Deep Fake Detection Using CNNs

Ayush Inamdar
230269
Ikrima Badr Shamim Ahmed
230482

Yenni Rishi
231199
Mansi Ghodke
230633

Abstract

Detecting manipulated facial images and videos is an increasingly important topic in digital media forensics. As advanced face synthesis and manipulation methods are made available, new types of fake face representations are being created which have raised significant concerns for their use in social media. The deepfake detection model proposed in this study is a dual-branch neural network architecture that combines EfficientNet (a pre-trained convolutional neural network) with a handcrafted feature extraction branch. This dual-branch structure aims to enhance detection performance by leveraging both high-level feature representations from a deep learning model and specific handcrafted statistical features, creating a more comprehensive approach to identify deepfake images. Our model was trained and evaluated on a comprehensive dataset of real and deepfake photos, achieving an accuracy of 73 precent , thus providing a robust approach for reliable deepfake detection.

1. Introduction

With the rapid growth of deepfake technology, synthetic media can now be generated with astonishing realism using readily available tools and resources, often on consumer-grade hardware. This ease of production has made deepfakes a significant challenge in various domains, from social media misinformation to identity theft. The novelty of this idea lies in the detection method of merging deep learning and handcrafted feature extraction in a unified architecture. While typical models rely solely on CNNs for feature extraction, our approach incorporates specific handcrafted features (color histograms, edge, and texture information) alongside deep network outputs to capture additional, nuanced image characteristics. This combined approach aims to improve detection accuracy and robustness, especially in scenarios where subtle visual artifacts are difficult for standard CNNs to detect alone.

2. Proposed method

2.1. EfficientNet Branch

The first branch of the model is based on EfficientNetB0, a highly efficient convolutional neural network architecture. EfficientNet was chosen due to its scalable, computationally efficient design and its strong performance on image classification tasks. Here is a detailed breakdown of its role in the model.

2.1.1 Input and Preprocessing

The model receives RGB images as input with a shape of 224x224 pixels and 3 color channels.Each image is preprocessed to normalize pixel values between 0 and 1, improving the training stability and convergence speed of the model.

2.1.2 Feature Extraction using EfficientNetB0

EfficientNetB0 is initialized with pre-trained ImageNet weights to leverage prior knowledge and features learned from a large-scale dataset. Using these pre-trained weights accelerates training and boosts performance by providing a strong initialization. In this architecture, EfficientNet is used without its top classification layers, effectively serving as a feature extractor. The output of EfficientNet is a feature map that represents high-level abstractions from the image, such as complex textures and spatial relationships that could indicate tampering.

2.1.3 Post-Extraction Processing

Local Binary Patterns are used to extract texture features. LBP encodes the local texture by comparing pixel intensities within a small neighborhood around each pixel. The output is a 10-bin histogram of texture values, representing how pixel intensities change across the image. Texture analysis with LBP is effective for identifying subtle structural differences between real and manipulated images.

2.2. Handcrafted Feature Branch

The second branch of the model is designed to capture unique, manually-defined image descriptors that can reveal anomalies in pixel-level statistics, color distributions, and texture, which are often indicative of deepfake manipulation. The handcrafted feature branch extracts and processes these features as follows.

2.2.1 Color Histogram Features (216 dimensions)

A 3D color histogram is computed across the RGB channels, with 6 bins per channel. This results in a 6x6x6 (216-dimensional) histogram, capturing the distribution of color intensities. Color inconsistencies or unnatural color distributions often indicate image manipulation, making color histograms valuable for identifying deepfakes.

2.2.2 Edge Features (16 dimensions)

Edge detection is performed using Canny edge detection on the grayscale version of the image. The edges are represented as a 1D histogram with 16 bins, encoding the frequency of different edge intensities. Edge features are essential because deepfake generation techniques can introduce irregular edge patterns, especially around facial boundaries.

2.2.3 Local Binary Patterns (LBP) - Texture Features (10 dimensions)

Local Binary Patterns are used to extract texture features. LBP encodes the local texture by comparing pixel intensities within a small neighborhood around each pixel. The output is a 10-bin histogram of texture values, representing how pixel intensities change across the image. Texture analysis with LBP is effective for identifying subtle structural differences between real and manipulated images.

2.2.4 Statistical Features (20 dimensions)

Global and regional statistics are computed from the grayscale version of the image to capture overall pixel distribution characteristics. Global statistics include the mean, standard deviation, median, and percentiles (25th and 75th) of pixel intensities. Regional statistics are computed over three horizontal image sections (top, middle, bottom), where each section provides similar statistical features. Statistical features are particularly helpful in capturing abnormal intensity distributions, which can occur due to inconsistencies in lighting, shading, or blending in fake images.

2.2.5 Feature Vector Construction

The color, edge, texture, and statistical features are concatenated to form a 262-dimensional handcrafted feature vector. This feature vector is fed into separate dense layers with ReLU activation and dropout, providing a refined, compressed representation of the handcrafted features.

2.3. Feature Fusion and Classification

After processing through their respective branches, the outputs from both EfficientNet and the handcrafted feature dense layers are concatenated to create a unified feature representation. This combined feature vector leverages the complementary strengths of both feature types, capturing nuanced spatial patterns from EfficientNet and statistical irregularities from handcrafted features.

2.3.1 Combined Feature Layers

The concatenated feature vector is passed through a series of dense layers with batch normalization and dropout, ensuring that the model can generalize effectively while avoiding overfitting. These dense layers enable the model to learn complex interactions between neural network-derived and handcrafted features, enhancing its ability to detect deepfake characteristics.

2.3.2 Output Layer

Finally, the model has a binary classification output layer with a sigmoid activation function, which provides a prediction score indicating the probability of an image being fake.

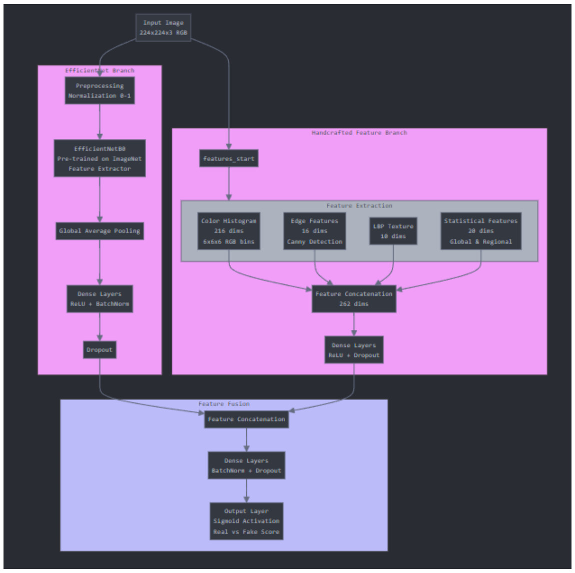


Figure 1. Flowchart depicting the method

Experiments Conducted

First Approach: Ensemble of CNNs Based on EfficientNetB4

Link to the research paper

The authors provide a technique to identify modified face information in films by employing an ensemble of convolutional neural networks (CNNs) based on EfficientNetB4, improved by attention layers and siamese training. This method combines several models, each trained to identify distinct facial manipulation signs, in an attempt to capture subtle deepfake artifacts.

Key Points of the Approach **Model Ensemble**: The ensemble uses both standard and attention-modified networks to concentrate on pertinent face regions by fusing CNNs with an EfficientNetB4 backbone.

Attention Mechanism: The EfficientNetB4 model is modified with an attention module to highlight areas like the eyes and mouth, where manipulation artifacts are most likely.

Siamese Training Strategy: A triplet loss-based siamese training technique is employed to improve feature separability between real and artificial faces, assisting the network in learning more accurate representations for every class.

Disadvantages

- **Computational Cost**: Combining multiple CNNs and attention mechanisms increases the computational load and memory requirements, which may not be suitable for resource-constrained environments.
- **Complexity in Training**: The siamese training strategy and attention layers add complexity to the model, requiring careful tuning of hyperparameters and increased training time.
- **Potential for Overfitting**: Given the model's high sensitivity to subtle features, it may overfit to the specific deepfake artifacts present in the training dataset, reducing its generalizability to unseen manipulation methods.

Process Overview The following table summarizes the main steps involved in this approach:

We were unable to continue with this approach because of insufficient computational power.

Second Approach: Detecting Deepfakes without seeing any

Link to the research paper

Traditional detection methods rely on supervised learning, which works well with known deepfake types but struggles with new, unseen methods. This research introduces a ”fact checking” approach adapted from fake news detection to improve generalizability against novel deepfake types.

Key Points of the Approach **Concept of False Facts**: Inaccuracies, or ”false facts,” including mismatched audio-visual content or modified identities, are common in deepfake media. Attacks like face swaps and audio-visual mismatches rely heavily on these misleading facts.

FACTOR Framework: By using pretrained feature extractors and avoiding the need for deepfake training data, the authors present FACTOR, a fact-checking system that compares certain media claims (such as identity and speech) to actual media attributes. FACTOR measures the degree of agreement between stated facts and observed media content using similarity ratings, or cosine similarity.

Attack Scenarios:FACTOR is applied across three attack types:

Face Swapping: By checking if the facial identity in a fake aligns with reference images of the claimed identity.

Audio-Visual Synthesis: Detects inconsistencies between audio and visual cues using pretrained audio-visual encoders.

Text-to-Image (TTI): Analyzes if images match their text prompts, exploiting weaknesses in models like Stable Diffusion, which often overfit to their training prompts (e.g., CLIP).

Process Overview The following table summarizes the main steps involved in the FACTOR approach:

Disadvantages While this ensemble approach with EfficientNetB4 and siamese training enhances deepfake detection accuracy, it comes with certain disadvantages:

- **Limited to falisifiable claims**: To validate deepfake content, FACTOR uses particular, verifiable claims (such identification or audio-visual alignment). When no specific fact can be verified, it is less successful in identifying ”unconditional” deepfakes, such as artificially created images or films that make no explicit claims.
- **Limited Applicability to Non-Standard Facts**: FACTOR is designed to work with standard facts like identity or alignment. Deepfakes with more nuanced manipulations (e.g., subtle expression changes or modifications to non-focal elements) that don't involve verifiable claims may evade detection.

Step	**Description**
Input Video Frame	Video frames are provided, containing both real and potentially manipulated (fake) faces.
Face Extraction and Preprocessing	Faces are extracted from frames and preprocessed (cropped and resized to 224x224 pixels) to ensure uniformity.
EfficientNetB4 Backbone	The base network, EfficientNetB4, processes the preprocessed images to learn features for classification.
Attention Mechanism	Highlights specific facial areas (e.g., eyes, mouth) to focus on relevant parts for manipulation detection.
Siamese Training	A training strategy that compares real vs. fake features through pairs, aiding in feature distinction.
Model Ensemble	Combines predictions from various models (e.g., EfficientNetB4, EfficientNetB4 with attention) to improve accuracy.
Detection Output	Outputs a confidence score indicating the likelihood of each face being manipulated.

Figure 2. Process overview of the CNN ensemble approach based on EfficientNetB4.

Step	**Description**
Input Media	Media input such as images, video frames, or audio-visual content is provided for analysis.
Feature Extraction	Off-the-shelf encoders (e.g., face recognition or audio-visual encoders) are used to extract features from the input media.
Claimed Fact Identification	Identify the claimed fact related to the media (e.g., the identity of a person, audio-video consistency).
Cosine Similarity Calculation	Compute the cosine similarity between the features of the input media and reference features to produce a truth score.
Truth Score Analysis	Analyze the truth score to determine if it is sufficiently high. Low truth scores indicate potential deepfake media.
Classification	Classify the media as real or fake based on the evaluated truth score.

Figure 3. Process overview of the FACTOR approach for detecting deepfakes.

- **Computational Intensity**: FACTOR is computationally intensive since it necessitates several feature extraction and similarity computation processes. This may make it impractical for real-time applications or for devices with constrained processing power.

We decided not to continue with this approach due to it's unreliability in scenarios where a fake news is not associated with the picture

Final Approach: EfficientNetB0 Backbone

The backbone of the model is based on EfficientNetB0, a popular CNN architecture known for its efficiency and accuracy in image classification tasks. EfficientNetB0 is used here as a feature extractor, leveraging weights pre-trained on ImageNet. By setting $include\_top$=False, the final fully connected layers were removed to enable fine-tuning for our specific task. Only the initial convolutional layers were used to extract spatial features from images. During initial experiments, these layers were frozen to prevent overfitting and reduce computational demands.

Key Points of the Approach **Training procedure: Data Preparation**: The dataset consisted of real and fake images with metadata indicating their labels. To manage computational resources, I downsampled both real and fake classes to 10,000 samples each, creating a balanced dataset.The data was then split into training (56%), validation (24%), and test (20%) sets using stratified sampling to preserve the class distribution across splits.

Initial Training: The model was initially trained with the EfficientNetB0 backbone frozen, allowing only the top layers and handcrafted feature branch to learn. We used binary cross-entropy loss and the Adam optimizer with a learning rate of 0.001. To prevent overfitting, early stopping and learning rate reduction, callbacks were used. Early stopping monitored validation loss, and ReduceLROnPlateau lowered the learning rate if validation perfor-

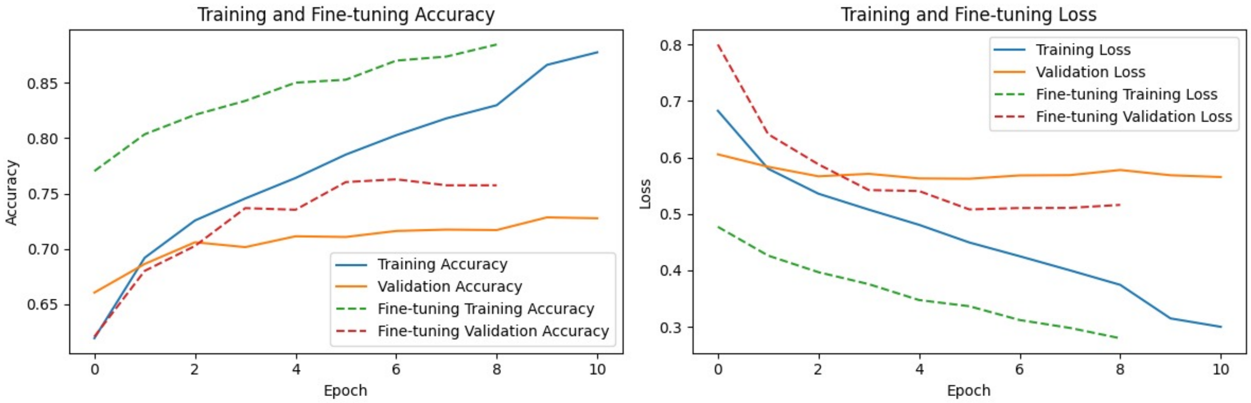


Figure 4. Enter Caption

mance plateaued, further optimizing convergence.

Fine-tuning:After initial training, We unfroze part of the EfficientNetB0 backbone to allow fine-tuning of the deeper layers, enhancing the model's feature extraction for our specific data. Approximately 30% of the EfficientNet layers (starting from the earlier layers) remained frozen to retain general features learned from ImageNet. This approach allowed the model to focus on adjusting only the higher-level, task-specific features. The learning rate was reduced to 1e-5 for fine-tuning, making the training process more stable and avoiding sudden fluctuations in model weights.

Challenges Faced: The handcrafted feature extraction (e.g., color histograms, LBP, and statistical features) significantly increased data loading time. Since these features needed to be calculated on-the-fly for each image, it required a carefully optimized generator. To mitigate this, We ensured efficient batching and reduced computational load per image.

Fine-tuning the dropout rates, batch normalization layers, and learning rates for each branch required multiple iterations. We used EarlyStopping and ReduceLROnPlateau callbacks extensively to find an optimal balance between learning speed and generalization.

Results and Evaluation: The final model achieved satisfactory test accuracy and low loss, indicating that both image features from EfficientNetB0 and handcrafted features contributed to improved detection capabilities.We plotted the training and validation accuracy and loss over epochs, capturing both the initial training phase and the fine-tuning phase. These visualizations provided insights into model convergence and the effectiveness of fine-tuning.

3. Appendix

3.1. Github Repository of Code and Dataset:

Link to Github Repository
Link to Dataset

4. Results

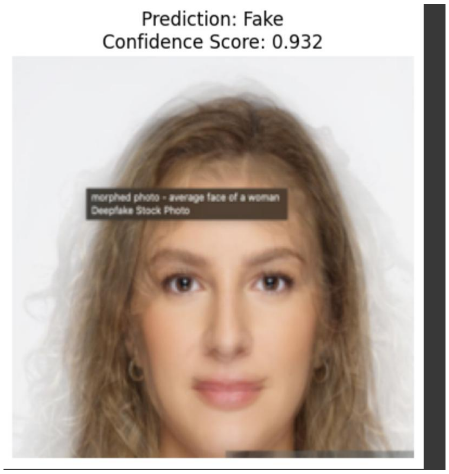
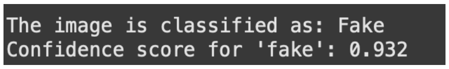


Figure 5
deepfake of a woman

Figure 6
deepfake of a woman



```
The image is classified as: Fake
Confidence score for 'fake': 0.932
```

Figure 7
deepfake of a woman