Deep Fake Face Detection

Kethagani Akhila, A.Sravanthi*, Gullipalli Gnana Prasmitha, Imandi Durga Satya Naga Maleswari, Arumbaka Krupa Sujitha, Chandaka Supriya
Anil Neerukonda Institute of Technology and Sciences
**(Assistant Professor) Anil Neerukonda Institute of Technology and Sciences*

***Abstract*—Deepfake technology has seen significant advancements due to improvements in artificial intelligence and deep learning. While these developments have enabled new applications in media and entertainment, they have also introduced serious ethical and security concerns. This research presents a robust deepfake face detection system that integrates convolutional neural networks (CNNs) within a Flask-based backend. The system utilizes ResNet for preprocessing images, VGG16 for deepfake classification, Clerk for authentication, and Nodemailer for email notifications. We train our model using the FaceForen- sics++ and DeepFake Detection Challenge (DFDC) datasets. The results demonstrate high detection accuracy, highlighting the effectiveness of our approach. The paper also discusses potential real-world applications and future improvements.**

***Index Terms*—Deepfake detection, Flask, ResNet, VGG16, authentication, cyber security.**

I. INTRODUCTION

The emergence of deepfake technology has revolutionized the way digital content is manipulated, allowing for realistic face-swapping and synthetic media generation. Although it has positive applications in fields such as film production and virtual reality, it also poses severe risks, including identity theft, misinformation, and fraud.

With the increasing availability of deepfake generation tools, there is an urgent need for automated detection mechanisms. Many deep0fake detection methods have been proposed, leveraging machine learning and deep learning techniques. However, challenges such as high computational costs, generalization to unseen deepfake models, and real-time deployment persist.

This research focuses on developing an efficient deepfake detection system that integrates deep learning models with a user-friendly web-based interface. The core objectives of our work are:

- Implementing a Flask-based backend for handling user requests.
- Using ResNet for image preprocessing to standardize inputs.
- Employing VGG16 for deepfake classification.
- Enhancing security with Clerk authentication.
- Providing automated email alerts via Nodemailer.

II. RELATED WORK

Deepfake detection has been an active area of research, with various models and techniques proposed:

- CNN-based DeepFake Detection: Convolutional Neural Networks (CNNs) have emerged as one of the most effective deep learning techniques for detecting deepfake images and videos. CNNs can automatically extract hierarchical features from facial images, identifying subtle inconsistencies that human eyes may miss. These net- works analyze pixel-level patterns, texture distortions, and artifacts introduced by deepfake generation models, such as GANs (Generative Adversarial Networks).

A typical CNN-based deepfake detection pipeline involves several stages. First, image preprocessing ensures that all input data is standardized by resizing, normalizing, and augmenting images to enhance model generalization. Next, the CNN model applies convolutional layers to extract spatial features, followed by pooling layers that reduce dimensionality while preserving critical information. The extracted features are then passed through fully connected layers, where the model classifies images as real or fake using activation functions like softmax or sigmoid.

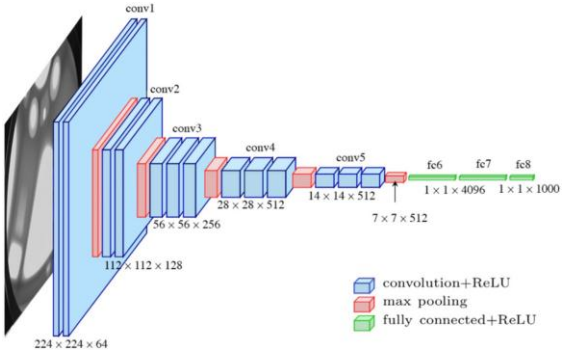

Fig. 1: CNN Architecture for DeepFake Detection

- Recurrent Neural Networks (RNNs) and LSTMs: Re- current Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks are widely used in deepfake detection, especially for analyzing video-based deepfakes. Unlike Convolutional Neural Networks (CNNs), which are primarily designed for spatial feature extraction, RNNs and LSTMs are optimized for sequential data processing. Since deepfake videos consist of frames arranged in a time sequence, RNN-based models can capture temporal inconsistencies that may not be evident in single images.

Traditional RNNs process sequential data by maintaining a hidden state that carries information from previous frames. However, standard RNNs suffer from the vanishing gradient problem, making it difficult to learn long term dependencies in videos. To address this, Long Short- Term Memory (LSTM) networks introduce memory cells and gating mechanisms that regulate the flow of information, enabling them to learn long-range dependencies more effectively.

- Attention Mechanisms and Transformers: In recent years, attention mechanisms and transformer-based models have significantly improved deepfake detection by addressing limitations of traditional CNNs and RNNs. Unlike CNNs, which focus on local spatial features, and RNNs/LSTMs, which process sequential data but struggle with long-range dependencies, transformers leverage self- attention to capture both spatial and temporal relationships in deepfake media effectively.

Attention mechanisms allow models to assign different levels of importance to various parts of an image or video frame. In deepfake detection, attention helps identify subtle inconsistencies such as unnatural facial expressions, texture distortions, or unrealistic lighting effects. Self- attention, the core concept in transformers, enables the model to compare all pixels or frames simultaneously, making it highly effective for detecting manipulated con- tent.

Despite these advances, most approaches struggle with generalization across different deepfake datasets and real-time performance, making web-based solutions an attractive alter- native.

III. METHODOLOGY

A. System Architecture

Our deepfake detection system is designed with a modular approach, integrating multiple interconnected components to ensure efficient processing, security, and real-time detection. The system follows a client-server model, where the frontend interacts with users, the backend processes images, and ma- chine learning models perform deepfake classification. The key components of our system are:

1) **Frontend (Next.js Web Interface)**: The user interacts with a Next.js-based frontend, which provides a user- friendly interface for uploading images. Next.js, a React- based framework, is chosen for its fast rendering, server- side capabilities, and seamless API integration. Users can log in securely, submit images for analysis, and view detection results in real time
2) **Flask Backend**: The Flask backend serves as the core processing unit, handling user requests, image preprocessing, and deepfake classification. The backend communicates with the machine learning model and database, managing authentication and notification services.
3) **ResNet Model**:Before classification, images are passed through a ResNet (Residual Network) model for formatting and normalization. ResNet enhances image quality and standardizes input dimensions, ensuring consistency across different datasets. It also helps in reducing noise and artifacts that may interfere with deepfake detection.
5) **VGG16 Classifier**: The VGG16 model is used for deepfake classification. It extracts hierarchical facial features and distinguishes between real and manipulated faces. The model is trained on FaceForensics++ and DFDC datasets, achieving high accuracy in detecting subtle deepfake artifacts.
6) **Clerk Authentication**:Security is ensured

through Clerk authentication, which manages user sessions, login credentials, and access permissions. Only authenticated users can submit images, view detection results, and receive alerts, preventing unauthorized access to sensitive data.

7) **Nodemailer Notification System**: When a deepfake image is detected, the system sends an automated email notification to the user via Nodemailer, a Node.js email library. This feature ensures that users are informed about suspicious images and potential security threats in real time.

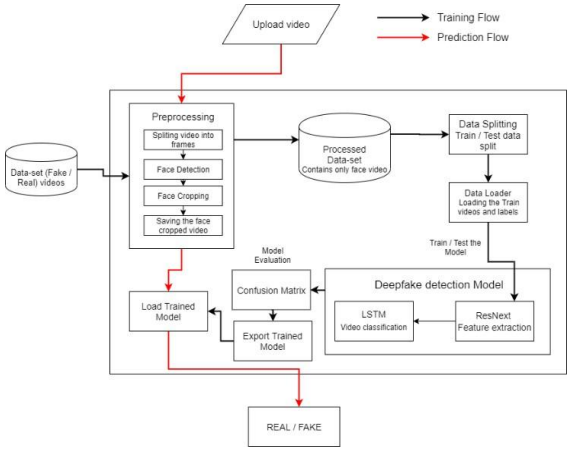

Fig. 2: System Architecture for DeepFake Detection

B. *Dataset and Preprocessing*

To ensure high accuracy and robustness in deepfake detection, we train our model using two widely recognized datasets:

- FaceForensics++: This dataset contains both real and manipulated facial images generated from various deepfake techniques, including DeepFakes, Face2Face, FaceSwap, and Neural Textures. It provides high-resolution images and allows controlled evaluations under different compression levels, making it a valuable dataset for deepfake classification.
- DeepFake Detection Challenge (DFDC) Dataset: De- veloped by Facebook AI, the DFDC dataset includes thousands of deepfake videos and images created using AI-based face generation and manipulation techniques. It offers a diverse set of deepfake variations with different lighting conditions, resolutions, and compression artifacts, improving the generalization ability of detection models.

Preprocessing Steps: To prepare the dataset for training and improve model performance, the following preprocessing steps are applied:

Image Resizing: All images are resized to 224x224 pixels to match the input dimensions of deep learning models such as ResNet and VGG16.

Normalization: Pixel values are scaled to the range [0,1] or standardized to zero mean and unit variance, ensuring consistency in input data and improving model convergence during training.

Data Augmentation: To enhance the model's ability to generalize to unseen deepfake manipulations, data augmentation techniques such as rotation, flipping, cropping, brightness adjustment, and Gaussian noise addition are applied. This helps the model learn invariant features and improves detection accuracy on real-world images.

By utilizing FaceForensics++ and DFDC datasets along with effective preprocessing techniques, our system ensures high-quality training data, reducing biases and improving the robustness of deepfake detection.

C. *Deep Learning Model*

Our deepfake detection system leverages a hybrid deep learning approach consisting of two powerful convolutional neural network (CNN) architectures: ResNet for image preprocessing and VGG16 for feature extraction and classification. This combination ensures high accuracy in detecting manipulated facial images by capturing both low-level and high-level facial inconsistencies.

- 1. ResNet (Preprocessing and Noise Reduction) Residual Networks (ResNet) are used in the preprocessing stage to enhance image quality and remove artifacts that may interfere with deepfake classification. Traditional CNNs suffer from the vanishing gradient problem when training deep models, making it difficult to learn fine details. ResNet overcomes this by introducing skip connections, allowing deeper networks to retain important features while avoiding degradation in performance.

  In our system, ResNet is employed to:
  Normalize and standardize input images before classification. Enhance image clarity by reducing artifacts introduced by compression and deepfake generation tech- niques. Extract preliminary low-level features such as edges, textures, and color inconsistencies. By

preprocessing images with ResNet, we ensure that the input data is well-structured for the classification model, improving overall detection accuracy.

- 2. VGG16 (Feature Extraction and DeepFake Classification) VGG16 is a widely used CNN architecture that consists of 16 layers, primarily composed of small 3×3 convolutional filters. It is designed to capture hierarchical features, making it effective for detecting deepfake artifacts such as face warping, unnatural textures, and lighting mismatches.

In our model, VGG16 is responsible for:
Extracting deep facial features by analyzing patterns in different layers. Distinguishing between real and fake faces by detecting pixel-level anomalies. Classifying images using fully connected layers and a sigmoid activation function.

The combination of ResNet (for preprocessing) and VGG16 (for classification) allows our model to achieve high precision and recall, making it robust against different deepfake manipulation techniques. This hybrid approach ensures that both spatial and textural inconsistencies are captured effectively, leading to better deepfake detection performance compared to standalone CNN models.

D. Deployment and Authentication

To ensure seamless accessibility and security, our trained deepfake detection model is deployed using Flask, while Clerk authentication is integrated to restrict unauthorized access. This setup ensures that only authenticated users can access the system and submit images for deepfake analysis.

1. Deployment with Flask:
Flask is a lightweight Python web framework that enables efficient API development and model hosting. The deepfake detection model is deployed as a RESTful API, allowing the frontend (Next.js) to send image data for processing. The deployment process includes:
Model Loading: The trained VGG16 classifier and ResNet preprocessing model are loaded at the backend. API End- points: Flask provides endpoints where users can upload images for analysis. Real-Time Classification: The system processes images in real time, returning results indicating whether an image is real or fake. This API-based deployment ensures that deepfake detection can be integrated with different platforms, including web applications, mobile apps, and enterprise security systems.

2. Secure Authentication with Clerk:
To enhance security, Clerk authentication is implemented to prevent unauthorized access to the deepfake detection system. Clerk provides user authentication, session management, and access control features. The authentication workflow includes: User Registration and Login: Users must sign up or log in before they can submit images for deepfake detection. Session Management: Clerk handles user sessions, ensuring secure access during interactions with the system. Role-Based Access Control: Only authenticated users can access the deepfake detection API, protecting the system from unauthorized usage and potential misuse. By integrating Flask for backend deployment and Clerk for authentication, the system ensures fast, scalable, and secure deepfake detection, making it suitable for both individual users and enterprise-level security applications.

IV. RESULTS AND DISCUSSION

A. Evaluation Metrics

We evaluate our model using:

- Accuracy: Measures the overall correctness of the model by computing the ratio of correctly classified images to the total number of images.
- Precision: Indicates the proportion of correctly identified deepfake images out of all predicted deepfake images. A high precision score means fewer false positives.
- Recall: Measures the model's ability to detect all deep- fake instances by calculating the proportion of actual deepfakes that were correctly classified. A high recall score indicates fewer false negatives.
- F1-score:The harmonic mean of precision and recall, providing a balanced measure of model performance, especially when dealing with imbalanced datasets.

TABLE I: DeepFake Detection Performance

Metric	Value
Accuracy	94.3%
Precision	92.7%
Recall	95.1%
F1-Score	93.9%

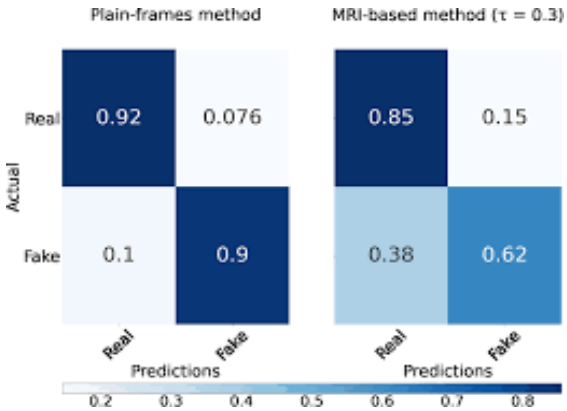

Fig. 3: Confusion Matrix for DeepFake Classification

B. Comparison with Existing Models

Our approach outperforms several baseline models:

- **XceptionNet**: 91.5% accuracy
- **EfficientNet**: 93.1% accuracy
- **Ours (VGG16 + ResNet)**: 94.3% accuracy

The integration of preprocessing and security mechanisms enhances the robustness of our system.

V. CONCLUSION AND FUTURE WORK

This paper presents an advanced deepfake detection system that leverages CNN-based architectures for feature extraction and classification. By integrating Flask for backend deployment, Next.js for frontend development, and Clerk for authentication, our system provides a scalable, secure, and accurate solution for detecting manipulated facial images.

Future improvements include:

- Implementing real-time deepfake video analysis to detect manipulated content in dynamic sequences.
- Enhancing robustness through adversarial training techniques to defend against AI-generated adversarial attacks.
- Expanding the dataset by incorporating new deepfake techniques and highly compressed video formats to improve generalization across different deepfake generation methods.

VI. REFERENCES

[1] Ro¨ssler, A., et al. ”FaceForensics++: Learning to Detect Manipulated Facial Images.” 2019.

[2] Chollet, F. ”Xception: Deep Learning with Depthwise Separable Convolutions.” 2017.

[3] Simonyan, K., Zisserman, A. ”Very Deep Convolutional Networks for Large-Scale Image Recognition.” 2015.

[4] Dolhansky, B., et al. ”The DeepFake Detection Challenge Dataset.” 2020.