Prof.Rinku Badgujar, 2025, 13:3
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology

An Open Access Journal

Pixelproof : Uncovering The Truth In Images

Prof. Rinku Badgujar ,Nagesh Kannure , Shriyog Borse
Department Of Computer Science And Engineering MIT School Of Computing Pune, India

Abstract- Deepfake videos are getting much better and appearing more often. This makes it hard to know what's real online and can cause problems for trust. To help with this, our project looked at how to change and improve a type of AI (called a Convolutional Neural Network, or CNN) to spot deepfakes in individual pictures (frames) from a video. We started with an AI model called EfficientNetB0 that was already trained on many different images. Then, we trained it more using special sets of real and fake video frames that we prepared. During this extra training, we used methods like 'data augmentation' (creating more varied training images from the ones we have) and adjusted the 'learning rate' (how fast the AI learns) to make it work better. These steps also helped fix issues like 'overfitting,' which happens when the AI learns the training data too specifically and doesn't do well on new, unseen data. Our work showed that the AI model could learn to work with these video frames. At first, we faced some challenges like overfitting. However, using the improved data augmentation and changing the learning rate during training helped make the model more stable and perform better on our test set of video frames. The model achieved a score (AUC) of about 0.72, showing it had a fair ability to tell the difference between real and fake frames. This project shows that it's tricky to make deepfake detectors trained on still pictures work well for videos, and it takes several steps of trying things out and making improvements. We learned some useful ways to make these models better, and this work can be a starting point for creating even better deepfake video detectors in the future.

Keywords- Deepfake videosAI trust issues, Convolutional Neural Network (CNN), EfficientNetB0 Transfer learning Video frame, classification Data augmentation, Learning rate adjustment,Overfitting, Model stability, Generalization.

I. INTRODUCTION

The rapid advancement of deepfake technology has blurred the line between authentic and manipulated media, posing serious risks to privacy, security, and the integrity of information. PixelProof is an AI-powered system designed to counter these challenges by detecting deepfake and forged content in images and videos. Utilizing state-of-the-art machine learning and computer vision techniques, it identifies subtle inconsistencies and artifacts, such as pixel anomalies and temporal mismatches, that reveal tampering. By empowering users with reliable tools for content verification, PixelProof aims to restore trust and ensure the authenticity of digital media in an increasingly manipulated digital landscape.

II. RELATED WORKS

FaceForensics :
It is a comprehensive dataset designed to facilitate the training and evaluation of deepfake detection models. It comprises over 1,000 high-resolution videos manipulated using four state-of-the-art face forgery methods: FaceSwap, Face2Face, DeepFakes, and NeuralTextures. The dataset includes three levels of compression (high, medium, and low) to simulate real-world scenarios where videos may be degraded by platforms like social media.

XceptionNet :
It is a deep convolutional neural network initially developed for image classification but later adapted for deepfake detection tasks. It uses depthwise separable convolutions, a technique that significantly reduces the number of parameters and computational cost compared to traditional convolutional layers.

GAN Fingerprints Analysis :
Generative Adversarial Networks (GANs) are the backbone of many deepfake generation techniques. These networks consist of two components: a generator that creates fake content and a discriminator that evaluates the authenticity of the content. Over time, GANs introduce unique "fingerprints" or patterns into the generated media due to the specific ways in which the networks learn to create realistic content. These fingerprints can manifest in various forms, such as pixel-level artifacts, unnatural textures, or discrepancies in lighting and shadows.

III. THE PROPOSED SYSTEM

i. System Architecture
PixelProof is designed to detect deepfake content in images or videos using AI and machine learning techniques. Here's an outline of its system architecture:

Data Input Layer
This layer is responsible for receiving the input to be analyzed. It accepts either individual images (e.g., JPG, PNG) or video files (e.g., MP4) provided by the user. (For the model's training phase, data was sourced from public image datasets like rvf10k and custom-collected real/fake videos from which frames were extracted).

Preprocessing Layer
This layer prepares the input data for the detection model.

- For Videos: The system first extracts individual frames from the input video.
- For Images/Frames: Each image or extracted frame is resized to a standard size. Images/frames are then converted into numerical arrays. Pixel values are maintained in the [0, 255] range, consistent with the model's training data.

Deepfake Detection Core

- Model: The core of the detection system employs a fine-tuned EfficientNetB0 convolutional neural network. This model was initially pre-trained on a broad image dataset and subsequently fine-tuned specifically on a custom dataset comprising real and synthetically generated deepfake video frames.
- Detection Process: The preprocessed image or frame is fed to this fine-tuned model. The model then classifies the input as "real" or "fake," outputting a prediction score (probability) that indicates its confidence.

Post-processing & Aggregation Layer (Primarily for Video Inputs)

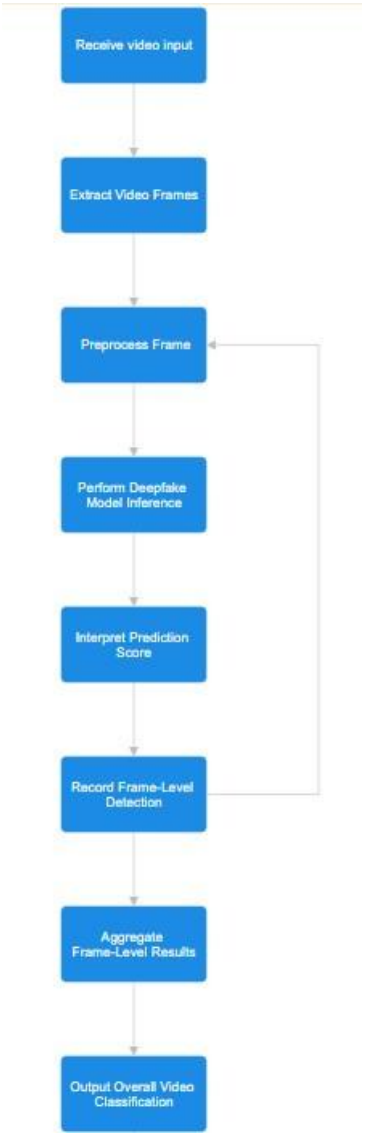
- For Single Images: The classification ("real" or "fake") from the Detection Core is treated as the final result for the input image.
- For Videos: The system collects the "real" or "fake" classifications for each processed frame. An aggregation strategy (e.g., determining if a majority of frames are classified as "fake," or if the count of "fake" frames exceeds a certain threshold) is then

applied to derive an overall classification for the entire video.

User Interface (UI) & Interaction Layer

- Interface: A web-based application developed using Streamlit provides an accessible and user-friendly frontend.
- Interaction: Users can upload an image or video file through this interface.
- Results Display: The final "real" or "fake" classification for the submitted content is presented to the user in a clear and understandable manner.

Flow Chart



ii. Data Collection and Dataset Creation

PixelProof's model development involved two key datasets: one for initial image-based training and a second, custom-created set of video frames for fine-tuning and evaluation.

Initial Image Training Dataset: The EfficientNetB0 model was first trained using rvf10k, a public Kaggle dataset. This dataset comprises 10,000 balanced still images of real and deepfake faces, structured into train and valid sets, providing a foundational model for detecting deepfake characteristics in static images.

Video Frame Dataset: For video-specific adaptation, a custom dataset was created by extracting frames from real and fake videos using OpenCV, with every 60th frame selected. These frames were organized into:

- Training set: 37,429 real and 37,429 fake frames.
- Validation set: 7,195 real and 7,195 fake frames. All frames were 224x224 pixel PNG files, labeled based on source video authenticity. This dataset was essential for fine-tuning the model for video deepfake nuances and for performance evaluation.

iii. Model Training and Optimization

PixelProof's core EfficientNetB0 model was developed through a two-stage process using the TensorFlow and Keras frameworks: initial training on still images, followed by fine-tuning on video frames.

Initial Image-Based Model Training:

The EfficientNetB0 model, pre-trained on ImageNet, was first adapted for still image deepfake detection using the rvf10k dataset. The process involved:

- Adding a custom classification head (GlobalAveragePooling2D, Dropout 0.2, and a final Dense sigmoid layer).
- An initial training phase with the EfficientNetB0 base layers frozen, using an Adam optimizer (e.g., learning rate 0.001) and BinaryCrossentropy loss.

- A subsequent fine-tuning stage where deeper layers of EfficientNetB0 (from layer 200 onwards) were unfrozen and trained with a much lower learning rate to refine weights.

Video Frame Fine-Tuning:
The image-trained model was then further fine-tuned for video frame analysis using the custom dataset (approx. 75k training, 14k validation frames, 224x224 pixels). Key aspects included:

- Loading the fine tuned image model
- Applying data augmentation techniques specifically for video frames (RandomFlip, RandomRotation, RandomZoom, RandomContrast,
- RandomBrightness) to the training set to improve generalization and combat overfitting.
- Employing an Adam optimizer with a low initial learning rate and a ReduceLROnPlateau callback (monitoring val_loss, factor 0.2, patience 2) for dynamic learning rate adjustment.
- Using Early Stopping) and ModelCheckpoint to save the best performing model based on validation loss. These strategies were crucial for adapting the model to video data and mitigating overfitting observed in training history.

iv. User Interface and Interaction Design
PixelProof utilizes a web-based Graphical User Interface (GUI) built with Streamlit to provide an accessible way for users to interact with the deepfake detection model. Streamlit was selected for its rapid development capabilities with Python.
The user interaction workflow is designed for simplicity:

Input Submission: The user accesses the Streamlit application and is presented with a file uploader. They can select and upload either a single image (e.g., JPG, PNG) or a video file (e.g., MP4).

Processing Trigger: Upon file submission and typically by clicking an "Analyze" button, the backend Python logic integrated within the Streamlit application is initiated.

Backend Operations:

- The system loads the fine-tuned fine tuned video model.
- For an image input: The image is preprocessed (resized to 224x224 pixels, pixel values [0, 255]).
- For a video input: Frames are extracted using OpenCV. Each relevant frame then undergoes the same preprocessing (resize to 224x224, pixel values [0, 255]).
- The model performs inference on the preprocessed image(s)/frame(s).
- For video input: Predictions from individual frames are aggregated (e.g., via majority vote or thresholding) to form a single classification for the entire video.

Result Display: The final detection result ("REAL" or "FAKE") for the submitted image or video is then clearly presented to the user within the Streamlit interface.

This design ensures that users can easily submit content for analysis and receive a straightforward deepfake detection outcome.

IV. CONCLUSION

This project, PixelProof, aimed to adapt and evaluate a deep learning model, specifically EfficientNetB0, for detecting deepfakes in both images and videos. The methodology involved initial training on an image dataset, followed by specialized fine-tuning on custom video frames, incorporating data augmentation and learning rate optimization.

PixelProof demonstrated the capability to learn distinguishing features between real and manipulated content. The fine-tuning process, while revealing challenges such as overfitting, provided valuable insights into strategies for adapting image-centric models to video data. A user interface was also conceptualized to illustrate the system's practical application.
Limitations noted offer clear directions for future work, including the exploration of temporal information in videos, advanced regularization techniques, and further expansion of training data diversity. This research contributes to the practical

understanding of building deepfake detectors and underscores the iterative efforts required to enhance their effectiveness against evolving manipulated media.

V. CHALLENGES AND LIMITATIONS

Throughout the development and evaluation of the PixelProof system, several challenges were encountered, highlighting practical considerations in deepfake detection research:

Environment Configuration: Establishing a stable and compatible GPU-accelerated deep learning environment using WSL2, NVIDIA drivers, CUDA, and cuDNN presented initial setup complexities due to strict version dependencies required by TensorFlow.

Video Data Diversity and Volume: Acquiring and curating a sufficiently large and diverse dataset of video deepfakes, representing various manipulation techniques, proved challenging for the video frame fine-tuning stage. The model's generalization capability is often linked to the breadth of examples seen during training.

Overfitting During Fine-Tuning: The EfficientNetB0 model, when fine-tuned on the custom video frame dataset, exhibited a tendency to overfit the training data. While strategies like data augmentation and learning rate scheduling were implemented and provided some mitigation, effectively managing overfitting to achieve higher validation scores remained a persistent challenge.

Computational Resources: The fine-tuning process, particularly with data augmentation and numerous video frames, was computationally intensive and required significant time, even with GPU acceleration, underscoring the resource demands of deep learning projects.

Achieving High Detection Accuracy: The final validation accuracy (approximately 63-69%) indicates that while the model learned to distinguish deepfake frames, significant scope for improvement exists. Adapting an image-centric CNN to the nuances of video, using a frame-by-frame approach, without explicit temporal modeling, presented inherent limitations.

Generalization to Unseen Deepfakes: While not exhaustively tested within the scope of this project, the general challenge of deepfake detectors struggling with novel manipulation techniques not encountered during training is an implicit limitation to consider.

RESULTS

The system was tested with various types of images and network conditions. The results showed that the system could successfully encrypt and decrypt images without any loss of quality. Furthermore, the system proved to be secure against common attacks such as eavesdropping and man-in-the-middle attacks.