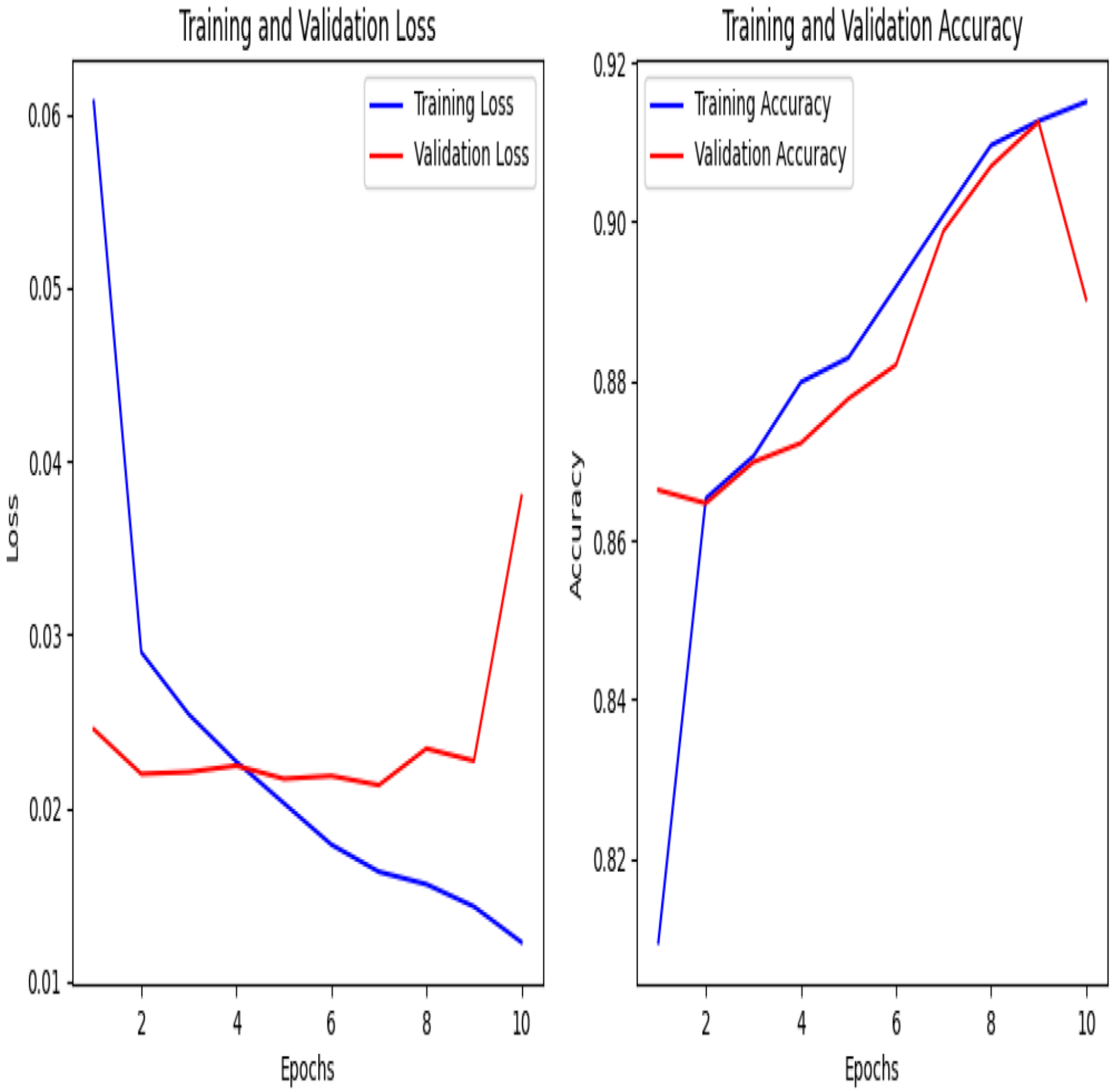
These results seem to be overfitting as this model is not correctly able to classify the real videos, which is mostly because of the class imbalance.

2. CNN with Bidirectional GRU

	Precision	Recall	f1-score
Real	0.45	0.04	0.07
Fake	0.87	0.99	0.93
Accuracy			0.87
macro avg	0.66	0.51	0.5
weighted avg	0.82	0.87	0.81

F1 Macro Score Test accuracy	0.4969 0.877		

In this model we were able to achieve an average precision of 66% as precision is the metric that is usually considered for computer vision tasks like these.This model is not overfitting as it considers the frames from both past and future making it more robust for this type of a problem and additional attention weights focus on data that might be crucial but isn't captured from the above model.



Results Summary:

Key Observations:

- **CNN with GRU**:
  - High overall accuracy but overfits to the majority class ("Fake"), leading to poor generalization for the "Real" class.
- **CNN with Bidirectional GRU**:
  - Improved robustness due to bidirectional layers and attention mechanisms, but it still struggles with the minority class ("Real"), achieving only 4% recall for this class.

Recommendations:

- Address the class imbalance by employing techniques like oversampling the minority class, undersampling the majority class, or using class weighting in the loss function.
- Explore additional models or adjustments, such as ensemble learning, to improve classification performance for the minority class without sacrificing overall accuracy.

Disclosure: Outside Code

In the initial phase of the project, I encountered challenges related to extracting frames from images, which I addressed by following online tutorials. This approach helped me quickly overcome the learning curve. During the training and testing phases, I initially attempted to write the code independently but faced memory constraints that required the use of image generators. To tackle this, I referred to online resources.

While I relied on code from the internet, most of it required significant modification to meet the specific needs of the project. Based on the formula in instructutions, for every 100 lines of code I sourced, I modified 45 lines and added 25 lines of original code. This means that approximately 44% of the code came from external sources.

Learning Curve

Our journey involved several milestones that shaped our understanding of forgery detection challenges. We began by exploring repositories like Awesome Deepfakes Detection to identify suitable datasets for our project. After evaluating multiple options, we selected the CelebDF dataset due to its diversity and relevance for deepfake detection.

As a proof of concept (POC), we started with image datasets to understand how to load, preprocess, and segment data into real and fake classes. Preprocessing steps included resizing images, normalizing pixel values, and extracting auxiliary features such as mean RGB values. When these auxiliary features were concatenated with image features at the fully connected layer stage in both VGG and CNN models, the model's accuracy doubled, demonstrating the importance of incorporating additional contextual information.

Exploratory Data Analysis (EDA) revealed key differences between real and fake images. For example, real images had lower red channel values compared to fake images. These insights guided our preprocessing strategies and helped refine our models.

After achieving satisfactory results on image datasets, we transitioned to video datasets. This phase involved extracting frames from videos at specific intervals (e.g., 100 FPS), calculating auxiliary features for each frame, and training models on these frames. The inclusion of auxiliary data significantly improved the model's ability to detect forged content.

We also explored Error Level Analysis (ELA) as a forensic technique for detecting image manipulation (source). ELA analyzes compression artifacts in JPEG images by re-saving them at a lower quality and comparing error levels between the original and recompressed versions. Regions with higher error levels often indicate tampering or editing. Although primarily used for static images, ELA provided valuable insights into preprocessing techniques for identifying manipulated content.

Additionally, we reviewed several research papers to refine our approach:

- "FaceForensics++: Learning to Detect Manipulated Facial Images": This paper introduced a large-scale dataset for detecting facial manipulations and explored deep learning architectures for forgery detection.
- "DeepFake Detection Using Recurrent Neural Networks": The authors proposed using RNNs to analyze temporal inconsistencies in video frames for detecting deepfakes.
- "Exposing DeepFake Videos by Detecting Face Warping Artifacts": This study focused on identifying warping artifacts introduced during face-swapping processes in deepfake videos.
- "Deepfake Detection Challenge Dataset": This was a challenge conducted by meta and they were able to achieve 82.56 percent average precision

These papers provided critical insights into state-of-the-art methods for forgery detection and inspired us to integrate auxiliary features into our models. Through these milestones, we gained valuable insights into forgery detection challenges and developed robust preprocessing pipelines tailored to our dataset.

Conclusion

This project represents a significant step toward combating misinformation caused by deepfakes and manipulated media. By leveraging state-of-the-art datasets like CelebDF and implementing advanced deep learning architectures , we demonstrated promising results in detecting forgery in multimedia content.
Key Takeaways:

1. Understanding the dataset is critical before implementation—EDA revealed valuable insights that guided our preprocessing strategies.
2. Correctly using Focal Loss can enhance model performance by eliminating a lot of class imbalance between real and fake data.
3. The effectiveness of combining CNNs with GRU-based architectures for detecting deep fake videos.
4. By leveraging **InceptionV3** for spatial feature extraction and GRUs for temporal modeling, we successfully captured both spatial and temporal inconsistencies in video data.

Despite promising results, challenges like computational overhead and class imbalance highlight the need for further optimization to achieve robust and efficient performance on large-scale video datasets.

Future Work

To enhance our forgery detection system further, we propose the following directions:

1. Face Mesh Analysis:
    - Generate 3D face meshes for detected faces in videos.
    - Analyze changes in node values on the mesh to identify subtle manipulations.
2. Image Masking Techniques:
    - Implement masking algorithms to highlight differences between real and altered images.

    - Use these masks as auxiliary features during training for improved accuracy.
3. Multi-modal Approaches:
    - Combine video and audio streams for forgery detection.
    - Analyze inconsistencies between visual (video) and auditory (audio) modalities to detect manipulation.
4. Temporal Feature Extraction:
    - Explore architectures like LSTMs or Transformers for capturing temporal dependencies in video datasets.
    - Train models on sequences of frames instead of individual frames for better context understanding.
5. Real-world Testing:
    - Test models on real-world examples from platforms like YouTube or social media.
    - Evaluate performance on diverse scenarios beyond curated datasets like CelebDF.

By addressing these areas, we aim to build a comprehensive forgery detection system capable of tackling real-world challenges effectively. This extended documentation provides an in-depth overview of your project's progress, achievements, challenges faced, key insights gained through research integration, and future directions! Let me know if you need further refinements!