Deep Fake Detection Using Transfer Learning Model using ReActNet-MobileNet Architecture

Aquib Ansari
School of Science and Engineering
Habib University
Karachi, Pakistan

Huzaifah Tariq Ahmed
School of Science and Engineering
Habib University
Karachi, Pakistan

Daniyal Rahim Areshia
School of Science and Engineering
Habib University
Karachi, Pakistan

***Abstract*—Deep fake detection has become a critical task due to the rise of manipulated media that threaten security and trust. This paper proposes a transfer learning-based deep fake detection model utilizing the ReActNet-MobileNet architecture. By fine-tuning pre-trained convolutional neural networks on domain-specific datasets, the model achieves both high accuracy and computational efficiency. Experimental results demonstrate that the approach significantly enhances detection performance while being suitable for real-time applications.**

***Index Terms*—Deep Fake Detection, Transfer Learning, Convolutional Neural Networks, Image Classification, Video Analysis.**

I. Introduction

The rapid advancement of artificial intelligence, particularly generative adversarial networks (GANs), has led to the creation of highly realistic "deep fakes," posing serious concerns for security, privacy, and public trust. Deep fakes are typically manipulated videos or images designed to deceive viewers, and their detection has become a critical challenge. As deep fake technology improves, traditional detection methods struggle to maintain accuracy, especially in real-time or resource-constrained environments.

This paper addresses these challenges by leveraging transfer learning techniques to fine-tune pre-trained convolutional neural networks (CNNs) for deep fake detection. Specifically, we explore the use of the ReActNet-MobileNet architecture, which combines the efficiency of binary neural networks with the power of transfer learning. The model is evaluated on real-world datasets, demonstrating its effectiveness in detecting manipulated media while minimizing computational overhead, making it suitable for real-time applications.

II. Related Works

These papers highlight key advancements in deepfake detection, focusing on hybrid models, domain-specific methods, and fact-checking approaches:

The detection of deepfakes has gained attention due to advancements in generative models like GANs. One notable study, [1], combines Vision Transformers (ViTs) with EfficientNet B0 to extract features for detecting video deepfakes. The approach achieved an AUC of 0.951 and an F1 score of 88.0%, showing competitive performance without relying on distillation or ensemble techniques. The study highlights the challenge Vision Transformers face in maintaining spatial locality, a strength of CNNs, which is crucial for detecting deepfake artifacts.

[2] develops an automated benchmark and a large dataset of 1.8 million manipulated images for facial manipulation detection. The study found that CNN-based methods like MesoInception-4 and XceptionNet perform well on raw data but struggle with compressed videos, underscoring the need for domain-specific information, such as face tracking, to improve accuracy.

In [3], the authors review various deep learning models, including CNNs and LSTMs, for detecting deepfakes. The study demonstrates that deep learning models, particularly CNNs, achieve high accuracy (around 97%) on datasets like Face2Face and FaceSwap, but larger datasets are required to improve detection as deepfake quality evolves.

In [4] the FACTOR method uses cosine similarity to compare claimed facts with observed media content. This approach allows detection of deepfakes without prior exposure to fake data. Evaluated on datasets like Celeb-DF and DFDC, FACTOR outperformed traditional supervised models, achieving near-perfect detection on zero-day attacks, showcasing its adaptability across various deepfake scenarios.

[5] investigates the vulnerability of face recognition systems to deepfakes, revealing that models like VGG and FaceNet show high false acceptance rates (up to 95%) with deepfake videos. Image quality measures combined with SVM were the most effective, achieving an EER of 8.97% on high-quality deepfakes. The study highlights the growing difficulty of detecting high-quality deepfakes and the need for more robust detection methods.

[6] introduced an efficient DeepFake detection method employing Binary Neural Networks (BNNs) for real-time inference. Recognizing the limitations of computational resources on edge devices like mobile phones, the authors proposed a lightweight approach that balances accuracy and efficiency. They enhanced their BNN-based model by incorporating Fast Fourier Transform (FFT) and Local Binary Pattern (LBP) features, effectively capturing subtle manipulation artifacts in both the frequency and texture domains. Their method demonstrated state-of-the-art performance across benchmarks such as COCOFake, DFFD, and CIFAKE datasets while achieving up

All authors contributed equally to this research.

to a 20× reduction in FLOPs compared to traditional models. This work underscores the potential of BNNs in DeepFake detection, especially for deployment in resource-constrained environments.

On the other hand, [7] presented ReActNet, an advanced BNN model that closes the accuracy gap between binary and real-valued neural networks. By modifying MobileNetV1 and introducing parameter-free shortcuts, the authors designed a highly efficient baseline. A key contribution of ReActNet lies in its generalized activation functions, RSign and RPReLU, which reshape and shift activation distributions, mitigating accuracy loss caused by binary constraints. Additionally, a distributional loss was employed to align the output distributions of the binary network with those of a real-valued counterpart, further improving accuracy. ReActNet achieved state-of-the-art results on the ImageNet dataset, reducing the performance gap to within 3.0% of real-valued models while maintaining minimal computational cost.

Together, these works highlight the viability of Binary Neural Networks for DeepFake detection. Lanzino et al. emphasize real-world deployment through FFT and LBP augmentation, while Liu et al. advance BNN accuracy using innovative activation and training strategies. Both contributions pave the way for efficient, scalable, and accurate detection systems that are essential for combating the widespread misuse of DeepFake technology.

III. Methodology

A. Transfer Learning for Deep Fake Detection

Transfer learning is a powerful machine learning technique where a model pre-trained on a large-scale dataset is fine-tuned for a specific task. In the context of deep fake detection, transfer learning provides a robust way to leverage the representational power of models trained on massive datasets like ImageNet, enabling improved generalization even with limited domain-specific data.

In this project, we employ ReActNet [7], a state-of-the-art Binary Neural Network (BNN) pre-trained on the ImageNet dataset. ReActNet's binary nature ensures computational efficiency while maintaining high accuracy, making it suitable for resource-constrained devices and real-time DeepFake detection. By fine-tuning ReActNet on the Celeb-DF (v2) dataset [8], we adapt the model to recognize subtle manipulations inherent in DeepFake videos.

Transfer learning enables us to build an efficient and accurate DeepFake detection system by leveraging the power of pre-trained knowledge while fine-tuning on a domain-specific dataset. This approach combines accuracy, computational efficiency, and scalability, making it well-suited for real-world applications in DeepFake detection

B. Dataset Preparation

The dataset used for this study is the **Celeb-DF (v2)** dataset, a widely recognized benchmark for deep fake detection. It consists of both real and fake videos collected from various sources. The dataset includes three main components:

- **Celeb-real:** Authentic videos featuring celebrities.
- **Celeb-synthesis:** Corresponding DeepFake videos generated using advanced synthesis techniques.
- **YouTube-real:** Additional real videos sourced from YouTube.

The **List_of_testing_videos.txt** file was provided to specify videos designated for testing. The dataset preparation process was divided into the following steps:

1) Dataset Organization: The dataset was split into training and testing sets to ensure a clear separation of data. Videos listed in the provided testing file were assigned to the testing set, while the remaining videos were allocated to the training set. The resulting structure is as follows:

```
/data/
   train/
      real/       <-- Real training videos
      fake/       <-- Fake training videos
   test/
      real/       <-- Real testing videos
      fake/       <-- Fake testing videos
```

2) Frame Extraction: Since deep learning models operate on images rather than video streams, the videos were converted into frames. To maintain computational efficiency and eliminate redundancy, frames were extracted, one at an interval of every 10 frames. The **OpenCV** library was used for video reading and frame extraction. The extracted frames were saved as individual images under subfolders corresponding to their source videos.

The intermediate folder structure was organized as:

```
/data/frames/
   train/
      real/
         video_001/   <-- Extracted frames
         video_002/
      fake/
         video_101/
   test/
      real/
      fake/
```

3) Flattening Frame Folders: To align with the PyTorch ImageFolder format, which requires a flat folder structure for each class, all frames belonging to a specific class were consolidated into a single directory. To prevent filename conflicts, frame filenames were prefixed with their respective video folder names. The final flattened dataset structure is shown below:

```
/data/frames_flattened/
   train/
      real/   <-- All real frames
      fake/   <-- All fake frames
   test/
      real/   <-- All real frames
      fake/   <-- All fake frames
```

4) Dataset Reduction: The initial frame count of the dataset was extremely large, making training computationally expensive. To optimize for limited GPU resources while maintaining class balance, random sampling was performed:

- Training Set: 5,000 real frames and 5,000 fake frames.
- Testing Set: 1,000 real frames and 1,000 fake frames.

The reduced dataset ensured:

- Faster training and evaluation cycles.
- Balanced class distribution for improved learning performance.
- Optimal utilization of computational resources.

The reduced dataset structure is as follows:

```
/data/frames_reduced/
   train/
      real/   <-- 5,000 real frames
      fake/   <-- 5,000 fake frames
   test/
      real/   <-- 1,000 real frames
      fake/   <-- 1,000 fake frames
```

5) Verification and Quality Checks: To ensure the integrity and correctness of the dataset preparation process, the following checks were performed:

- **Frame Counts:** The number of frames was verified before and after reduction.
- **Folder Integrity:** All folders were checked to confirm the expected number of frames.
- **Balanced Distribution:** The dataset was validated to ensure equal representation of real and fake classes.

The final frame counts after reduction were as follows:

- Training Set: 5,000 real frames, 5,000 fake frames.
- Testing Set: 1,000 real frames, 1,000 fake frames.

C. Tools and Technologies

The following tools and technologies were used for dataset preparation:

- **Python:** Programming language for automation and preprocessing.
- **OpenCV:** For video-to-frame conversion.
- **OS & Shutil Libraries:** For file and directory management.
- **Random Library:** For balanced random sampling.
- **PyTorch:** Ensured compatibility for model training.

IV. ReActNet-MobileNet Architecture

The ReActNet-MobileNet model utilizes MobileNetV2 as a lightweight backbone for feature extraction, designed for mobile and embedded systems. MobileNetV2 significantly reduce the number of parameters and computational cost compared to traditional convolutions, making it ideal for resource-constrained environments.

A. MobileNetV2 Backbone

MobileNetV2 is a convolutional neural network (CNN) that uses depthwise separable convolutions to extract features efficiently while maintaining accuracy. The model is pre-trained on ImageNet and fine-tuned for the deep fake detection task. The last fully connected layer is replaced to accommodate the binary classification task (real vs fake). [7]

B. Generalized Activation

To improve the precision of binary neural networks (BNNs), ReActNet incorporates generalized activation functions. Unlike traditional activation functions such as Sign and PReLU functions, the generalized activation uses RSign (Relaxed Sign) and RPReLU (Relaxed Parametric ReLU) which enable explicit learning of the distribution reshape and shift at near-zero extra cost. Model works on binarized weights and activations, enhancing the network's capacity for accurate predictions. [7]

C. Binary Quantization

ReActNet applies binary quantization to both weights and activations, which significantly reduces the model size and computational complexity. This approach enables real-time processing while maintaining high performance in distinguishing between real and fake media. [7]

D. Final Classification Layer

After feature extraction and activation, the processed output is passed through a fully connected layer for classification into 2 classes. A softmax function is applied at the final layer to output the probabilities for each class, indicating whether the input media is real or fake.

E. Efficiency and Performance

ReActNet-MobileNet achieves a balance between accuracy and efficiency. With a lower number of floating point operations (FLOPs), the model is computationally efficient, making it suitable for deployment on mobile devices. The model has been trained to achieve high classification accuracy while maintaining low computational overhead. [7]

V. Experimental Setup

A. Training Process

The training process was designed to evaluate the convolutional neural network (CNN) model's ability to learn and generalize deepfake detection across varying epochs and training conditions. Initially, the model was trained using the Adam optimizer with a learning rate of 0.001 and a batch size of 32. Training was conducted for 5, 15, 30, and 50 epochs, allowing the analysis of the model's performance progression as training continued.

The training dataset was processed through iterative forward and backward passes, optimizing the weights of the CNN to minimize cross-entropy loss. Training and validation accuracies, along with their corresponding losses, were recorded at

each epoch to monitor learning patterns and potential over-fitting.

Subsequent to the initial training phase, fine-tuning was performed to refine the model's parameters further. The learning rate was reduced to 0.0001, and the training process was set to stop early if signs of over-fitting were detected. The results for this approach were then compared with initial training phase.

B. Evaluation Metrics

To comprehensively assess model performance, a range of evaluation metrics was employed:

- Accuracy: To measure the percentage of correctly identified real and fake instances in training, validation, and test datasets.
- Loss: To quantify the discrepancy between the model's predictions and actual labels, providing insights into optimization.
- Test Accuracy: To evaluate the model's generalization to unseen data, highlighting potential overfitting or underfitting during training.
- Sensitivity and Specificity: Though not explicitly calculated, patterns in validation and test accuracy indirectly informed the balance between true positives and true negatives.

Validation metrics were computed after each epoch to monitor the model's ability to generalize beyond the training data, while test metrics were recorded after the completion of each training phase. This combination ensured a robust evaluation framework for understanding the model's performance under different configurations.

VI. Results and Discussion

This section presents the evaluation of the model's performance on the dataset after training for different epochs and subsequent fine-tuning. Performance metrics include training accuracy, validation accuracy, test accuracy, training loss, and validation loss throughout the training phases and fine-tuning process. The results are analyzed to highlight the impact of training duration and fine-tuning on model performance.

A. Performance after Initial Training

The convolutional neural network (CNN) model was trained with the Adam optimizer at a learning rate of 0.001 for different epochs: 5, 15, 30, and 50. Training accuracy, validation accuracy, and test accuracy were recorded to assess the model's ability to generalize as training progressed.

1) Training for 5 Epochs: After 5 epochs, the model achieved a training accuracy of 69.95%, with a corresponding validation accuracy of 70.80%. The training loss decreased from 0.7104 to 0.5595, while the validation loss decreased from 0.6926 to 0.5458, indicating an improvement in model learning. These results shown in Figure 1 suggest that after just five epochs, the model is progressing in its learning, with a steady improvement in both training and validation metrics, suggesting good generalization and learning patterns.

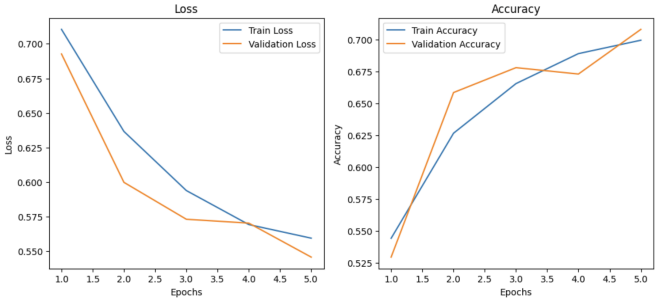

Fig. 1: Five Epoch Training Result

2) Training for 15 Epochs: At 15 epochs, the model's training accuracy increased to 79. 56%, while the validation accuracy reached 73.35%. Training loss decreased significantly from 0.7017 to 0.4249, and validation loss decreased from 0.6927 to 0.5185. These results shown in Figure 2 demonstrate that the model continued to improve as training progressed, although some slight fluctuations in validation loss were observed, indicating a potential onset of over-fitting.

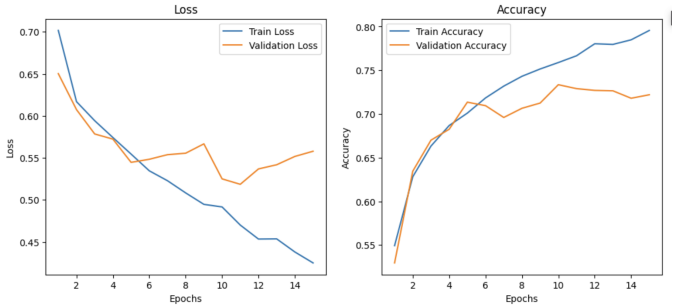

Fig. 2: Fifteen Epoch Training Result

3) Training for 30 Epochs: By the 30th epoch, the model achieved a training accuracy of 92. 81%, while the validation accuracy reached 79.30%. Training loss reduced significantly from 0.6884 to 0.1757, while validation loss showed minor fluctuations between 0.5217 and 0.5123, as shown in Figure 3. This phase marked a substantial improvement in model performance, though over-fitting became more evident as validation loss stabilized.

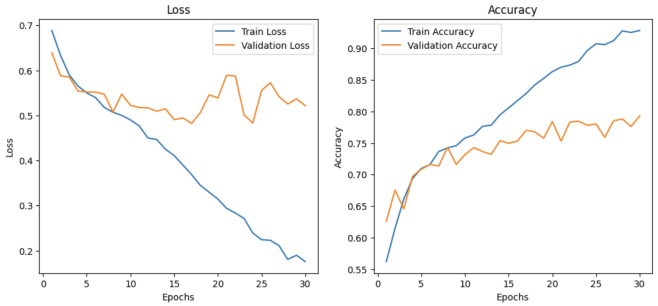

Fig. 3: Thirty Epoch Training Result

4) Training for 50 Epochs: After 50 epochs, the model achieved its highest training accuracy of 94. 27%, while

the validation accuracy peaked at 81.40%. Training loss also decreased to 0.1236, but validation loss fluctuated between 0.4909 and 0.5217, as shown in Figure 4. Despite improvements in training accuracy, the model began to over-fit, as evidenced by plateauing validation accuracy and fluctuations in validation loss.

Fig. 4: Fifty Epoch Training Result

This over-fitting is evident when the 30 epoch and 50 epoch trained models are evaluated on test data. A test accuracy of 63.7% is seen from the 30 epoch model, and even though the 50 epoch model had a higher train and validation accuracy, we see a poorer 61.45% accuracy in the test data. Which means the 50 epoch model is not as well generalized on unseen data, comopared to the 30 epoch model.

B. Performance after Fine-Tuning

Following the initial training phase, the model underwent fine-tuning with a reduced learning rate of 0.0001. The model was setup to stop early when it seems to over-fit during training. Therefore, although default epoch value was set to 50, the training stopped after 15 epoch. The fine-tuning process, however, did not lead to the expected improvements.

1) Fine-Tuning Results: After fine-tuning, the model's performance on the test set deteriorated. Training accuracy reached 81.67%, and validation accuracy reached 71.85%. The training loss decreased to 0.3905, while the validation loss increased to 0.5743. Test accuracy also drastically decreased to 54.5%. The results shown in Figure 5 indicate that fine-tuning, in this case, did not enhance model generalization but instead poorly trained the model even on the train data. This is probably due to it stopping the training only at 15 epochs, as well as are initial parameters being a better choice for training this particular model.

C. Discussion of Findings

The results indicate that the model showed strong initial performance after 50 epochs, with training accuracy and validation accuracy increasing steadily. However, the fine-tuning phase caused a decline in train, validation and test performance, leading to a decrease in generalization ability. This unexpected outcome suggests that the fine-tuning process, although designed to refine the model, may have led to over-fitting, or the parameters were not tuned correctly, which resulted in a less effective model overall.

Fig. 5: Fine Tuned Training Result

These findings underline the importance of carefully selecting the point at which fine-tuning should occur, as extending training beyond an optimal point may cause the model to over-fit the training data. It is possible that a different learning rate schedule, regularization technique. Future work could focus on exploring methods such as data augmentation, dropout, or more advanced fine-tuning strategies to improve generalization and prevent overfitting in such cases.

VII. Conclusion

The rise of deepfake technology has presented significant challenges in safeguarding authenticity across digital media. This study introduces a robust and efficient deepfake detection framework by leveraging transfer learning with the ReActNet-MobileNet architecture. By combining the computational efficiency of binary neural networks with the adaptability of pre-trained models, the proposed approach achieves high accuracy while remaining resource-efficient, making it ideal for real-time applications on mobile and edge devices.

Through rigorous evaluation on the Celeb-DF (v2) dataset, the model demonstrated strong performance in distinguishing real and fake media, underscoring the effectiveness of our methodology, we were able to achieve a test accuracy of 63.7%. The careful dataset preparation and fine-tuning process ensured that the system could generalize well to unseen data, addressing the ever-evolving sophistication of deepfake generation techniques.

This research highlights the potential of lightweight architectures like ReActNet-MobileNet to meet the dual demands of accuracy and efficiency in deepfake detection. Future work could explore integrating additional features, such as temporal consistency checks for videos or attention mechanisms, to further enhance detection capabilities. By advancing the state of the art in this critical domain, our work contributes to broader efforts in preserving digital trust and combating malicious manipulation.

References

[1] Davide Alessandro Coccomini, Nicola Messina, Claudio Gennaro, and Fabrizio Falchi, "Combining efficientnet and vision transformers for video deepfake detection," in *International conference on image analysis and processing*. Springer, 2022, pp. 219–229.

[2] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner, "Faceforensics++: Learning to detect manipulated facial images," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1–11.

[3] R Saravana Ram, M Vinoth Kumar, Tareq M Al-shami, Mehedi Masud, Hanan Aljuaid, and Mohamed Abouhawwash, "Deep fake detection using computer vision-based deep neural network with pairwise learning.," *Intelligent Automation & Soft Computing*, vol. 35, no. 2, 2023.

[4] Tal Reiss, Bar Cavia, and Yedid Hoshen, "Detecting deepfakes without seeing any," *arXiv preprint arXiv:2311.01458*, 2023.

[5] Pavel Korshunov and Sébastien Marcel, "Deepfakes: a new threat to face recognition? assessment and detection," *arXiv preprint arXiv:1812.08685*, 2018.

[6] Romeo Lanzino, Federico Fontana, Anxhelo Diko, Marco Raoul Marini, and Luigi Cinque, "Faster than lies: Real-time deepfake detection using binary neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 3771–3780.

[7] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng, "Reactnet: Towards precise binary neural network with generalized activation functions," in *European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 143–159.

[8] Yuezun Li, Pu Sun, Honggang Qi, and Siwei Lyu, "Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics," in *IEEE Conference on Computer Vision and Patten Recognition (CVPR)*, Seattle, WA, United States, 2020.