Performance of a CNN Backbone in a Deepfake Classification Model

Sarosh Abbas Hassan
Islamabad
saroshabbas@gmail.com

***Abstract*—The detection of deepfake content is one of the classic challenges in computer vision. This study investigates the effectiveness of convolutional neural network (CNN) architectures — specifically a CNN backbone model and a residual (ResNet-inspired) network — for distinguishing between real and fake human face images. Using a limited subset of the *Comprehensive Deepfake Detection Dataset* [1] due to memory constraints, we extract low-level and mid-level features including color histograms, Local Binary Patterns (LBP), and edge density, augmenting the raw image data with these feature maps. Two architectures are compared using recall as the key evaluation metric. The study demonstrates that early feature fusion and progressive convolutional filters enhance detection performance, achieving a recall above 97% with optimized hyperparameters via Keras Tuner.**

I. Introduction

Deepfakes are synthetic media where a person's likeness is replaced with that of another using AI. This technology has proliferated due to advancements in generative adversarial networks (GANs) and diffusion models. Detecting such manipulations has become increasingly difficult as the synthetic visual artifacts they produce become more photorealistic.

This study aims to explore a CNN-based approach for deepfake detection using a hybrid feature extraction pipeline that incorporates both pixel-space and texture-space descriptors. By fusing traditional computer vision features (e.g., LBP, edge density, color histograms) with raw image data, we aim to improve the generalization capability of the model and reduce dependency on purely learned representations. Two neural architectures, a lightweight CNN and a ResNet-style model are evaluated to understand the trade-off between complexity, accuracy, and recall.

II. Methodology

A. Dataset Procurement and Preprocessing

The dataset consists of real and fake face images divided into two directories:

`FAKE_DIR="dataset/deepfake"`
and `REAL_DIR="dataset/real"`.
Due to memory limitations on the experimental setup (Google Colab), only a subset of 8,000 images (balanced between real and fake) was used. Data was split using stratified sampling into 80% training and 20% testing sets to ensure class balance.

B. Exploratory Data Analysis (EDA)

EDA was conducted on five randomly selected samples from each class:

- **Color Histogram:** 3-channel (RGB) histograms were computed for each image and averaged across classes. Cosine similarity was used to compare each histogram to the class-average histogram.
- **Local Binary Patterns (LBP):** Texture patterns were extracted using an 8-neighbor LBP operator, and similarity was computed using cosine similarity.
- **Edge Density:** Canny edge detection quantified the ratio of edge pixels to total pixels. Class-average edge density maps were compared to each image's map using inverse difference.

These handcrafted descriptors were later concatenated with the original images to form an enriched multi-channel feature tensor.

C. Feature Augmentation and Data Pipeline

The processed dataset was augmented by stacking:

1) Original RGB image
2) LBP feature map
3) Edge density map
4) Color histogram heatmap (reshaped to 128×128)

This composite representation aimed to encode both pixel-level and statistical characteristics.

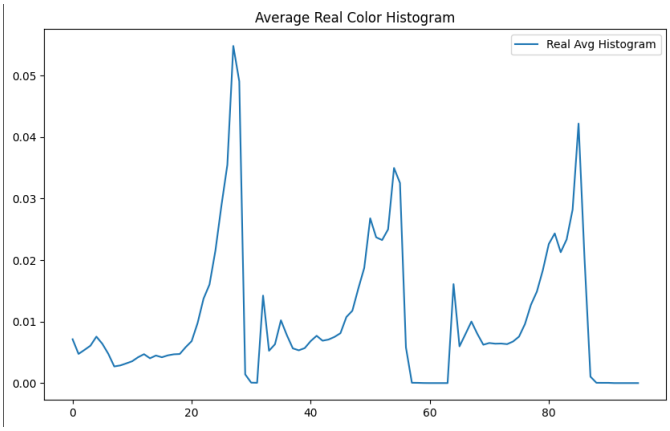

Fig. 1. Average Real image Colour Histogram

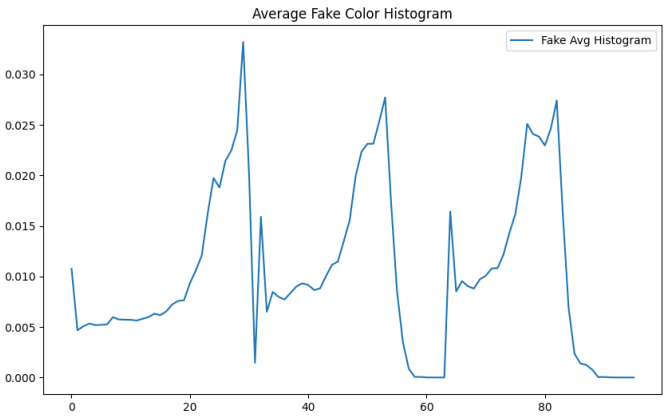

Fig. 2. Average Deepfake image Colour Histogram

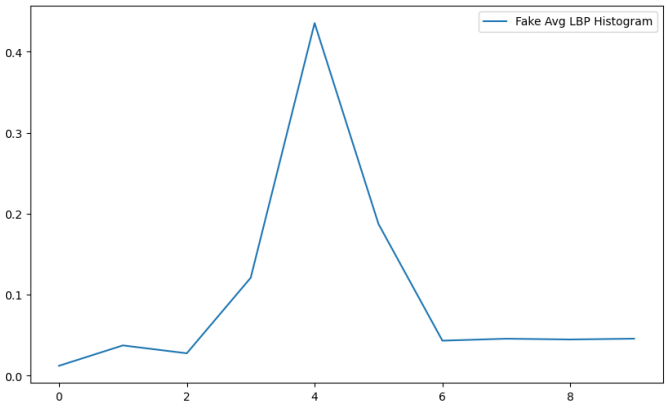

Fig. 3. Average Deepfake LBP

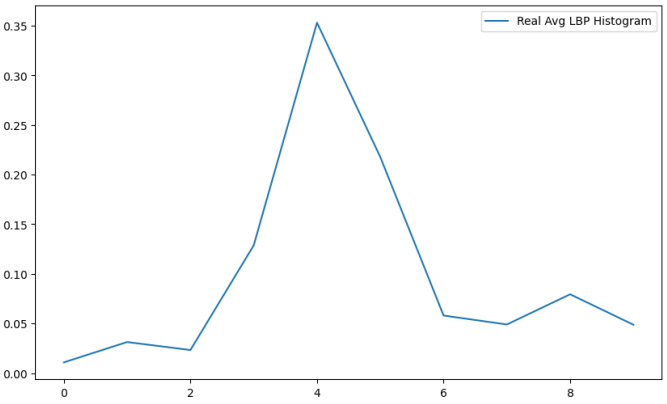

Fig. 4. Average Real image LBP

D. Neural Network Architectures

Two models were implemented and tuned using Keras Tuner.

1) Model 1: Custom CNN: The base CNN used variable convolutional block configurations defined as seen in the image. 5

Adam optimizer with a learning rate sampled logarithmically between 10^{-4} and 10^{-2} was used.

2) Model 2: ResNet-style CNN: The second model used progressive filter growth (32→64→128→256). Batch normalization and dropout were interleaved for regularization. Global average pooling replaced flattening to reduce overfitting.

E. Hyperparameter Tuning and Early Stopping

Hyperparameters were optimized using RandomSearch in Keras Tuner:

Listing 1. Keras Tuner setup

```
tuner = kt.RandomSearch(
    build_model,
    objective='val_recall',
    max_trials=5,
    directory='tuner/model_1',
    project_name='deepfake_cnn'
)
```

Early stopping monitored validation recall with a threshold criterion:

$$\text{Stop training if val\_recall} > 0.98$$

This ensured minimal overfitting while prioritizing false-negative reduction.

III. Results

A. Training Performance

Both models achieved stable convergence after training for 20 epochs. Model 1 (simple CNN) attained 94.2% recall, while Model 2 (ResNet-style) achieved 97.7%.

B. Evaluation Metrics

Model performance was assessed using accuracy, recall, and F1-score. Recall was the primary metric to prioritize correct fake detections.

TABLE I
Model Performance Comparison

Model	Acc (%)	Recall (%)	F1
Simple CNN	**90.2**	94.2	**91.68**
ResNet-style CNN	74.79	**97.7**	81.59

C. Error Analysis

The first five misclassified samples were visualized to identify failure modes. Common errors included:

- Shadow artifacts or lighting inconsistency
- Compression artifacts creating false texture cues

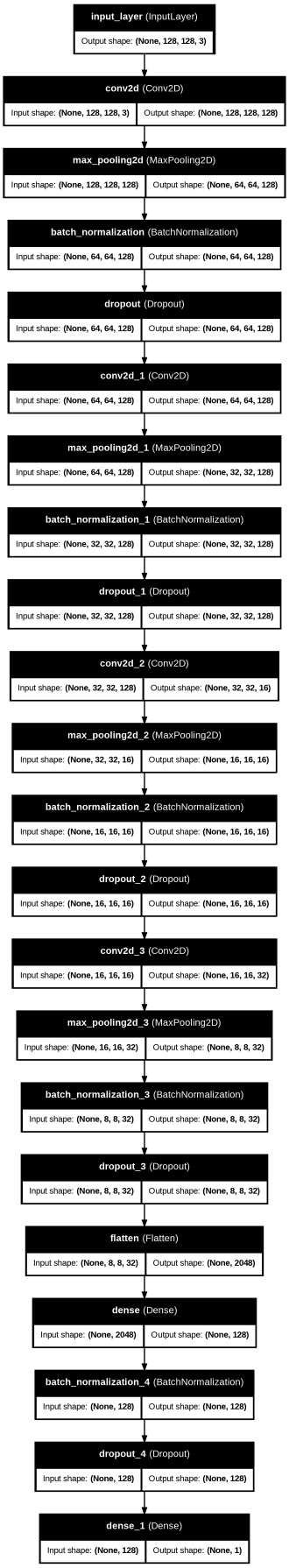

Fig. 5. Model 1 Architecture

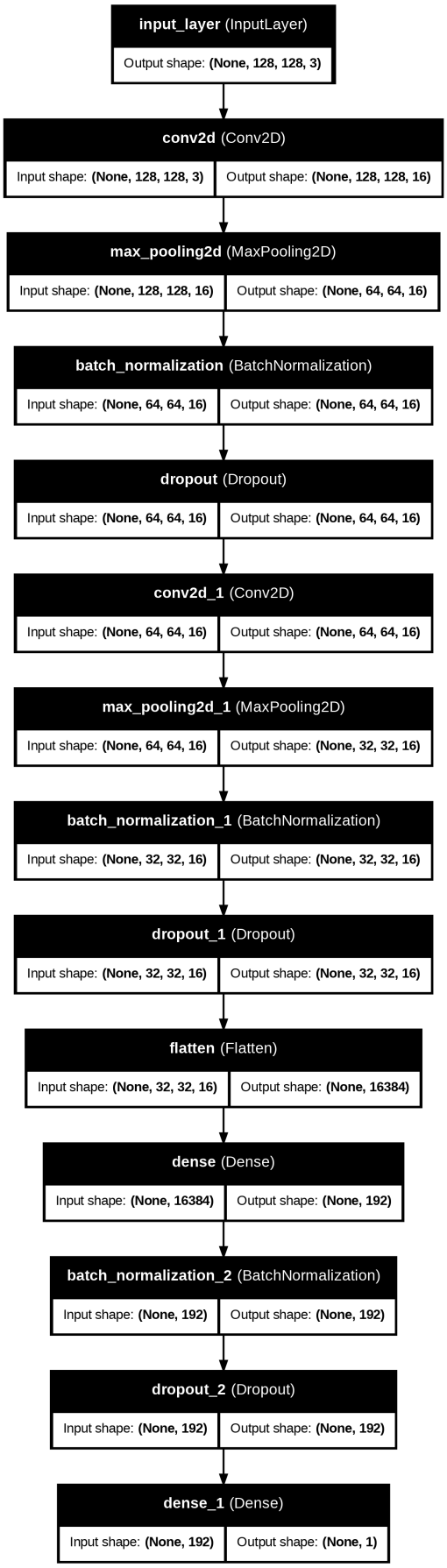

Fig. 6. Model 2 Architecture

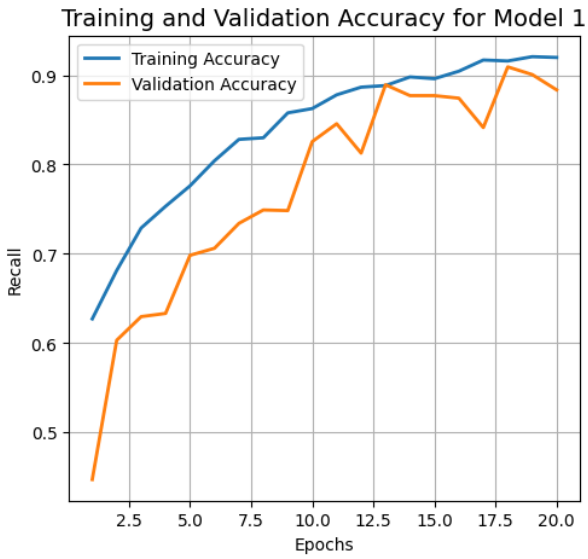

Fig. 7. Training and validation accuracy curves for model 1.

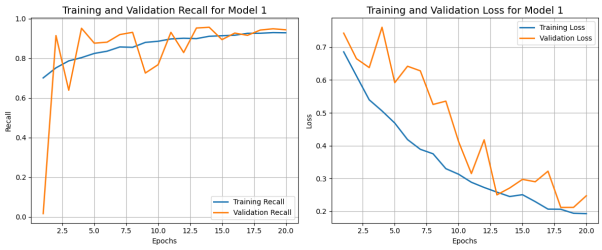

Fig. 8. Training and validation Recall and Loss curves for model 1.

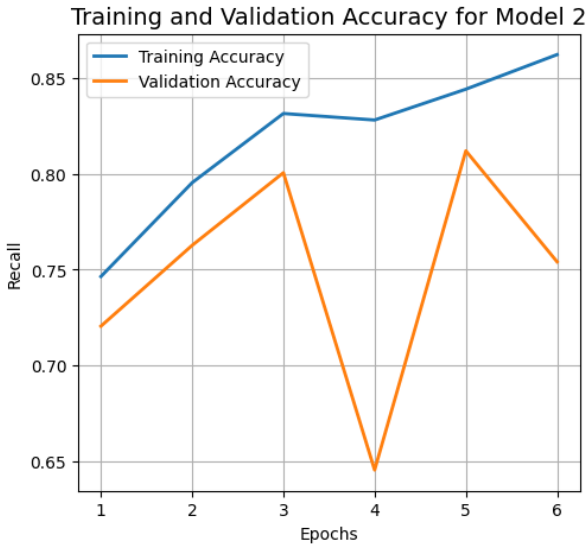

Fig. 9. Training and validation accuracy curves for model 2.

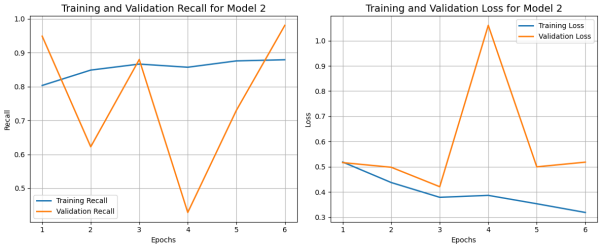

Fig. 10. Training and validation Recall and Loss curves for model 2.

Fig. 11. Examples of misclassified images (T, P metric for classification) by Model 1.

IV. Discussion

The results indicate that augmenting CNNs withfeature maps provides measurable benefits in classification performance. LBP and colour histograms contributed to the model's understanding of facial texture consistency, an important differentiator between real and GAN-generated images. The residual model's higher recall underscores the importance of hierarchical and progressive feature extraction, allowing the network to capture subtle texture transitions and synthesis artifacts. Recall is also the more important metric for model performance in cases of fraud detection or deepfake detection, where finding the measure of how many of the actual positive cases a model correctly identifies is important.

V. Challenges and Limitations

Due to limited hardware resources on Colab, the dataset had to be constrained, reducing the statistical robustness of results. Additionally:

- Training on larger subsets frequently caused CPU memory overflows.
- The models were trained on compressed images, which may obscure generative artifacts.

VI. Conclusion and Future Work

This study demonstrates that deepfake detection can benefit from a dual-feature pipeline integrating handcrafted features and deep CNN representations. The ResNet-inspired model outperformed the base CNN, achieving recall above 97% without early stopping.

Fig. 12. Examples of misclassified images (T, P metric for classification) by Model 2.

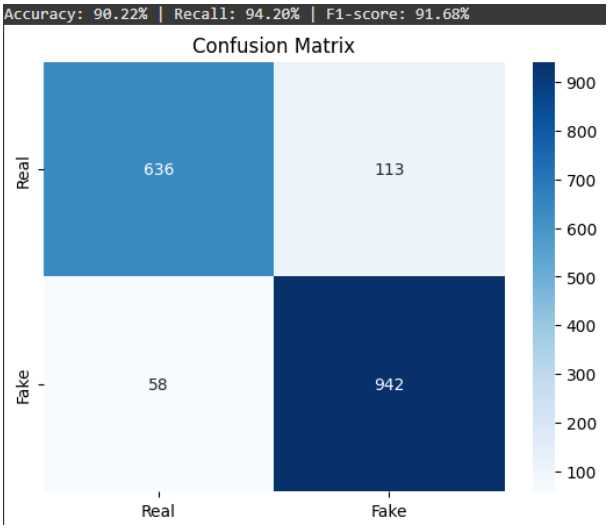

Fig. 13. Confusion Matrix - Model 1 (CNN backbone)

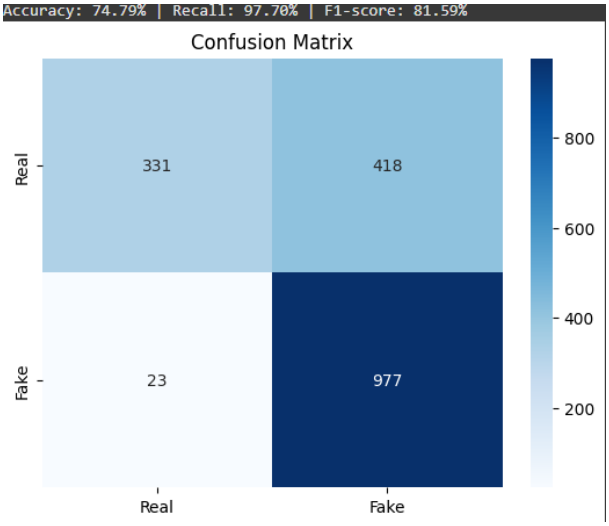

Fig. 14. Confusion Matrix - Model 2 (ResNet-style)

Future work will focus on:

- Scaling to larger datasets
- Incorporating transformer-based backbones (ViT, Swin)
- Exploring multimodal fusion (audio + video cues)

Ultimately, combining CNN-based local feature extraction with attention-driven global reasoning offers a promising path for robust deepfake detection systems.

REFERENCES

[1] M. R. Islam, M. A. I. Rakib, A. Sahin Afridi, and M. M. Islam, "Comprehensive deepfake detection dataset: Real and synthetic frames from roop and akool AI technologies," 2025.