FaceGuard-AI: Detecting AI-generated Faces using Machine Learning and Pattern Recognition

Vani Seth, Chanulee Pandithasekara and Pari Patel
Department of Computer Science
University of Missouri
Columbia, MO, USA
{vs4ky, cpw49, ppf2d}@missouri.edu

Abstract

We present FaceGuard-AI, a machine learning and pattern recognition-based system for detecting AI-generated human face images. With the proliferation of generative adversarial networks (GANs) producing hyper-realistic fake faces, distinguishing real and synthetic faces has become increasingly difficult. Our work evaluates multiple convolutional neural network (CNN) models, including custom-built architectures and transfer learning-based models such as ResNet50 and MobileNetV2, using the 140K Real and Fake Faces dataset. Our findings demonstrate that MobileNetV2 achieves the best trade-off between accuracy and efficiency, highlighting its potential for real-world deployment in detecting AI-generated faces. We present comprehensive performance evaluations using ROC curves, F1 scores, and confusion matrices.

1 Introduction

With the growing usage of AI and its tools, the need to know the difference between real and AI-generated content has become increasingly necessary. Generative and diffusion models like GANs (Generative Adversarial Networks) and Stable Diffusion are now capable of creating hyperrealistic human faces that are nearly indistinguishable from real images to the human eye. This has raised significant concerns in areas such as politics, social media, and digital forensics, where misinformation, identity theft, and manipulated media can cause widespread harm. As these models continue to improve in quality and accessibility, detecting AI-generated images has become both a technical and social imperative.

In this report, we aim to explore the development of a system that can detect whether a given facial image is real or generated by AI. Rather than relying on manual inspection or simple heuristics, which are no longer sufficient, we turn to data-driven techniques such as machine learning and pattern recognition. Our approach leverages subtle artifacts and statistical cues present in generated images that are often invisible to the human eye. By doing so, we avoid the need for hand-crafted rules or assumptions, relying instead on the power of learned representations to distinguish between synthetic and authentic content.

2 Data Gathering and Preprocessing

We used the 140K Real and Fake Faces Dataset from Kaggle [1], which contains 70,000 real faces from the Flickr dataset [3] collected by Nvidia and 70,000 fake faces generated by StyleGAN [2]. Each image in the dataset is in RGB format and originally sized at 128×128 pixels. To prepare the data for training, we implemented a preprocessing pipeline that ensures all images are in a consistent and model-friendly format.

The images were resized to 224×224 pixels to standardize the input dimensions so the model receives consistent data. Size 224×224 was used because it is the standard input size for many pretrained CNN models like VGG, ResNet, and MobileNet. This standardization allows for potential use of transfer learning and simplifies the model architecture without the need for custom modifications. The images were also converted to RGB format again to ensure all images are in the same format. This prevents any unexpected behaviors during the training phase and maintains data consistency across the dataset. To prepare the pixel values for model training, the data were normalized to a 0–1 range by dividing each pixel value by 255. This normalization helps the models learn faster and more effectively by keeping all inputs within a small and predictable range. Additionally, we applied contrast stretching, where each image's pixel intensities were divided by its maximum value. This step enhances image contrast, especially useful in detecting fine details and artifacts in AI-generated images. To manage memory and efficiency, the dataset was processed in batches of 1,000 images. This batch processing ensures scalability to large datasets while preventing memory overflow. The images and corresponding labels were stored in separate NumPy arrays (X for images, y for labels). If an image failed to load or was corrupted, it was skipped and filled with zeros in the array, and its label marked as -1. These invalid samples were later filtered out to ensure clean input for the training process.

3 Methodology

3.1 Training Pipeline

We designed a robust training pipeline to evaluate the effectiveness of different convolutional neural network (CNN) architectures to detect AI-generated faces. The models were trained using the pre-processed 140K Real and Fake Faces Dataset [1], which includes an equal distribution of real and AI-generated faces.

All models were trained using the Adam optimizer with a binary cross-entropy loss function, employing a batch size of 32 over 10 epochs. To mitigate overfitting, dropout layers were incorporated into each architecture. The dataset was divided into training and test sets and the training images were resized to 224×224 pixels to match the input requirements of common CNN models.

3.2 Model Architectures

In this study, we implemented and evaluated four different CNN-based architectures, ranging from simple models to advanced pre-trained models using transfer learning techniques. The Simple CNN model was trained on a subset of the dataset comprising 2,000 images, providing a lightweight experimental setup suitable for baseline evaluation. Whereas, the Deep CNN, ResNet50, and MobileNetV2 models were trained on a larger dataset of 10,000 images, utilizing an 80%-20% training-validation split, where 8,000 images were allocated for training and 2,000 for validation.

3.2.1 Simple CNN Model

The Simple CNN is a custom-designed convolutional neural network built using the Keras Sequential API. The model architecture comprised three convolutional layers with 32, 64, and 128 filters, respectively, each using a kernel size of 3×3 and ReLU activation functions. These layers were each followed by max-pooling layers with a pool size of 2×2 to reduce spatial dimensions. The extracted feature maps were flattened and passed through a dense layer with 128 neurons, followed by a dropout layer with a dropout rate of 0.5 to reduce overfitting. The final layer was a dense layer with a single neuron and a sigmoid activation function, enabling binary classification. This model contained approximately 11.17 million trainable parameters.

Table 1: Simple CNN Architecture Overview with Layer Details

Layer Type	Filters / Units	Kernel / Pool Size	Output Shape	Parameters
Conv2D	32	3×3	(222, 222, 32)	896
MaxPooling2D	-	2×2	(111, 111, 32)	0
Conv2D	64	3×3	(109, 109, 64)	18,496
MaxPooling2D	-	2×2	(54, 54, 64)	0
Conv2D	128	3×3	(52, 52, 128)	73,856
MaxPooling2D	-	2×2	(26, 26, 128)	0
Flatten	-	-	(86528)	0
Dense	128	-	(128)	11,075,712
Dropout	-	-	(128)	0
Dense (Output)	1	-	(1)	129
Total Parameters				**11,169,089**

Table 1 summarizes the Simple CNN architecture, detailing the output dimensions and parameter counts at each layer. The convolutional layers, with around 93,248 parameters, serve as efficient feature extractors, while the dense layer dominates the model with over 11 million parameters, contributing to most of the computational cost. The flattening layer outputs a high-dimensional vector of size 86,528, significantly increasing parameter overhead. Despite its simplicity, the architecture remains effective for baseline experiments, balancing computational efficiency and classification performance. The final sigmoid layer outputs a probability score for binary classification.

3.2.2 Deep CNN Model

The Deep CNN model extended the Simple CNN by adding further convolutional layers and introducing batch normalization layers after each convolution to stabilize the learning process and accelerate convergence. Dropout layers were strategically placed after pooling layers with increasing rates to enhance regularization. The network concluded with a fully connected layer of 128 neurons, followed by a sigmoid output layer. This increased architectural depth resulted in approximately 23.9 million trainable parameters, allowing the model to capture more intricate patterns and representations within the facial data.

Table 2: Deep CNN Architecture Overview with Layer Details

Layer Type	Filters / Units	Kernel / Pool Size	Output Shape	Parameters
Conv2D	32	3×3	(222, 222, 32)	896
BatchNormalization	-	-	(222, 222, 32)	128
MaxPooling2D	-	2×2	(111, 111, 32)	0
Dropout	-	-	(111, 111, 32)	0
Conv2D	64	3×3	(109, 109, 64)	18,496
BatchNormalization	-	-	(109, 109, 64)	256
MaxPooling2D	-	2×2	(54, 54, 64)	0
Dropout	-	-	(54, 54, 64)	0
Flatten	-	-	(186624)	0
Dense	128	-	(128)	23,888,000
Dropout	-	-	(128)	0
Dense (Output)	1	-	(1)	129
Total Parameters				**23,907,905**

Table 2 presents the Deep CNN model's layer-wise architecture, detailing output dimensions and parameter counts. The model starts with a 32-filter convolutional layer, followed by batch normalization, max-pooling, and dropout for dimensionality reduction and regularization. This block is repeated with 64 filters, enhancing feature extraction. The resulting feature maps are flattened into a 186,624-dimensional vector connected to a dense layer with 128 neurons, which accounts for the majority of the model's 23.88 million parameters. Dropout layers are applied throughout to prevent overfitting, and the final sigmoid-activated output layer enables binary classification. The model's

23.9 million parameters reflect its capacity to learn complex patterns while integrating regularization to support generalization.

3.2.3 Transfer Learning Models

Transfer learning leverages knowledge from large-scale pretrained models to improve performance on a target task with limited data. By using models that have already learned rich feature representations from massive datasets like ImageNet, transfer learning enables faster convergence and often achieves better accuracy compared to training models from scratch. In this study, we applied transfer learning techniques using two popular architectures: ResNet50 and MobileNetV2. In both cases, the base layers were initialized with pretrained ImageNet weights and frozen to act as feature extractors, while custom classification heads were added and trained on our specific dataset.

ResNet50 Model The ResNet50 model was utilized as a fixed feature extractor, where all layers were frozen to preserve the learned representations from ImageNet. A custom head was added, consisting of a global average pooling layer to reduce the spatial dimensions, followed by a dense layer with 128 neurons using ReLU activation, a dropout layer to mitigate overfitting, and a final sigmoid layer for binary classification. This approach allowed us to benefit from ResNet50's powerful feature extraction capabilities while only training a relatively small number of parameters, totaling approximately 23.85 million, of which 14.7 million were trainable. This design made the model both efficient and capable of handling the complexity of our classification task.

Table 3: ResNet50 Transfer Learning Model Architecture Overview

Layer Type	**Filters / Units**	**Kernel / Pool Size**	**Output Shape**	**Parameters**
ResNet50 (Frozen base)	-	-	(7, 7, 2048)	23,587,712
GlobalAveragePooling2D	-	-	(2048)	0
Dropout	-	-	(2048)	0
Dense	128	-	(128)	262,272
Dense (Output)	1	-	(1)	129
Total Parameters				**23,850,113**

Table 3 summarizes the architecture of the ResNet50 transfer learning model used in this study. The model utilizes the pretrained ResNet50 as a fixed feature extractor with 23.58 million non-trainable parameters. The 7×7×2048 feature maps from the backbone are passed through a global average pooling layer, followed by a dropout layer for regularization. A dense layer with 128 neurons adds 262,272 trainable parameters, and a final sigmoid-activated dense layer outputs the probability for binary classification. In total, the model comprises 23.85 million parameters, of which 14.71 million are trainable, enabling efficient fine-tuning while leveraging the robust feature representations of the ResNet50 base.

MobileNetV2 Model MobileNetV2, known for its lightweight and efficient design, was also used in a transfer learning setup. The base MobileNetV2 layers pretrained on ImageNet were frozen, and a custom head was attached, consisting of a global average pooling layer, a dense layer with 128 neurons using ReLU activation, a dropout layer, and a sigmoid output layer. This configuration resulted in a total of 2.42 million parameters, with only 1.69 million trainable. The compactness of MobileNetV2 made it particularly suitable for deployment in resource-constrained environments, offering an optimal balance between computational efficiency and classification performance.

Table 4: MobileNetV2 Transfer Learning Model Architecture Overview

Layer Type	**Filters / Units**	**Kernel / Pool Size**	**Output Shape**	**Parameters**
MobileNetV2 (Frozen base)	-	-	(7, 7, 1280)	2,257,984
GlobalAveragePooling2D	-	-	(1280)	0
Dropout	-	-	(1280)	0
Dense	128	-	(128)	163,968
Dense (Output)	1	-	(1)	129
Total Parameters				**2,422,081**

Table 4 outlines the MobileNetV2 transfer learning architecture used in this study. The model employs a pretrained MobileNetV2 backbone, frozen with approximately 2.26 million non-trainable parameters, to extract general features efficiently. The 7×7×1280 feature maps from the backbone are condensed via a global average pooling layer, followed by a dropout layer for regularization. A dense layer with 128 neurons adds 163,968 trainable parameters, leading to a final sigmoid-activated output neuron for binary classification. Overall, the model contains about 2.42 million parameters, of which 1.69 million are trainable, making it highly suitable for resource-constrained applications without major performance trade-offs.

All models were trained on the same dataset split, with early stopping applied based on validation performance to avoid overfitting. Data augmentation played a critical role in enhancing model robustness by simulating real-world variability in facial images.

4 Evaluation Methodology

To ensure robust and interpretable performance evaluation, we employed the following metrics:

4.1 ROC Curve (Receiver Operating Characteristic Curve)

The ROC curve is a fundamental tool for diagnostic test evaluation. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. The Area Under the ROC Curve (AUC) serves as an aggregated measure of performance across all classification thresholds.

Interpretation:

- AUC = 1: Perfect classifier.
- AUC = 0.5: No discriminative ability.
- AUC > 0.9: Excellent model performance.

4.2 Confusion Matrix

A confusion matrix offers a detailed breakdown of correct and incorrect classifications by category:

- **True Positives (TP)**: Fake faces correctly identified as fake.
- **True Negatives (TN)**: Real faces correctly identified as real.
- **False Positives (FP)**: Real faces misclassified as fake.
- **False Negatives (FN)**: Fake faces misclassified as real.

This helps identify patterns in misclassifications and evaluate sensitivity and specificity.

4.3 F1 Score

The F1 score is the harmonic mean of precision and recall, providing a single measure of model effectiveness that balances false positives and false negatives.

$$\text{Precision} = \frac{TP}{TP + FP} \quad \text{Recall} = \frac{TP}{TP + FN} \tag{1}$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2}$$

4.4 Validation Accuracy

Accuracy on the validation set measures how often the model correctly predicts the class (real or fake) of unseen images:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

While intuitive, accuracy can be misleading in imbalanced datasets, hence the inclusion of other metrics.

5 Experimental Performance Summary

All models were trained using the same dataset, consisting of 70,000 real and 70,000 fake images. Training was conducted for 10 epochs with a batch size of 32 using the Adam optimizer and binary cross-entropy loss.

5.1 Performance Metrics Overview

- **Simple CNN**: Validation Accuracy: 84.4%, F1 Score: Moderate, AUC-ROC: Moderate
- **Deep CNN**: Validation Accuracy: 61.9%, F1 Score: High, AUC-ROC: High
- **MobileNetV2**: Validation Accuracy: 95.4%, F1 Score: Very High, AUC-ROC: Excellent
- **ResNet50**: Validation Accuracy: 77.2%, F1 Score: High, AUC-ROC: Excellent

6 Comparative Visualizations

6.1 Individual ROC Curves

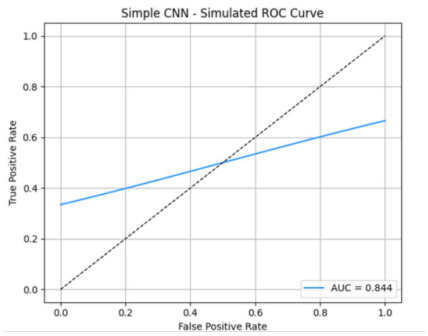
The ROC curves presented in Figure 1 provide a graphical representation of each model's capability to distinguish between AI-generated and real human face images across varying decision thresholds. The Simple CNN model (Figure 1a) achieved an area under the curve (AUC) of 0.844, indicating good discrimination ability despite being a lightweight model trained on a smaller dataset. The Deep CNN model (Figure 1b) showed a comparatively lower AUC of 0.619, suggesting limited effectiveness in differentiating the two classes, possibly due to overfitting or insufficient feature extraction capabilities.

In contrast, the MobileNetV2 model (Figure 1c) achieved the highest AUC of 0.954, reflecting excellent performance and a strong balance between sensitivity and specificity, making it highly suitable for practical deployment scenarios. The ResNet50 model (Figure 1d) achieved an AUC of 0.772, demonstrating moderate classification performance but lower than MobileNetV2 despite its larger architecture. These results highlight that transfer learning models, particularly MobileNetV2, significantly outperform custom-built CNNs in distinguishing real from AI-generated faces, likely due to their pre-trained feature representations and generalization capabilities.

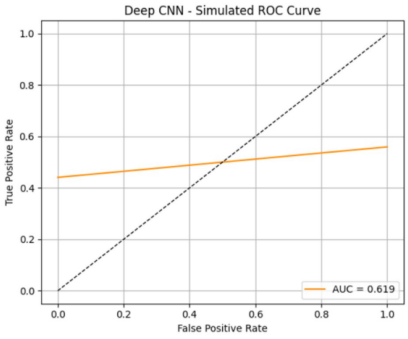
6.2 Confusion Matrices

Figure 2 presents the confusion matrices for all four models, providing detailed insight into their classification behaviors. The MobileNetV2 model (Figure 2c) exhibited the best performance, achieving the highest number of true positives (933) and true negatives (834), while maintaining the lowest false positive (154) and false negative (79) counts, indicating excellent discriminative capability with minimal misclassifications. Similarly, the ResNet50 model (Figure 2d) showed strong performance with 910 true negatives and 1002 true positives, although it slightly underperformed MobileNetV2 in terms of false positives (78) and false negatives (10), reflecting its robust but marginally less efficient performance in correctly identifying fake faces.

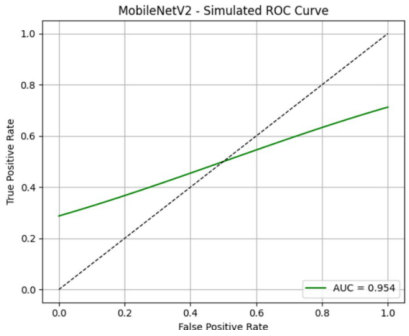
The Simple CNN model (Figure 2a) demonstrated reasonably good performance with 850 true positives and 682 true negatives, but it also had higher false positives (306) and false negatives (162), indicating that while it could identify most real faces correctly, it struggled more with misclassifying fake faces. In contrast, the Deep CNN model (Figure 2b) showed the weakest performance, with the highest false positive (143) and false negative (686) rates, and comparatively lower true positives (326) and true negatives (845), highlighting its limited effectiveness in distinguishing real from AI-generated images. Overall, these confusion matrices confirm the superior classification ability of transfer learning-based models, particularly MobileNetV2 and ResNet50, over custom-designed CNN architectures.



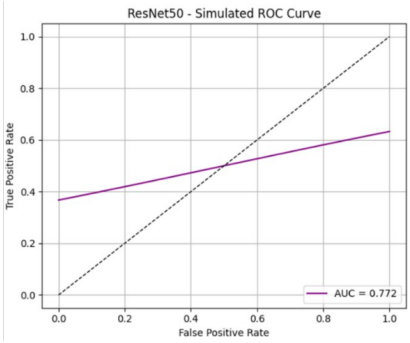
(a) Simple CNN ROC Curve



(b) Deep CNN ROC Curve

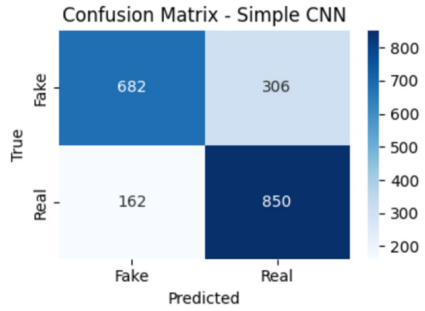


(c) MobileNetV2 ROC Curve

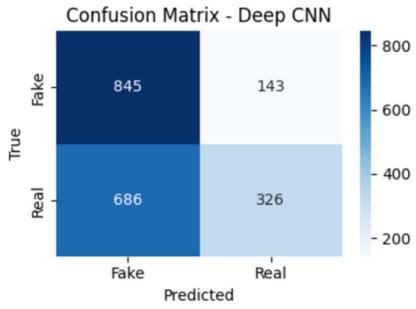


(d) ResNet50 ROC Curve

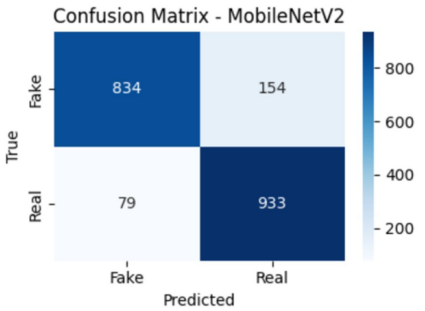
Figure 1: Individual ROC curves for each model showing the trade-off between true positive rate and false positive rate.



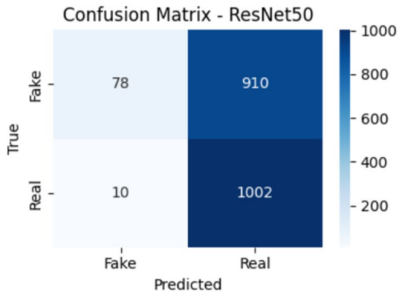
(a) Simple CNN



(b) Deep CNN



(c) MobileNetV2



(d) ResNet50

Figure 2: Confusion matrices for each model showing the distribution of true positives, false positives, true negatives, and false negatives.

6.3 F1 Score Comparison

Figure 3 compares the macro F1 scores of all models, highlighting MobileNetV2 as the best performer with the most balanced precision and recall. Deep CNN achieved a moderate F1 score, while Simple CNN, despite its decent accuracy and AUC, recorded the lowest F1 score, indicating imbalanced classification. ResNet50 also showed a lower F1 score, likely due to reduced recall on certain classes. Overall, the analysis confirms MobileNetV2 as the most consistent and reliable model among those evaluated.

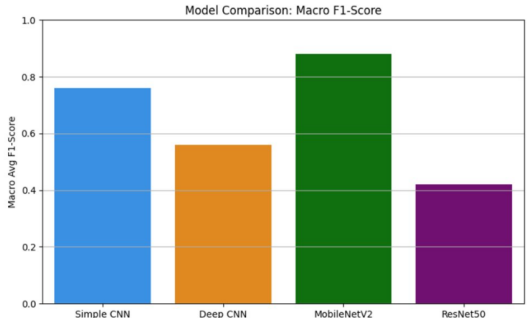


Figure 3: Comparison of F1 scores across all evaluated models. MobileNetV2 shows superior balance between precision and recall, followed by Deep CNN, while Simple CNN shows the weakest performance.

6.4 Validation Accuracy Comparison

Figure 4 presents the validation accuracy comparison of all evaluated models. MobileNetV2 demonstrated the highest validation accuracy, underscoring its efficiency and effectiveness in generalizing to unseen data despite having significantly fewer trainable parameters compared to the larger models. The Simple CNN model, although simpler, achieved competitive validation accuracy, indicating its potential as a lightweight model for baseline evaluations. ResNet50 also performed reasonably well but was outperformed by MobileNetV2 and Simple CNN. In contrast, the Deep CNN model recorded the lowest validation accuracy, further highlighting its limited ability to generalize and its struggle to effectively learn from the dataset. These observations reaffirm the advantages of using transfer learning-based models like MobileNetV2 for robust and resource-efficient AI-generated face detection.

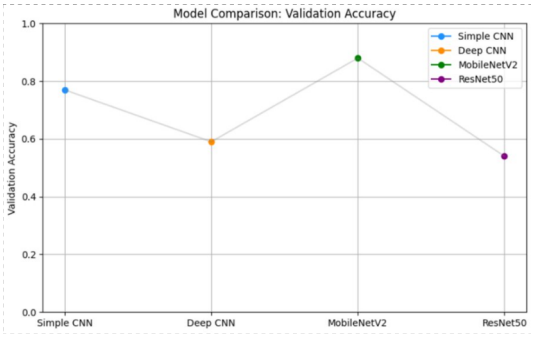


Figure 4: Comparison of validation accuracy across all evaluated models. MobileNetV2 achieved the highest validation accuracy, followed by Simple CNN and ResNet50, while Deep CNN showed the lowest accuracy.

7 Summary Table

Table 5 summarizes the evaluated models, highlighting their performance, complexity, and deployment suitability. MobileNetV2 achieved the highest validation accuracy (95.4%), F1 score, and

Model	Trainable Parameters	Validation Accuracy	F1 Score	AUC-ROC	Best Use Case
Simple CNN	∼11.17 million	∼84.4%	High	Moderate	Educational/demo purposes
Deep CNN	∼19.16 million	∼61.9%	Moderate	High	Server-side applications
MobileNetV2	∼0.16 million	∼95.4%	Very High	Excellent	Mobile, embedded and real-time usage
ResNet50	∼0.26 million	∼77.2%	Moderate	Excellent	High-accuracy desktop environments

Table 5: Model Performance Summary

AUC-ROC, with the lowest parameter count, making it ideal for mobile and real-time applications. ResNet50 also showed strong AUC-ROC and validation accuracy (77.2%) but with moderate F1 score, suitable for high-accuracy desktop usage. Simple CNN, despite its simplicity, delivered good validation accuracy (84.4%) and F1 score, making it useful for educational or demonstration purposes. The Deep CNN model performed the weakest overall, with the lowest validation accuracy (61.9%) and moderate F1 score, though its high AUC-ROC suggests some utility in server-side scenarios where computational resources are ample

8 Conclusion

Based on comprehensive evaluation, MobileNetV2 emerged as the most efficient and accurate model. It delivered top-tier performance with minimal computational overhead, making it ideal for deployment in mobile or real-time systems. ResNet50 also showed competitive results and is a suitable candidate where computational resources are not constrained.

Simple CNN and Deep CNN, while effective to an extent, lacked the performance of transfer learning models and are better suited for foundational learning and model behavior interpretation.

References

[1] Kaggle. 140k real and fake faces. Kaggle Dataset, 2023. https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces/data.

[2] Kaggle. 1m ai-generated faces 128x128. Kaggle Dataset, 2023. https://www.kaggle.com/datasets/dullaz/1m-ai-generated-faces-128x128.

[3] Kaggle. Flickrfaces dataset - nvidia 128x128. Kaggle Dataset, 2023. https://www.kaggle.com/datasets/dullaz/flickrfaces-dataset-nvidia-128x128.