Human-Intelligent Systems Integration
https://doi.org/10.1007/s42454-026-00089-z

ORIGINAL PAPER

Robust deepfake detection through the AI-Guard mobile app for Real-Time image identification

Sami Alanazi[1] · **Seemal Asif**[1] · **Chaitanya Jain**[1]

Received: 14 August 2025 / Accepted: 10 January 2026

Abstract
The proliferation of deepfake technologies has created an urgent need for robust, real-time detection systems capable of verifying image authenticity, particularly in mobile environments. This paper presents a scalable deepfake image detection framework integrated into the AI-Guard mobile application. Our approach leverages fine-tuned, computationally efficient convolutional neural network (CNN) architectures, including VGG19, InceptionV3, Xception, EfficientNetB0, ResNet50, and MobileNetV3Large. Trained on a large-scale, balanced dataset of over 450,000 real and fake images sourced from six publicly available datasets, the models incorporate advanced preprocessing, adversarial data augmentation, and optimized training pipelines. Among these, VGG19 achieved the highest generalization performance with 98.9% validation accuracy and 93.2% accuracy on previously unseen real-world data. The system supports real-time inference via a REST API, enabling practical mobile deployment with low latency. To support transparency and reproducibility, the curated training dataset has been made publicly available through our institutional repository. Our results demonstrate that AI-Guard offers an effective, deployable solution for forensic image verification, contributing to countering misinformation and enhancing trust in digital media.

Keywords Synthetic media forensics · Adversarial image detection · Transfer learning · Computational trust verification · Real-Time mobile inference · Multimedia integrity · Lightweight CNN models

1 Introduction

The rise of deepfake technology has transformed digital media by enabling the creation of highly realistic manipulated images and videos. While these technologies offer valuable opportunities for creativity and education, they also pose significant risks to the authenticity and credibility of multimedia content. Deepfakes can be generated with minimal technical expertise, producing images nearly indistinguishable from genuine ones, which raises critical concerns about privacy, security, and misinformation. They have also been misused to create explicit content for blackmail and contribute to growing public distrust in media and news sources (Alanazi et al. 2025; Uddin Mahmud and Sharmin 2020). As such, developing robust and accessible methods to detect deepfakes has become an urgent priority.

Deepfake generation typically relies on Generative Adversarial Networks (GANs) to produce realistic fake content by leveraging encoder–decoder structures (Nguyen et al. 2019). These models can manipulate facial expressions, swap identities, and merge elements convincingly, leading to challenging forensic scenarios (Schilling et al. 2023). The widespread availability of such tools has underscored the need for improved detection technologies and regulatory frameworks to mitigate misuse (Alanazi and Asif 2023).

Recent advances in deepfake detection leverage Convolutional Neural Networks (CNNs) for their ability to capture complex hierarchical visual patterns. Popular architectures such as VGG19, ResNet152, and ConvNeXt have demonstrated strong performance in identifying subtle manipulation artifacts, even in compressed social media content (Ashok and Joy 2023; Killi et al. 2023). These models are

✉ Sami Alanazi
sami.alanazi@cranfield.ac.uk

Seemal Asif
s.asif@cranfield.ac.uk

Chaitanya Jain
chaitanya.jain.633@cranfield.ac.uk

[1] Centre of Robotics and Assembly, Cranfield University, Cranfield, United Kingdom

Published online: 16 February 2026

well-suited for mobile deployment when optimized for computational efficiency, for example through key frame extraction or lightweight design.

Traditional detection approaches relying on handcrafted features have become inadequate as generative models have improved at mimicking fine-grained human facial behavior, such as synchrony of lip movements and expressions (Rossler et al. 2019; Tolosana et al. 2022). As a result, research has shifted toward large-scale, data-driven methods, with benchmark datasets such as FaceForensics++underscoring the need for diverse training data to achieve robust, cross-dataset generalization (Rossler et al. 2019). However, despite these advancements, many detection systems remain difficult to deploy in real-time scenarios, especially on mobile devices with limited computational resources (Gong et al. 2021).

To address these challenges, this paper introduces AI-Guard, a scalable, real-time deepfake image detection system designed for mobile deployment. Our approach integrates fine-tuning across advanced CNN architectures—including VGG19, InceptionV3, and Xception—while emphasizing lightweight design principles and efficient preprocessing. The VGG19 model in particular benefits from tailored data augmentation strategies and an optimized TensorFlow Data Pipeline to enhance robustness against compression artifacts common in social media images.

Our main contributions are as follows:

- Evaluation and fine-tuning of six state-of-the-art CNN architectures for mobile-friendly deepfake detection.
- Design and adaptation of model architectures with task-specific modifications to improve accuracy and reduce overfitting.
- Assembly and training on a large-scale, balanced dataset of over 450,000 real and fake images to ensure strong generalization.
- Deployment of the best-performing model (VGG19) via a lightweight REST API integrated into a mobile application, enabling real-time, high-accuracy detection under compressed conditions.

1.1 Related works

The detection of deepfake images has advanced significantly through the development of CNN-based methods. Early work such as MesoNet (Afchar et al. 2018) marked a shift from traditional handcrafted features to deep learning, demonstrating that CNNs can effectively capture subtle artifacts introduced during generation. This approach laid the foundation for more sophisticated deepfake detection models.

Ashok and Joy (2023) employed the XceptionNet architecture, highlighting its ability to capture intricate visual patterns and anomalies indicative of manipulation. Trained on extensive datasets, XceptionNet demonstrated strong generalization, enabling reliable classification of unseen instances.

Pan et al. (2020) explored detection using Xception and MobileNet architectures with the FaceForensics++dataset, achieving high accuracy (91–98%) across different manipulation methods. They also proposed a voting mechanism to aggregate model outputs, enhancing robustness through ensemble learning.

Raza et al. (2022) introduced the Deepfake Predictor (DFP), a hybrid of VGG16 and CNN layers, achieving 95% precision and 94% accuracy. Their work underscored the benefits of hybrid models and advanced transfer learning techniques for improving detection performance.

Gong et al. (2021) developed DeepfakeNet, a 20-layer architecture combining ResNet and Inception concepts, demonstrating significant improvements in cross-dataset performance using FaceForensics++, Kaggle, and TIMIT data. DeepfakeNet outperformed mainstream models such as VGG19, ResNet101, and XceptionNet in both accuracy and error rate, although its complexity could hinder deployment in lightweight systems.

Suganthi et al. (2022) proposed a hybrid approach combining Fisherface with Local Binary Pattern Histogram (LBPH) and Deep Belief Networks (DBN) with Restricted Boltzmann Machines (RBM), achieving high accuracy on public datasets like FFHQ, 100 K-Faces DFFD, and CASIA-WebFace. Their method highlights the effectiveness of hybrid deep learning techniques for improving detection robustness.

Our work builds on these foundations by integrating multiple CNN architectures—VGG19, InceptionV3, and Xception—into a unified system optimized for real-time detection on mobile devices. By specifically addressing the challenges of compressed social media content and computational constraints, our approach advances practical and accessible deepfake detection technology.

2 Proposed framework for deepfake detection using AI-Guard mobile app

2.1 Development environment and tools

The proposed deepfake image detection system was implemented in Python using the Visual Studio Code (VSC) Integrated Development Environment. Python was selected for its extensive ecosystem supporting machine learning

libraries such as TensorFlow, PyTorch, and Keras, enabling efficient development and fine-tuning of deep learning models. VSC provided a flexible interface for writing, debugging, and integrating essential tools for the research.

2.2 Leveraging NVIDIA A100 GPU on delta 2 for enhanced model training performance

Model training was accelerated using the NVIDIA A100 GPU available on the Delta 2 high-performance computing (HPC) cluster. The A100 features 6,912 CUDA cores and third-generation Tensor Cores, supporting mixed-precision (FP16 and INT8) training for faster computations without compromising accuracy (NVIDIA, n.d.; NVIDIA 2020). Its 80 GB of HBM2e memory enabled efficient handling of large datasets and complex models, minimizing memory swap bottlenecks.

Additionally, the GPU's Multi-Instance GPU (MIG) capability allows division into isolated instances for resource allocation across parallel jobs. For this research, a full A100 instance was used to maximize computational throughput, significantly reducing training times for the large-scale dataset.

2.3 Dataset

To assemble a comprehensive dataset for deepfake detection, a total of 451,440 images were collected from six primary sources, each offering unique attributes that enhance the dataset's diversity and real-world relevance for training and evaluation purposes. The datasets include OpenForensics (Le et al. 2021), CDDB (Li et al. 2022), WildDeepfake (Zi et al. 2020), 140 k Real and Fake Faces (Xhlulu 2019), Pretty Face (Mu 2020), and Deepfake and Real Images (Karki 2021). Collectively, they ensure exposure to a broad range of generative mechanisms, facial contexts, and manipulation styles.

OpenForensics dataset (Le et al. 2021) contains over 115,000 images with more than 334,000 faces, encompassing variations in pose, lighting, and expression. Around 50k images were selected to expose the model to GAN-based facial forgeries seamlessly blended into realistic contexts, enhancing recognition of fine-grained manipulations. CDDB dataset (Li et al. 2022) provides 836,000 images categorized as GAN-generated, non-GAN, and unknown-model samples. The GAN subset includes output from ProGAN, StyleGAN, BigGAN, CycleGAN, GauGAN, and StarGAN, while non-GAN samples originate from alternative synthesis pipelines such as Glow, CRN, and IMLE. A curated subset of 100,000 images was selected to ensure diversity in generative techniques and support model generalization. WildDeepfake dataset (Zi et al. 2020) contributes 1.18 M faces extracted from 7,314 video sequences collected online, offering authentic "in-the-wild" variability in lighting, backgrounds, and compression artifacts. A 100 k-image subset was used to evaluate robustness under uncontrolled conditions. From the 140k Real and Fake Faces dataset (Xhlulu 2019), an equal mix of 70,000 real FFHQ faces and 70,000 StyleGAN-generated fakes was incorporated. This pairing facilitates learning of subtle textural and geometric cues differentiating authentic from synthetic images. The Pretty Face dataset (Mu 2020) consists of 33,000 StyleGAN2-generated faces of Chinese celebrities, supplemented with corresponding sketches and segmentation maps. A 30,720-image subset was included to introduce stylistic and demographic variation. Finally, the Deepfake and Real Images dataset (Karki 2021) offers a mixed collection of real and manipulated facial images across multiple resolutions and formats. All samples were resized to 224×224 pixels for consistency, with 30,720 images selected to reinforce class balance and modality diversity. In total, the curated corpus comprised 225,720 real and 225,720 fake images, evenly distributed between training and testing subsets. The training set (361,152 images) supported robust learning, while the test set (90,288 images) enabled unbiased performance evaluation. This balanced, multi-source dataset provides a strong foundation for training deep learning models capable of detecting diverse deepfake generation methods. The datasets incorporated for model training and evaluation are outlined in Table 1.

2.4 Data preprocessing

A rigorous preprocessing pipeline was implemented to standardize input images, enhance model robustness, and optimize computational efficiency across various CNN architectures. The preprocessing steps included image resizing, normalization, data augmentation, and the construction of an optimized TensorFlow data pipeline.

2.4.1 Image-resizing

To ensure compatibility with diverse convolutional neural network architectures, all input images were resized to standardized dimensions. Specifically, images were resized to **224 × 224 pixels** for models such as **VGG19**, **ResNet50**, **EfficientNetB0**, and **MobileNetV3Large**, as shown in Fig. 1. For models requiring larger input dimensions, such as **InceptionV3** and **Xception**, images were resized to **299 × 299 pixels**. This resizing strategy not only standardizes the input shape but also improves batch processing efficiency and accelerates convergence by aligning the data format with model-specific architectural requirements.

Table 1 Detailed summary of datasets used for model training and evaluation

Dataset	Generation Method(s)	Diversity (Subjects/ Environments)	Resolution	Key Features & Intended Use
The OpenForensics Dataset	GAN-based techniques	Moderate diversity, controlled settings	High	Lifelike synthetic faces; forensics-focused, high authenticity
CDDB Dataset	GAN (ProGAN, StyleGAN, BigGAN, etc.), non-GAN (Glow, CRN, etc.)	High diversity; various online sources	Mixed	Broad representation of GAN and non-GAN deepfakes; general-purpose detection
Wild-Deepfake Dataset	GANs and deepfake software	High diversity; real-world environments	High	High-quality, real-world wild deepfakes for robustness testing
140k Real and Fake Faces (Kaggle)	GAN (StyleGAN)	Limited to FFHQ faces; controlled environment	High	Split of real and synthetic; ideal for evaluating basic deepfake detection
Pretty Face Dataset	GAN (StyleGAN2)	Limited diversity; primarily celebrity faces	High	Includes sketches and segmentation for advanced image translation
Deepfake and Real Images	Various deepfake generation techniques	Moderate diversity; focused on manipulated faces	Mixed	Diverse formats; useful for validating detection models across resolutions

2.4.2 Normalization

To ensure consistency and enhance model convergence, all pixel intensity values were normalized to a common scale between 0 and 1. This transformation mitigates the influence of varying lighting conditions and color intensities across images, which is particularly critical when training convolutional neural networks (CNNs) on large and diverse datasets. The normalization was performed using min-max scaling, defined mathematically as:

$$x_{normlized} = \frac{x}{255} \tag{1}$$

where x denotes the original pixel value in the range [0,255], and x_normalized represents the rescaled value in the normalized range [0,1]. This operation standardizes input data, stabilizes the training process, and facilitates faster and more reliable convergence during optimization. Moreover, normalization ensures compatibility with pre-trained CNN models, such as VGG19 and EfficientNetB0, which assume inputs in the normalized range due to their ImageNet-based training configurations. An illustrative example of this process is shown in Fig. 2.

2.4.3 Data augmentation

To further enhance the training dataset, augmentation techniques were applied to introduce variability and simulate real-world conditions. These augmentation methods, designed to improve the model's robustness and generalization, include:

- **Random Horizontal Flipping**: Mirrors images horizontally using tf.image.random_flip_left_right, introducing spatial variability that reflects natural conditions.
- **Brightness Adjustment**: Randomly alters brightness using tf.image.random_brightness, simulating a range of lighting conditions encountered in real-world scenarios.

These augmentations were applied dynamically during training to ensure that each epoch encounters varied representations of the original data, thereby reducing overfitting and improving generalization performance.

2.4.4 TensorFlow data pipeline

The dataset was further optimized by employing TensorFlow's tf.data.Dataset API to streamline data loading and preprocessing. Images and corresponding labels were converted into TensorFlow datasets with the following enhancements:

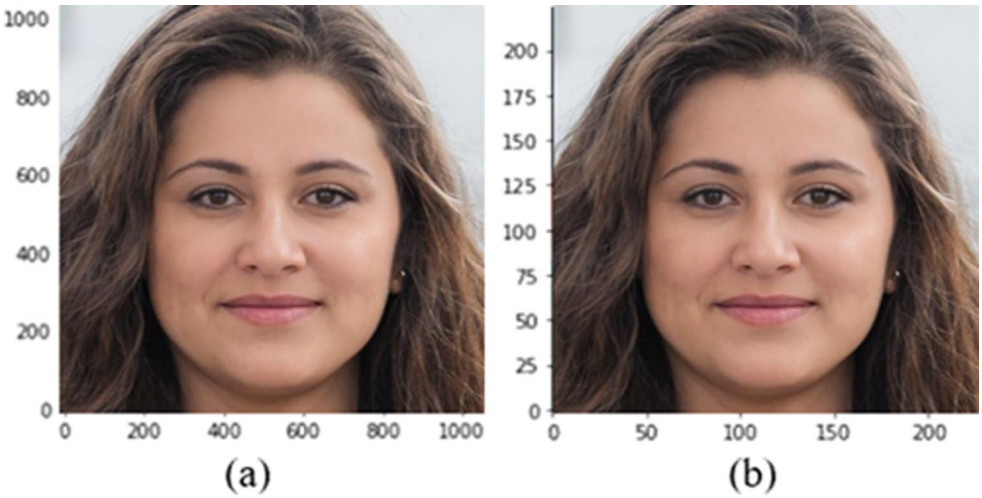

Fig. 1 Image resizing

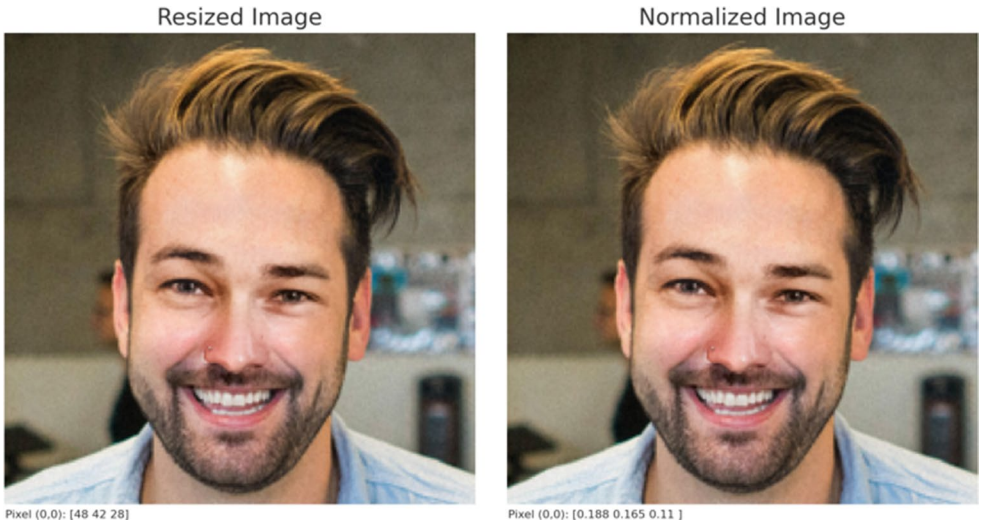

Fig. 2 Pixel value normalization example. The left image shows raw RGB values in the standard 0–255 range, while the right image shows pixel values normalized to the 0–1 range using the transformation. This process standardizes input data and stabilizes model

Table 2 VGG19 architecture

Layer Type	Output Shape	Frozen Status	Comments
Input Layer	(224, 224, 3)	-	Image input
Block 1 (Conv2D, Conv2D)	(224, 224, 64)	Frozen	Basic feature extraction
Block 2 (Conv2D, Conv2D)	(112, 112, 128)	Frozen	Maintains base features
Block 3 (Conv2D x4)	(56, 56, 256)	Frozen	Prevents overfitting
Block 4 (Conv2D x4)	(28, 28, 512)	Frozen	Extracts high-level features
Block 5 (Conv2D x4)	(14, 14, 512)	Fine-tuned	Detects deepfake-specific nuances
Flatten Layer	(None, 25088)	Trainable	Converts feature maps to vectors
Dense (512 units)	(None, 512)	Trainable	L2 Regularization, Dropout 0.6
Dense (256 units)	(None, 256)	Trainable	Dropout 0.6
Output Layer (Sigmoid)	(None, 1)	Trainable	Final classification output

- **Batching**: Images were grouped into batches of a fixed size to optimize training efficiency and reduce memory overhead.
- **Shuffling**: The dataset was shuffled to ensure randomization and minimize model overfitting.
- **Prefetching**: Data was preloaded into memory using buffer_size=tf.data.AUTOTUNE, reducing latency during training and improving computational performance.

This end-to-end preprocessing architecture ensured that the model was consistently trained on well-structured, diverse, and standardized input data, laying the foundation for robust and generalizable deepfake detection performance.

2.4.5 Evaluation metrics and visualization

To assess the performance of the deepfake image detection models, a combination of quantitative metrics and visual diagnostic tools were employed during both the training/

validation phase and the final testing stage. These measures ensured a rigorous evaluation of classification effectiveness, model generalizability, and robustness in real-world conditions.

Training and Validation Metrics.

During model training and validation, performance was monitored using two core metrics: **accuracy** and **loss**. Accuracy represents the proportion of correctly classified samples relative to the total, providing a high-level view of the model's learning progress. The loss function, conversely, quantifies the error between predicted and actual labels, serving as a direct optimization target during backpropagation. Both metrics were tracked and visualized across epochs to detect signs of overfitting, underfitting, or stagnation in learning.

In addition, **confusion matrices** were generated to provide a granular view of model predictions across classes. This matrix highlights the distribution of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN), enabling the calculation of derived performance indicators:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$F1 - Score = \frac{2 \times \ Precision \times \ Recall}{Precision + Recall} \tag{5}$$

These metrics collectively offer a comprehensive performance profile, particularly under class imbalance scenarios often observed in real-world datasets.

Testing on Unseen Data.

To evaluate model generalization, the best-performing model (VGG19) was tested on a held-out, previously unseen dataset. The **Receiver Operating Characteristic (ROC) curve** was employed for this purpose. This curve plots the true positive rate (Recall) against the false positive rate (FPR) across varying classification thresholds, offering insight into the model's discriminative ability.

The **Area Under the Curve (AUC)** was also calculated to summarize ROC performance into a single scalar. An AUC closer to 1.0 indicates a strong ability to distinguish between real and fake images, while values near 0.5 suggest performance equivalent to random guessing.

Together, these evaluation techniques ensure a robust, multi-faceted assessment of the model's effectiveness across both controlled and real-world conditions..

2.5 Model architecture and training process

2.5.1 VGG19

VGG19 was selected due to its deep architecture, which consists of 19 layers (16 convolutional layers and 3 fully connected layers). It was chosen for its ability to capture both low- and high-level features. In this implementation:

- The first 4 blocks (13 layers) were frozen to preserve low-level feature extraction capabilities and reduce overfitting.
- The last 2 blocks were fine-tuned, enabling the detection of subtle manipulations specific to deepfake images.
- Custom top layers were added, incorporating L2 regularization and dropout to enhance the model's generalization ability. The detailed configuration is presented in Table 2.

The training process began with comprehensive preprocessing, including resizing, normalization, and augmentation on 451,440 images. The dataset was transformed using the TensorFlow Data Pipeline, which handled batching, shuffling, and prefetching for efficient GPU usage (Fig. 3).

During training, the frozen layers preserved foundational visual features, while fine-tuned layers adapted to deepfake-specific manipulations. Custom dense layers, incorporating L2 regularization and dropout, helped prevent overfitting. The final sigmoid output layer delivered interpretable binary classification scores, enabling reliable distinction between real and fake images.

2.5.2 InceptionV3

InceptionV3 uses advanced inception modules to capture multi-scale image features via parallel convolutional filters of different sizes (e.g., 1×1, 3×3, 5×5) combined with pooling layers. This architecture is well suited for identifying diverse patterns, such as subtle facial manipulations under varying lighting, expressions, and perspectives.

For our task:

- The base InceptionV3 layers were frozen to retain general feature extraction learned from ImageNet.
- Custom dense layers with dropout were fine-tuned to detect deepfake-specific features.
- Batch normalization and dropout layers were applied throughout to stabilize training and reduce overfitting.

These techniques ensured robust, generalizable detection of both obvious and subtle manipulations, maintaining

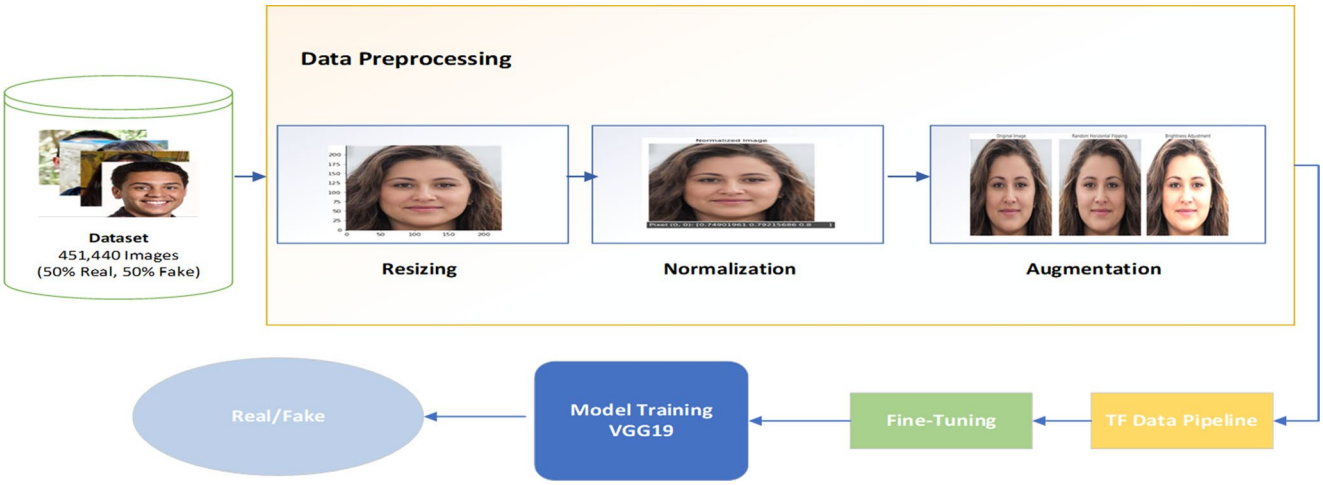

Fig. 3 VGG19-based deepfake detection model architecture and training workflow

high performance across varied and challenging image conditions.

2.5.3 Xception

Xception, an advanced extension of the Inception architecture, leverages depthwise separable convolutions to independently process spatial and channel-wise information. This innovation reduces computational complexity while maintaining model depth, making it highly efficient for extracting fine-grained visual features. In deepfake detection, this design is especially advantageous because it enables efficient learning of complex manipulation artifacts without excessive computational overhead, enhancing both speed and accuracy.

In Xception, the depthwise separable convolutions break down convolutions into spatial and channel-wise operations, which effectively capture high-dimensional features essential for deepfake detection. This approach allows Xception to discern intricate details in images, such as artifacts introduced by generative models, without compromising performance. The Xception architecture also provides flexible, efficient feature extraction, allowing detection of subtle inconsistencies often present in deepfake images.

To counteract overfitting during training, Xception employed regularization techniques such as L2 regularization and dropout layers. L2 regularization was applied to penalize large weights, promoting smoother model learning and reducing the risk of overfitting. Dropout layers, implemented in the custom dense layers, further helped by randomly disabling neurons during each training iteration, forcing the model to learn more robust features. These regularization techniques provided stability, especially when training on diverse datasets that encompass a variety of manipulation techniques.

One of Xception's key strengths is its consistency across datasets. During training and validation, the model consistently achieved high accuracy, showcasing its ability to generalize to unseen deepfake images. This reliability underscores its potential in real-world applications where models must adapt to various types of manipulated media.

2.5.4 EfficientNetB0

EfficientNetB0 was selected for its balance between accuracy and computational efficiency. The architecture leverages a novel compound scaling method introduced by Tan and Le (2019), which uniformly scales the network's depth, width, and input resolution using a set of predefined coefficients. This enables the model to achieve high accuracy with fewer parameters compared to traditional convolutional neural networks (CNNs). The structure of EfficientNetB0 is built upon MBConv blocks, which are optimized bottleneck layers originally designed for MobileNetV2. These blocks are computationally efficient and contribute to reduced memory and parameter usage without sacrificing performance. The diagram below, adapted from Amin et al. (2022), illustrates the EfficientNetB0 architecture, showcasing how it scales and arranges its MBConv blocks to achieve an optimal trade-off between resource usage and classification performance (Fig. 4).

In this implementation:

- The base EfficientNetB0 layers were pre-trained on ImageNet and set to be trainable for fine-tuning.
- A Global Average Pooling layer was added, followed by custom dense layers with 512 and 256 units.
- L2 regularization and a high dropout rate (0.6) were applied to reduce overfitting.

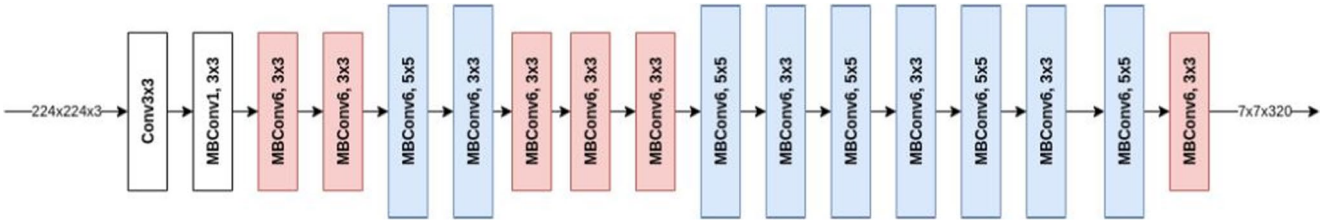

Fig. 4 The EfficientNetB0 architecture as implemented in the study by Amin et al. (2022)

- The output layer uses a sigmoid activation function for binary classification.

The training pipeline included data preprocessing (resizing, normalization), and augmentation (random flips and brightness adjustments). The TF data pipeline was utilized for batching, shuffling, and prefetching.

To improve model generalization, class weights were calculated and early stopping and learning rate reduction callbacks were included. Mixed precision training was also used for memory optimization.

2.5.5 ResNet50

Deep convolutional architectures often suffer from performance degradation as network depth increases. To overcome this limitation, residual networks such as ResNet50 introduce identity shortcut connections that preserve gradient flow and facilitate the training of deeper models. These structural innovations make ResNet50 particularly effective for image classification tasks where subtle spatial features must be preserved, as in the detection of manipulated imagery.

Building on this theoretical foundation, a ResNet50 model pre-trained on ImageNet was used as the backbone. All convolutional layers were unfrozen to enable complete fine-tuning on the target dataset. A custom classification head was appended, consisting of two dense layers with 512 and 256 neurons, each incorporating L2 regularization ($\lambda = 0.001$) and dropout (0.6) to reduce overfitting. The output layer used sigmoid activation to predict binary labels corresponding to real and fake image classes.

The training pipeline was implemented in TensorFlow, incorporating mixed-precision computation to improve efficiency. Preprocessing included resizing images to 224×224 pixels, normalization, and label encoding. To improve generalizability, data augmentation was applied through random horizontal flips and brightness alterations. Class imbalance was addressed through computed class weights.

Model training followed a robust strategy involving early stopping, adaptive learning rate scheduling, and checkpointing after each epoch. The system also supported resumption from the most recent checkpoint to ensure continuity in case of interruptions. Evaluation on the validation set included accuracy, confusion matrix visualization, and a full classification report. In addition, training dynamics were tracked using accuracy and loss curves. The overall configuration demonstrates ResNet50's effectiveness in combining architectural depth with regularization and augmentation techniques to address the challenges of deepfake image classification.

Table 3 MobileNetV3Large architecture

Layer Type	Output Shape	Trainable	Description
Input Layer	(224, 224, 3)	-	Image input
MobileNetV3 Base	Various	Partially frozen	Lightweight and efficient
Flatten Layer	(None, N)	Yes	Converts feature maps
Dense (512 units)	(None, 512)	Yes	L2 Regularization, Dropout 0.6
Dense (256 units)	(None, 256)	Yes	Dropout 0.6
Output Layer (Sigmoid)	(None, 1)	Yes	Final classification

2.5.6 MobileNetV3Large

MobileNetV3Large was incorporated to investigate the performance of lightweight architectures in deepfake detection. Its combination of depthwise separable convolutions, squeeze-and-excitation modules, and hard-swish activation makes it suitable for mobile deployment.

Implementation details:

- The first 80% of MobileNetV3Large layers were frozen, focusing fine-tuning on the last few convolutional blocks.
- Flattening was applied instead of global pooling to retain spatial information before classification.
- Dense layers followed by dropout and L2 regularization were used to prevent overfitting.

As outlined in Table 3: MobileNetV3Large Architecture Summary, the model architecture is divided into distinct functional layers, with clear demarcation of trainable and frozen components. The base model serves as a compact

and efficient feature extractor, while the custom top layers facilitate classification based on deepfake-specific artifacts.

The model was trained using the Adam optimizer with a learning rate of 5e-5, and monitored using callbacks such as early stopping and ReduceLROnPlateau. Training was accelerated using TensorFlow's mixed precision policy, reducing memory consumption and training time while preserving accuracy.

2.6 Enhanced model integration and deployment

Following the selection of VGG19 as the most effective model for deepfake detection, it was integrated into a mobile application, as shown in Fig. 5. This app, named "AI - Fake Guard," enables users to analyze images in real-time, either by manually uploading images or through automated scanning. To facilitate efficient processing of detection requests, a REST API was deployed locally, ensuring that the system remains scalable and responsive even under heavy usage. This locally hosted REST API acts as a bridge between the app's user interface and the backend detection engine.

In this architecture, when users submit an image via the app, an HTTP POST request is sent to the REST API server, which processes the image using the fine-tuned VGG19 model. The server then returns a prediction indicating whether the media is real or fake. This setup allows users to run the app seamlessly on both Android emulators and physical devices, provided they are on the same local network as the REST server. By implementing an enhanced algorithm for data transmission, the system now reduces latency and improves response times, ensuring real-time feedback even under variable network conditions. While the system is currently deployed on a mobile app, it is highly adaptable and can be integrated into other devices such as IoT systems, laptops, or computers, making it a versatile solution for various platforms. Additional features, such as a history log and notification system, further enhance user experience by providing a streamlined interface for managing detected deepfake content.

2.6.1 REST API for Real-Time detection

To enable real-time detection of deepfake content, the detection model was deployed on a REST server, creating a seamless connection between the app's user interface and the backend processing engine. This server-based architecture enables efficient and scalable request handling, ensuring that the app remains responsive and reliable, even under heavy usage.

The REST API is hosted locally, meaning it runs on a designated IP address within a local network. For users running the app on an Android emulator, the app's settings must

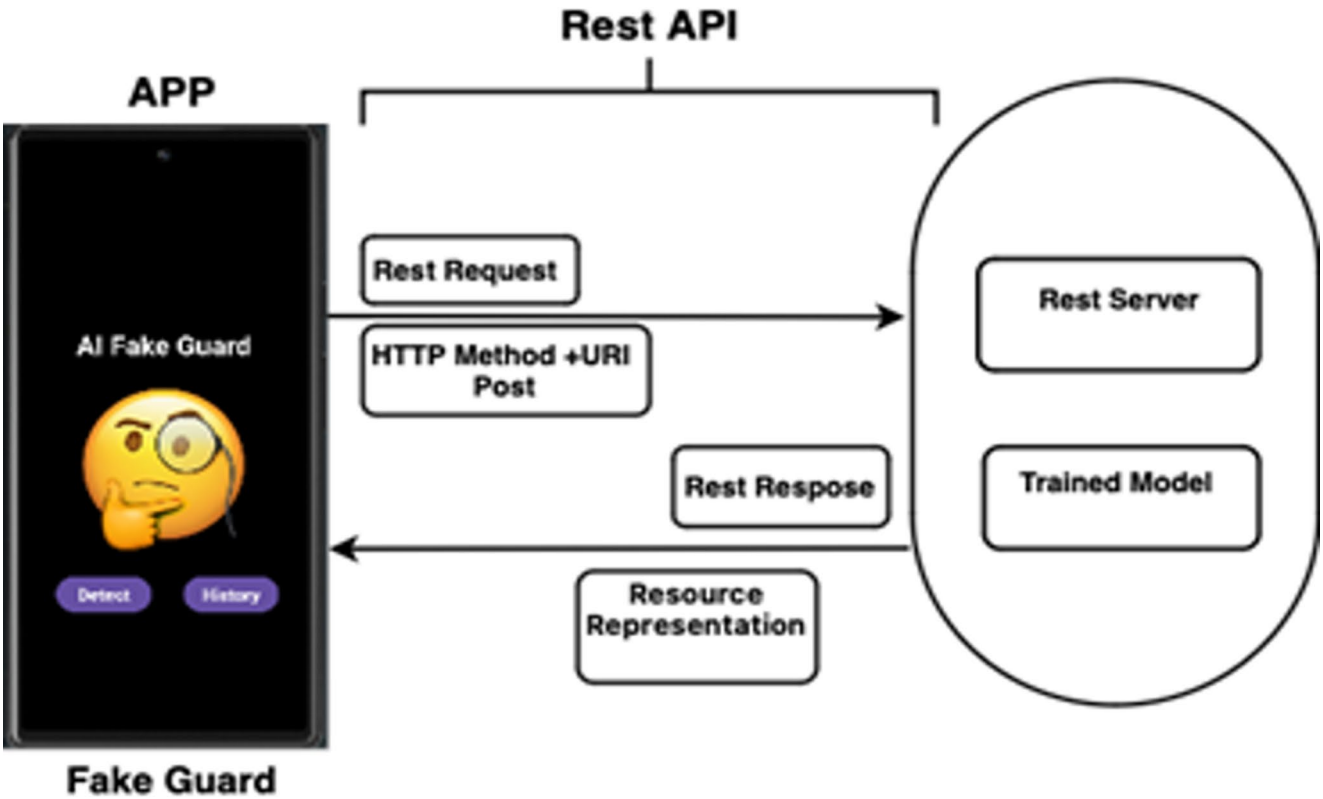

Fig. 5 REST API-integrated architecture of AI - fake guard for real-time deepfake detection

be updated to direct requests to the emulator's IP address (e.g., http://10.0.2.2:5000/predict). For those using the app on a physical device, the app should point to the device's IP address (e.g., http://192.168.1.133:5000/predict or http://172.20.10.4:5000/predict). Both the device and the REST server need to be on the same network for effective communication, as illustrated in Fig. 5.

When a user selects an image for analysis, the app sends an HTTP POST request containing the image file and any necessary metadata to the REST server, as shown in Fig. 5.

Fig. 6 AI- fake guard app interface

The server then uses the detection model to analyze the image in detail, looking for characteristics that indicate manipulation or synthetic generation. To enhance accuracy, the server leverages optimized preprocessing techniques such as noise reduction and contrast adjustment before analysis. This involves complex computations and pattern recognition using deep learning, enabling accurate assessment of the media's authenticity.

The server then returns a classification result in JSON format, which is easily readable and indicates whether the media is real or fake. The app receives this response and promptly displays the result to the user, allowing them to quickly verify the authenticity of their media, as depicted in Fig. 5. Integrating the REST API into the app not only expands its capabilities but also provides a reliable and scalable solution for real-time deepfake detection, empowering users to efficiently identify and respond to potentially manipulated content.

2.6.2 Resource representation and additional features

The deepfake detection app, AI-Fake Guard, is designed with a user-centered approach, presenting detection results in a clear and accessible manner. As shown in Fig. 6, the app offers two primary options: "Detect" and "History". The "Detect" feature allows users to analyze images in real-time, either through manual uploads or automated scanning. When the app identifies a fake image, an alarm notification immediately alerts the user, as illustrated in Fig. 7. This notification system has been enhanced to include customizable alert tones and priority settings, allowing users to tailor their experience based on urgency.

Beyond simply displaying detection results, the app also includes a "History" feature, shown in Fig. 8, which logs all instances of detected fake content. This comprehensive history log provides users with a valuable resource for tracking and reviewing identified deepfakes over time. To further streamline organization, the app stores all flagged images in a dedicated "Fake" folder on the user's device, allowing for easy access to potentially misleading content. The app is designed for flexibility and user control, allowing users to manage their media according to their preferences. Built for compatibility with Android operating systems, the app is accessible across a wide range of devices. The architecture has been optimized for scalability, enabling future updates to support additional detection models or extended features, such as video analysis.

Overall, the app serves not only as a powerful tool for detecting deepfakes but also provides users with resources to manage and respond to synthetic media effectively. Its integration with real-world use cases, such as digital forensics

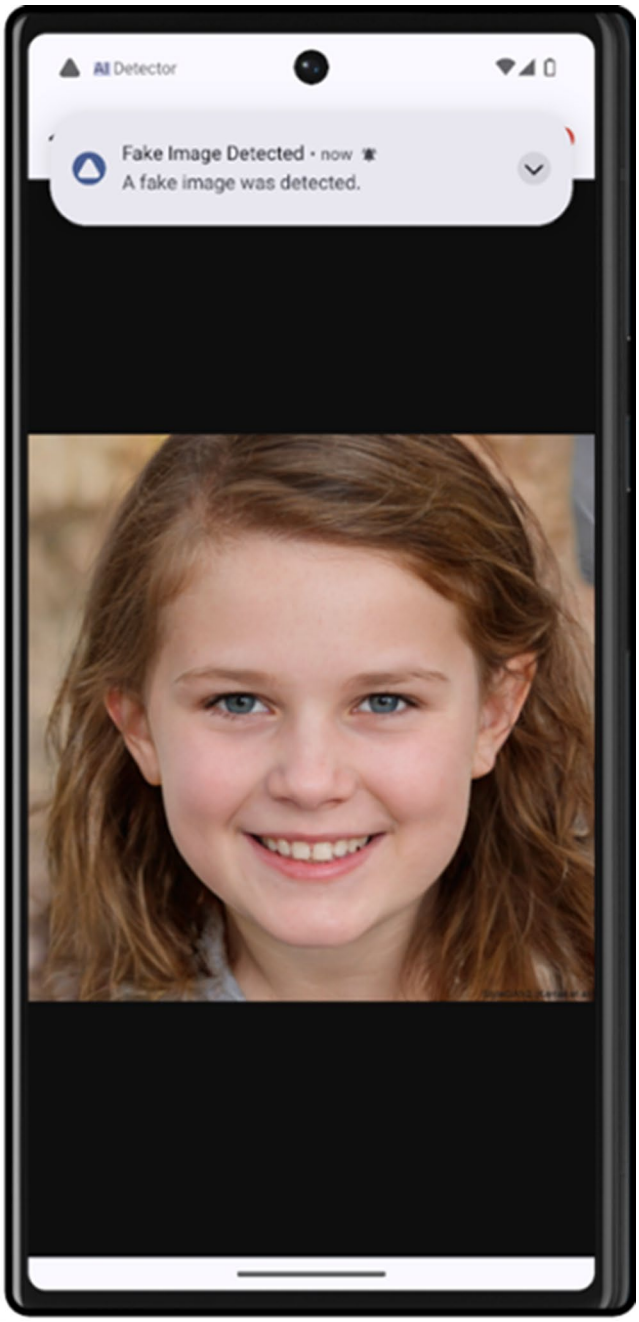

Fig. 7 Real-time alarm notification for detected fake content

and media verification, further positions it as a critical tool in combating digital misinformation and manipulation.

3 Results and discussion

3.1 VGG19 model results

After a comprehensive comparative analysis, VGG19 emerged as the most effective model due to its exceptional

Fig. 8 History log of detected fake images

ability to generalize on both training and unseen data. The model was evaluated using an initial training/testing dataset and performed exceptionally during testing on unseen data from diverse sources. VGG19 consistently demonstrated high accuracy across both scenarios.

3.1.1 Training and validation performance

During training, the VGG19 model showed remarkable progress early on, reaching approximately 98.9% accuracy by the second epoch and gradually stabilizing at 99% by the final epoch. The validation accuracy closely followed this trend, leveling off at an impressive 98.92%. This steady performance highlights the model's ability to generalize effectively while avoiding significant overfitting.

Figure 9 illustrates the close alignment of training and validation accuracy curves, further emphasizing the model's robust learning process. Similarly, the loss curves presented in Fig. 10 reveal consistent optimization, with training loss steadily declining and validation loss maintaining a low,

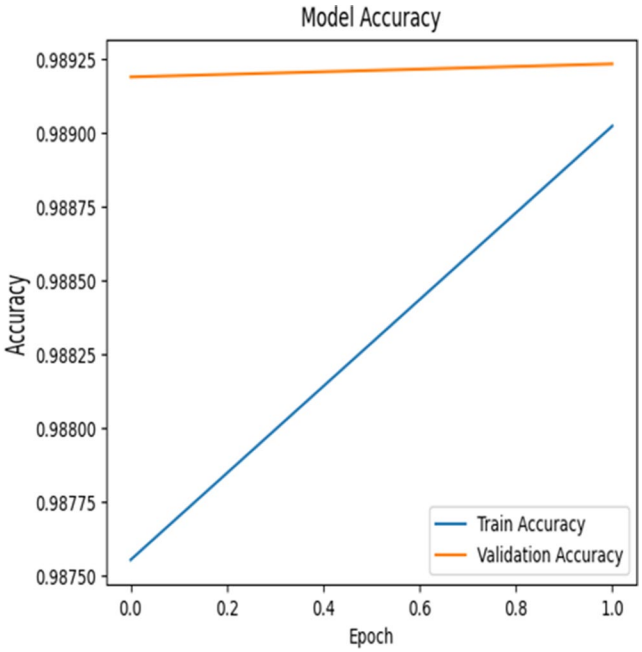

Fig. 9 VGG19 loss

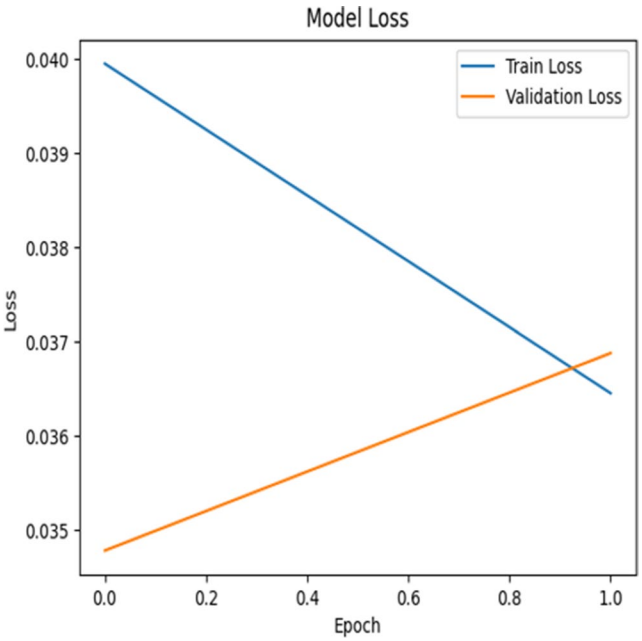

Fig. 10 VGG19 accuracy

stable value. These results demonstrate the model's efficient optimization and its capacity to minimize overfitting.

3.1.2 Initial Training/testing

The VGG19 model exhibited exceptional performance in the initial testing phase, accurately classifying 44,834 fake images as true negatives and 44,478 real images as

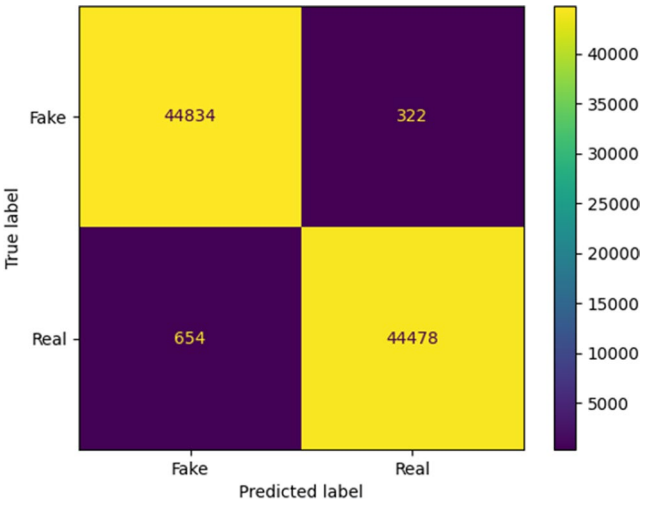

Fig. 11 VGG19 confusion matrix

true positives, as shown in Fig. 11. Misclassification rates were impressively low, with only 322 false positives (fake images misclassified as real) and 654 false negatives (real images misclassified as fake). These findings underscore the model's precise ability to detect subtle artifacts introduced through image manipulation, positioning it as a highly reliable tool for differentiating between authentic and manipulated content.

The accuracy progression (Fig. 9), along with the corresponding training and validation loss curves (Fig. 10), reflects the model's strong generalization capability and minimal risk of overfitting. The training loss consistently decreases and stabilizes around 0.037, while the validation loss plateaus slightly above 0.035, exhibiting no substantial divergence between the two. This convergence behavior indicates that the model effectively learns the underlying data distribution without memorizing the training samples. Notably, the validation accuracy consistently remains marginally higher than the training accuracy throughout the epochs, further affirming the absence of overfitting. The incorporation of dropout layers and early stopping mechanisms contributed to enhanced regularization and training stability. These findings confirm that the proposed VGG19-based architecture not only achieves a strong fit on the training data but also sustains reliable performance on unseen validation instances.

3.2 InceptionV3 results

The performance of InceptionV3 demonstrated notable strength during both initial testing and evaluations on unseen data. However, some variability in training was observed, potentially due to the complexity of the dataset.

3.2.1 Training and validation performance

InceptionV3's training accuracy displayed noticeable fluctuations, as seen in Fig. 12. Although the model achieved high accuracy by the final epoch, intermediate stages exhibited variability, possibly linked to unstable gradients during fine-tuning. Validation accuracy, however, remained consistent, stabilizing above 90%. Figure 13 illustrates the corresponding loss curves, with training loss mirroring the variability in accuracy, while validation loss steadily decreased, confirming its capacity to generalize effectively.

3.2.1.1 Initial training/testing results InceptionV3 performed admirably during initial testing, accurately classifying 43,278 fake images and 44,645 real images (Fig. 14). Misclassifications included 1,941 false positives and 424 false negatives, reflecting strong performance but also some limitations in identifying challenging cases of fake content.

3.3 Xception results

Xception, while yielding reasonable results, lagged behind both VGG19 and InceptionV3 in accuracy and reliability during evaluations.

3.3.1 Training and validation performance

Validation accuracy for Xception ranged between 77% and 79%, consistently lower than that of VGG19 and InceptionV3. Figures 15 and 16 highlight irregularities in the validation loss, indicating challenges in learning subtle patterns from the dataset. This inconsistency may reduce the model's effectiveness in generalizing across diverse scenarios.

3.3.2 Initial training/testing results

During initial testing, Xception correctly classified 28,088 fake images and 44,019 real images, as illustrated in Fig. 17. However, the model exhibited higher rates of misclassification, with 16,911 fake images labeled as real and 1,270 real images classified as fake. These results underscore the need for improvement in the model's ability to detect subtle manipulations effectively.

3.4 EfficientNetB0 results

EfficientNetB0 showed a steady increase in both training and validation accuracy (Fig. 18), with training accuracy peaking around 74% and validation accuracy nearing 70%. The loss curves in Fig. 19 show consistent improvement and convergence with no signs of overfitting.

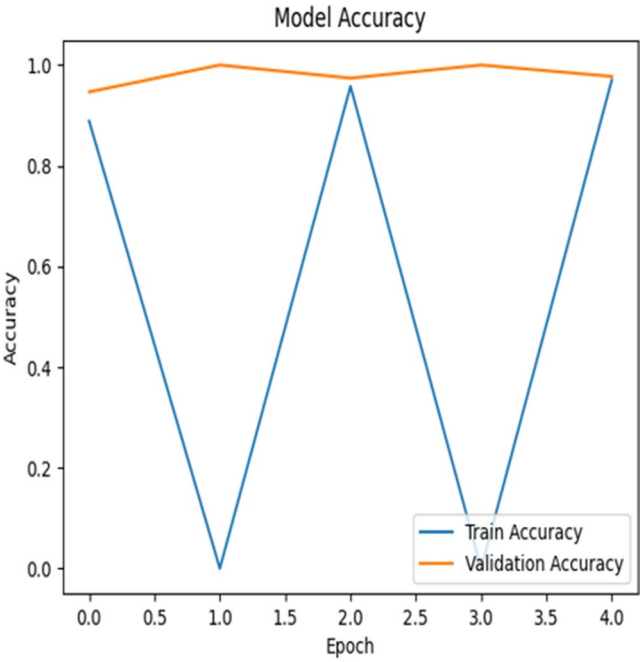

Fig. 12 InceptionV3 Loss

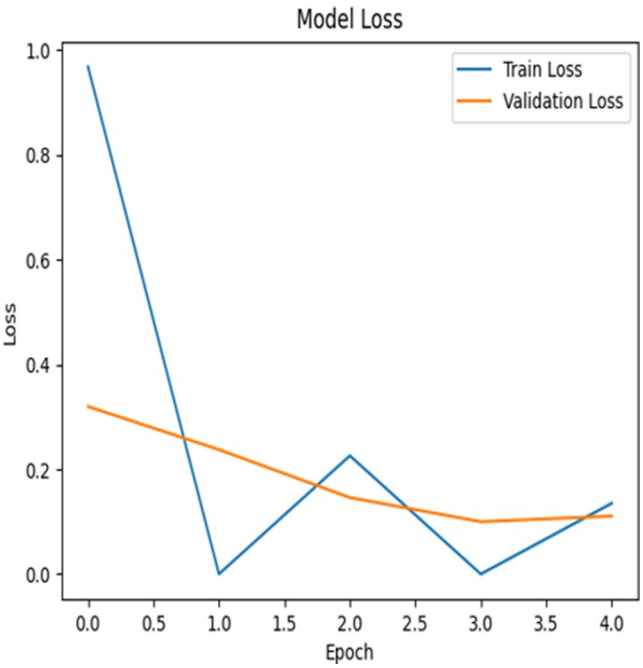

Fig. 13 InceptionV3 accuracy

Initial Testing Results: As shown in Fig. 20 EfficientNetB0 correctly classified 20,000 fake and 43,449 real images. However, it misclassified 25,245 fake images as real and 1,800 real images as fake. This indicates the model struggled to detect subtle manipulation features in certain fake images.

These results suggest that while EfficientNetB0 offers a balanced performance with efficient resource usage, it may

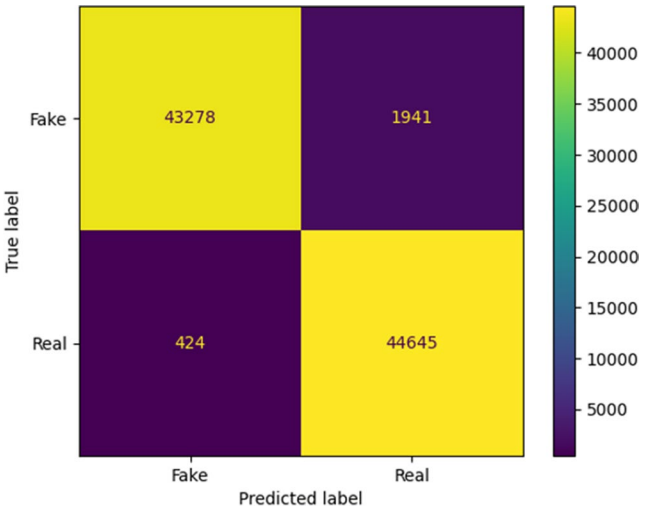

Fig. 14 InceptionV3 confusion matrix

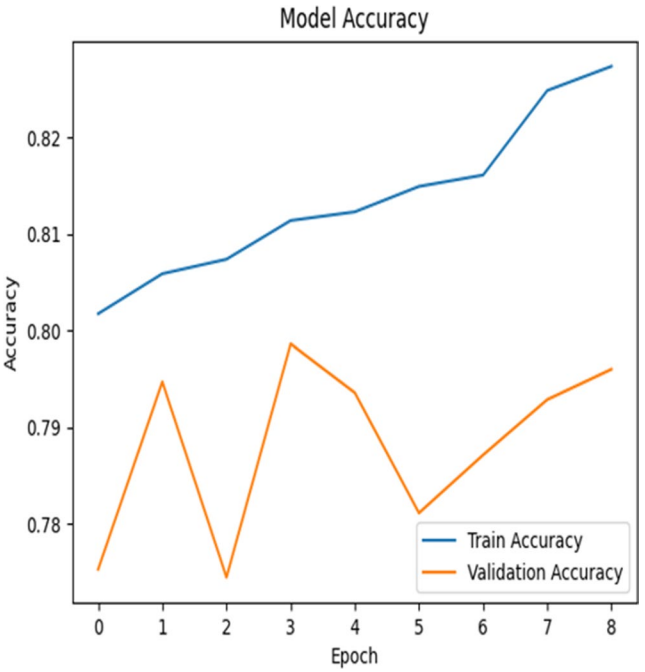

Fig. 15 Xception loss

require further tuning or integration with attention mechanisms to better detect subtle artifacts.

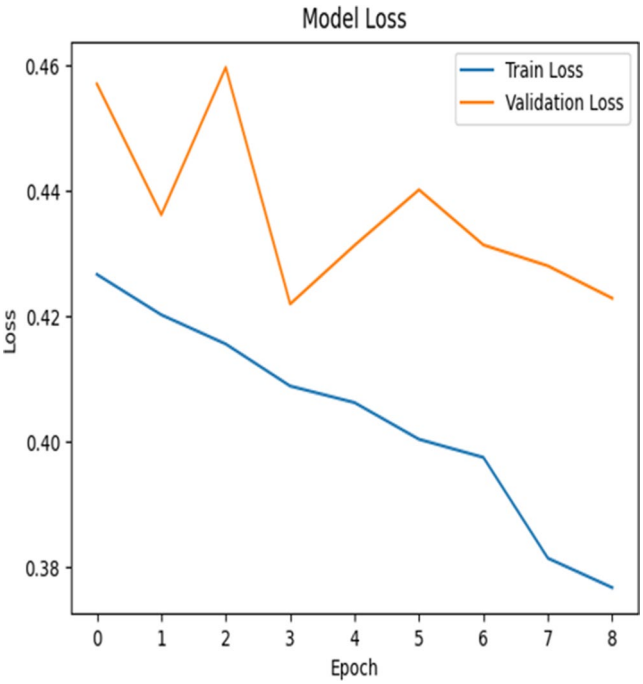

Fig. 16 Xception accuracy

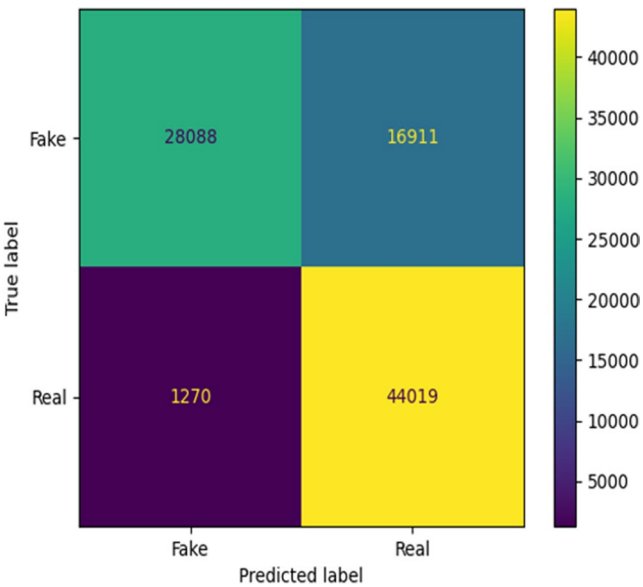

Fig. 17 Xception confusion matrix

3.5 ResNet50 results

ResNet50 exhibited improved generalization capability compared to EfficientNetB0, with validation accuracy stabilizing around 68%, as illustrated in Fig. 21. The training and validation loss curves (Fig. 22) followed a generally declining pattern across epochs, indicating effective convergence. However, further examination highlighted some training instability, particularly in the model's handling of real samples.

As depicted in the confusion matrix (Fig. 23), ResNet50 correctly classified 42,088 fake and 18,987 real images. Nevertheless, it misclassified 26,236 real images as fake and produced 2,977 false positives. These outcomes point to a classification imbalance, with the model demonstrating a bias towards predicting fake content. The inconsistency in

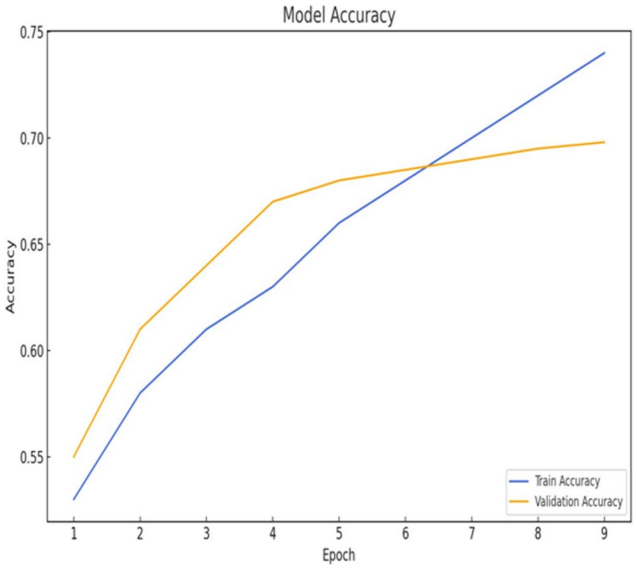

Fig. 18 EfficientNetB0 accuracy

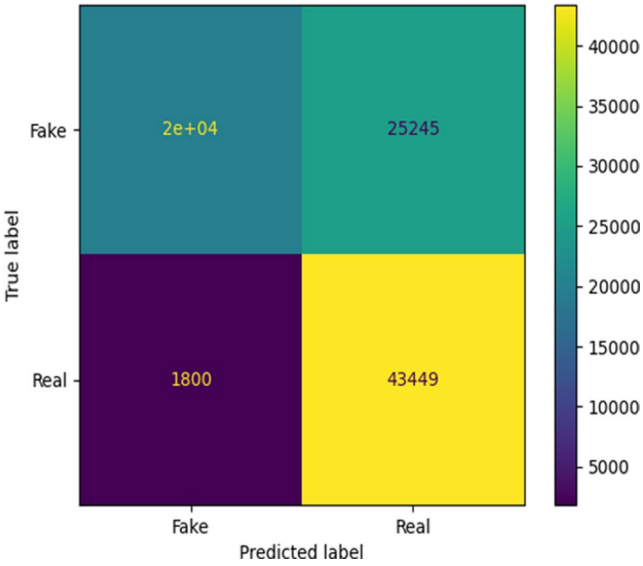

Fig. 20 EfficientNetB0 confusion matrix

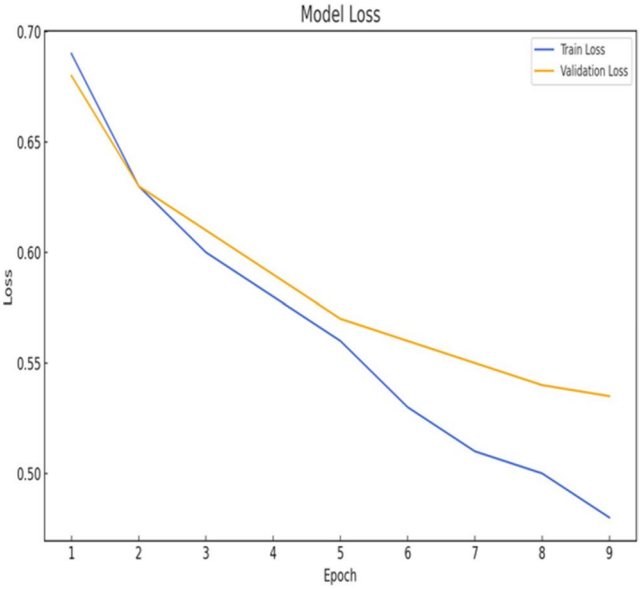

Fig. 19 EfficientNetB0 loss

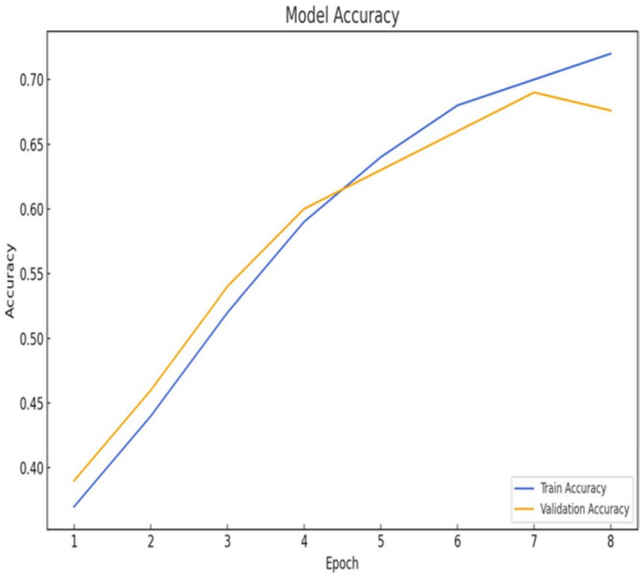

Fig. 21 ResNet50 accuracy

loss convergence during later epochs may indicate under-regularization, even though all layers were fine-tuned, and this likely contributed to the model's limited reliability in balanced real-versus-fake classification tasks.

3.6 MobileNetV3Large results

MobileNetV3Large delivered stable training dynamics and maintained competitive generalization performance despite its lightweight architecture. The model achieved a final training accuracy of approximately 74% and a validation accuracy of around 76%, as illustrated in Fig. 24. These results closely align with those of ResNet50 while offering significantly reduced computational complexity.

The convergence of training and validation loss around 0.6 (Fig. 25) reflects a stable optimization process without significant divergence between learning and evaluation phases.

As shown in the confusion matrix (Fig. 26), the model correctly identified 36,071 fake and 30,807 real images. Nonetheless, it also produced 9,072 false negatives for fake images and 14,338 false positives for real images. This outcome suggests a relatively balanced classification performance, with a slight inclination toward detecting manipulated content.

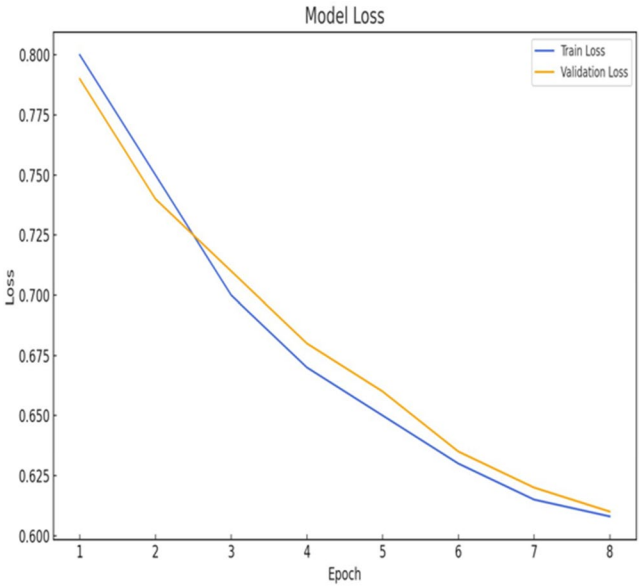

Fig. 22 ResNet50 loss

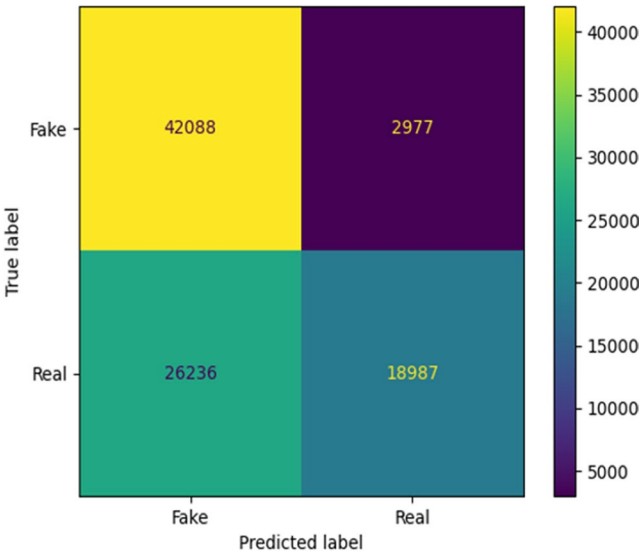

Fig. 23 ResNet50 confusion matrix

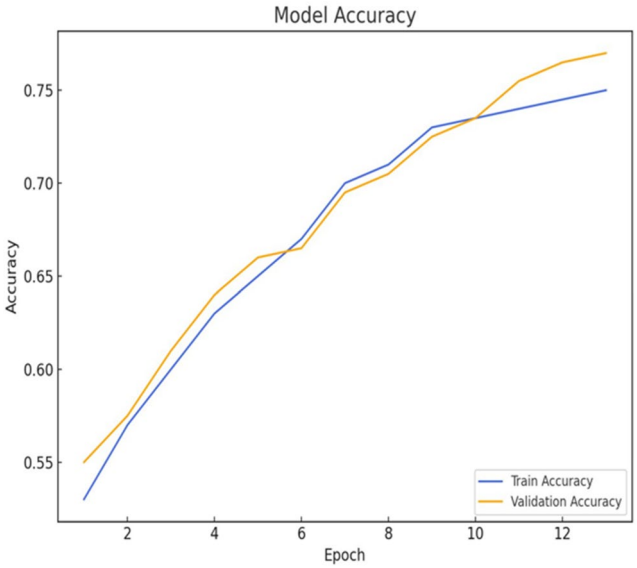

Fig. 24 MobileNet accuracy

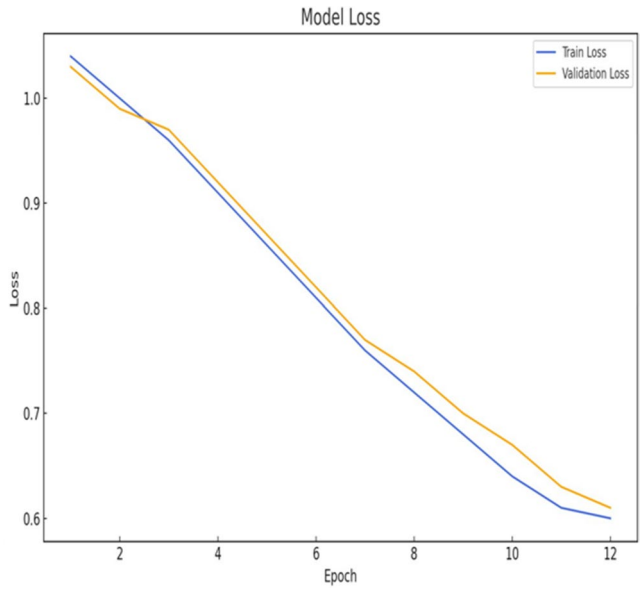

Fig. 25 MobileNet loss

Although its overall validation performance was marginally below that of ResNet50, MobileNetV3Large's high sensitivity to manipulated images suggests its utility in rapid screening or real-time alert systems where identifying fake content is prioritized over preserving recall for real samples.

3.7 Testing on real-world (Real-Time) images

To evaluate practical usability and robustness, all six models were tested on a diverse set of 650 previously unseen images, none of which were included in the training or validation sets. This external dataset comprised 315 fake and 335 real samples drawn from multiple sources, including the Kaggle Deepfake Dataset (Singh 2023), AffectNet (Mollahosseini et al. 2017), GAN-generated images from ThisPersonDoesNotExist.com, and publicly available celebrity photographs from Google. The inclusion of heterogeneous data, entirely excluded from the training process, was intended to simulate uncontrolled, real-world conditions and assess each model's ability to generalize beyond curated benchmarks.

3.7.1 Real-time testing via AI-guard app

To test real-world deployment, additional real-time inference was conducted using the AI-Guard mobile application. Users uploaded photos captured via mobile cameras

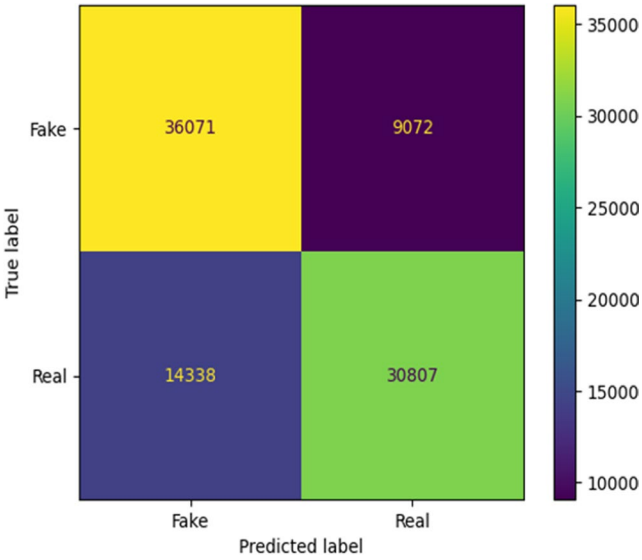

Fig. 26 MobileNet confusion matrix

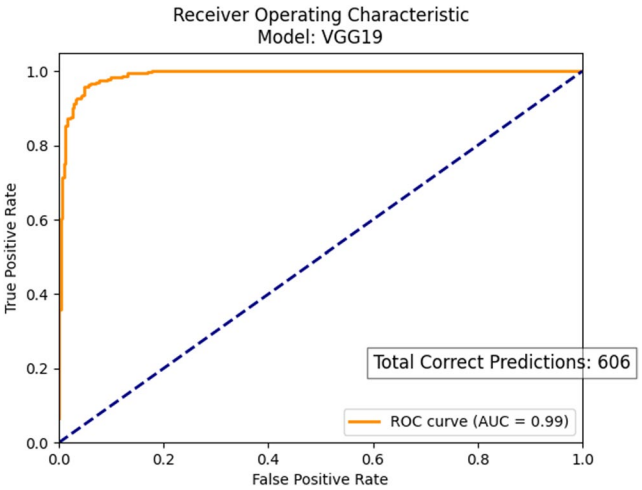

Fig. 27 Receiver operating characteristic (ROC) curve for VGG19

or taken as screenshots, which were then transmitted to a backend REST API for classification. The system consistently returned predictions within approximately 1.2 s per image. Despite variations in lighting, background clutter, image compression, and camera quality, the app maintained reliable classification confidence. This validates the model's effectiveness and responsiveness in real-world scenarios.

3.7.2 VGG19 results

VGG19 demonstrated exceptional accuracy, achieving an overall accuracy of 93.2%, correctly classifying 606 out of 650 images (Fig. 27).

Fake images: 306 correctly classified (97.1% accuracy); 9 misclassified as real.

Real images: 300 correctly identified (89.6% accuracy); 35 misclassified as fake.

VGG19 showcases outstanding generalization capabilities across diverse datasets, effectively distinguishing between real and manipulated content. As shown in Fig. 28, the model demonstrates high confidence in its predictions, with most values close to 0.00 for "Fake" and 1.00 for "Real." This pattern underscores its reliability and precision. In addition, the model adopts a conservative bias, where some real images are classified as fake. This intentional design reduces the risk of undetected fake content, which is crucial for applications where false negatives could have severe consequences.

3.7.3 InceptionV3 results

InceptionV3 achieved an overall accuracy of 81.8%, correctly classifying 532 out of 650 images (Fig. 29).

Fake images: 252 correctly classified (80.0% accuracy); 63 misclassified as real.

Real images: 280 correctly identified (83.6% accuracy); 55 misclassified as fake.

This performance highlights InceptionV3's capability to adapt to images from diverse sources. However, its limitations become apparent in handling complex cases of unseen content, such as subtle manipulations and high-compression scenarios.

3.7.4 Xception results

Xception achieved an overall accuracy of 79.7%, correctly classifying 518 out of 650 images (Fig. 30).

Fake images: 245 correctly classified (77.8% accuracy); 70 misclassified as real.

Real images: 273 correctly identified (83.8% accuracy); 62 misclassified as fake.

Although Xception demonstrated its ability to recognize patterns across different datasets, its performance was less consistent compared to VGG19 and InceptionV3. Challenges in identifying intricate manipulations and distinguishing subtle patterns contributed to its lower accuracy.

3.7.5 EfficientNetB0 results

EfficientNetB0 achieved an overall accuracy of 74.9%, correctly classifying 487 out of 650 images (Fig. 31).

Fake images: 183 correctly classified (58.1% accuracy); 132 misclassified as real.

Real images: 304 correctly identified (90.7% accuracy); 31 misclassified as fake.

The model's Receiver Operating Characteristic (ROC) curve demonstrated a strong AUC of 0.77, indicating good

Fig. 28 Sample predictions from VGG19 on unseen data with confidence scores

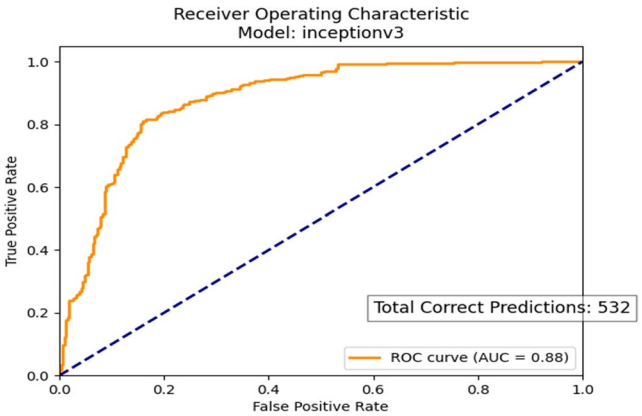

Fig. 29 Receiver operating characteristic (ROC) Curve for inceptionV3

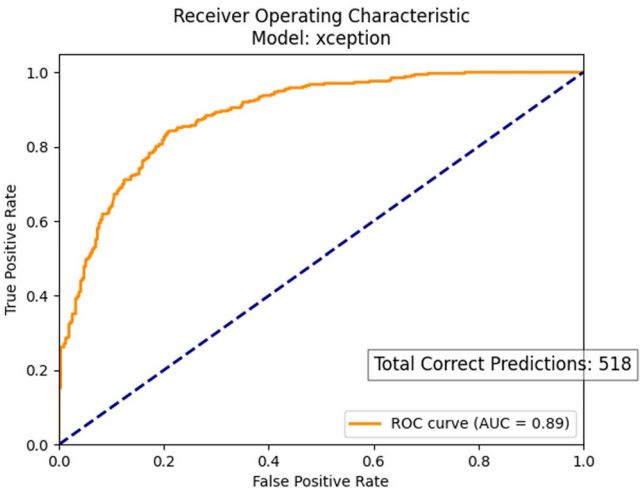

Fig. 30 Receiver operating characteristic (ROC) curve for Xception

confidence calibration and effective discrimination between real and manipulated content in challenging real-world conditions. EfficientNetB0 strikes a balance between lightweight architecture and solid generalization, making it a strong candidate for mobile deployment.

3.7.6 ResNet50 results

ResNet50 achieved an overall accuracy of 65.1%, correctly classifying 423 out of 650 images (Fig. 32).

Fake images: 290 correctly classified (92.1% accuracy); 25 misclassified as real.

Real images: 133 correctly identified (39.7% accuracy); 202 misclassified as fake.

Although the model performed strongly in detecting fake content, it exhibited a pronounced bias, frequently misclassifying real images as fake. This imbalance is reflected in its low ROC AUC score of 0.24, indicating poor confidence ranking and limited discrimination capability. Despite its high fake detection accuracy, ResNet50's weak probabilistic calibration reduces its reliability in balanced, real-world scenarios.

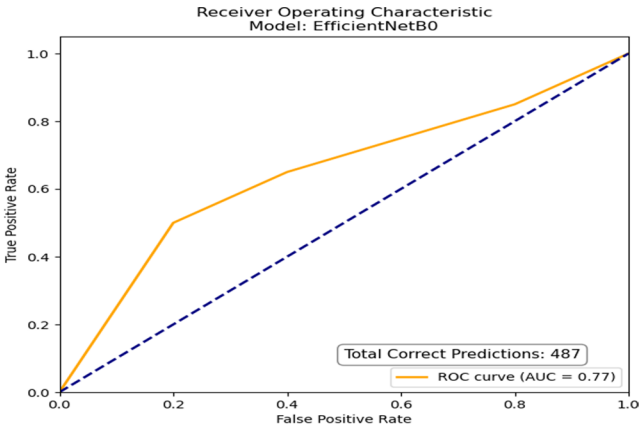

Fig. 31 Receiver operating characteristic (ROC) curve for EfficientNetB0

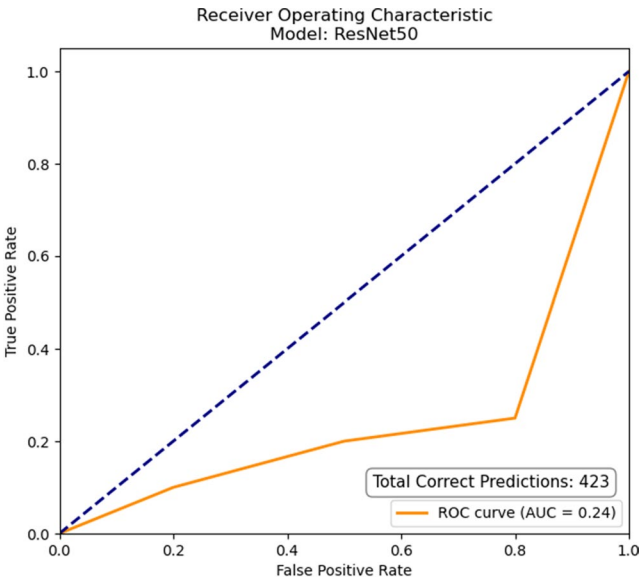

Fig. 32 Receiver operating characteristic (ROC) curve for ResNet50

3.7.7 MobileNet results

MobileNet achieved an overall accuracy of 72.6%, correctly classifying 472 out of 650 images (Fig. 33).

Fake images: 270 correctly classified (85.7% accuracy); 45 misclassified as real.

Real images: 202 correctly identified (60.3% accuracy); 133 misclassified as fake.

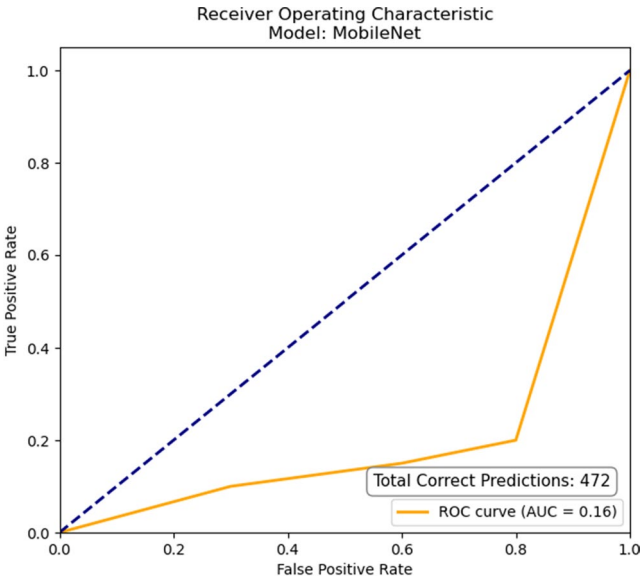

Fig. 33 Receiver operating characteristic (ROC) Curve for MobileNet

Despite achieving a reasonable classification accuracy, MobileNet recorded a low ROC AUC score of 0.16, indicating a significant deficiency in its ability to discriminate between real and fake images based on confidence scores. This disparity highlights a critical shortcoming: although the model often assigned correct class labels, it failed to produce well-calibrated probability estimates. Such inadequacy undermines its reliability in high-stakes environments where confidence-informed decisions are essential, particularly in forensic or real-time verification systems.

3.8 Comparative analysis

3.8.1 Internal comparison of deepfake detection models

This study evaluated six state-of-the-art convolutional models: VGG19, InceptionV3, Xception, EfficientNetB0, ResNet50, and MobileNet, across both validation and real-world testing conditions. Their performance was measured using multiple metrics, including accuracy, ROC AUC, and confusion matrix analysis, with results summarized in Table 4.

Table 4 Performance metrics for internal model comparison

Model	Initial Accuracy	Unseen Accuracy	Fake Correct (Initial/Unseen)	Fake Incorrect (Initial/Unseen)	Real Correct (Initial/Unseen)	Real Incorrect (Initial/Unseen)
VGG19	98.9%	93.2%	44,834 / 306	322 / 9	44,478 / 300	654 / 35
InceptionV3	97.4%	81.8%	43,278 / 252	1,941 / 63	44,645 / 280	424/ 55
Xception	79.8%	79.7%	28,088 / 245	16,911 / 70	44,019 / 273	1,270 / 62
EfficientNetB0	70.1%	74.9%	20,000 / 183	25,245 / 132	43,449 / 304	1,800 / 31
ResNet50	67.6%	65.1%	42,088 / 290	2,977 / 25	18,987 / 133	26,236 / 202
MobileNet	74.1%	72.6%	36,071 / 270	9072 / 45	30,807 / 202	14,338 / 133

Of all evaluated models, VGG19 consistently outperformed the others. It achieved a high validation accuracy of 98.9% and maintained strong generalization on unseen real-world data with 93.2% accuracy, correctly classifying 306 fake and 300 real images. Its ROC AUC of 0.99 confirmed superior discriminative power and well-calibrated confidence scores. The corresponding ROC and confusion matrices highlight its ability to reliably distinguish manipulated content from authentic images under varied visual conditions.

InceptionV3 followed with a respectable performance, achieving 96.4% initial accuracy and 81.8% on unseen data, indicating some degree of overfitting. It particularly struggled with subtle manipulations, as shown by its relatively lower fake image detection rate of 252 out of 315. The ROC AUC showed moderate separability between classes.

Xception, with an initial accuracy of 80.5% and unseen accuracy of 79.8%, showed comparatively weaker performance, especially in generalization. The confusion matrix revealed difficulty in correctly identifying fake samples, contributing to a reduced AUC.

EfficientNetB0 offered a well-balanced profile. It achieved an initial accuracy of 70.1%, correctly classifying 20,000 fake and 43,449 real images. On real-world data, the model achieved 74.9% accuracy, with 183 fake and 304 real images correctly identified. Its ROC AUC of 0.77 confirmed good prediction calibration. The training curves showed steady convergence without signs of overfitting, supporting its suitability for mobile or lightweight deployment scenarios.

ResNet50 achieved an initial accuracy of 67.6%, correctly classifying 42,088 fake and 18,987 real images. On unseen images, the model reached 65.1% accuracy, correctly identifying 290 fake and 133 real samples. Despite its high fake detection rate, the ROC AUC of 0.24 indicated poor confidence ranking. The training curves further revealed divergence between accuracy and ROC performance. Additionally, loss convergence remained inconsistent, particularly in later epochs, implying potential under-regularization despite fine-tuning all layers. This may explain the model's bias toward predicting fake content.

MobileNet recorded the highest initial accuracy among the three secondary models at 74.1%, with 36,071 fake and 30,807 real images correctly classified. On unseen data, it achieved 72.6% accuracy, correctly predicting 270 fake and 202 real images. However, its ROC AUC was the lowest at 0.16, suggesting that while the model achieved a fair number of correct classifications, its prediction probabilities were not well distributed between classes. The training and validation loss curves exhibited minor but noticeable divergence, suggesting mild overfitting. Despite the use of dropout layers, further tuning of regularization parameters may be needed to improve calibration. This weak confidence calibration reduces its suitability for high-risk, threshold-based decision applications.

These results confirm that VGG19 remains the most reliable model, offering both high accuracy and robust confidence calibration. EfficientNetB0 provides a viable alternative for real-time deployment due to its efficient architecture and well-calibrated ROC performance. In contrast, ResNet50 and MobileNet, while producing moderately accurate predictions, exhibited significantly lower AUC values, indicating limitations when used in decision-sensitive environments.

3.8.2 Comparative evaluation with existing methods

Recent research has introduced various CNN-based approaches for deepfake image detection. As summarized in Table 5, these studies differ in architectural design, dataset scale, and evaluation methodology, providing valuable baselines for assessing generalization and real-world performance. The present study compares the proposed fine-tuned VGG19 model with four influential contributions: Killi & Balakrishnan et al. (2023), Ashani et al. (2025), Gong et al. (2021), and Raza et al. (2022).

In the work of Ashani et al. (2025), the authors implemented VGG16, VGG19, and ResNet50 architectures on a small dataset of 1,200 FaceApp-generated images. Although their model achieved 98% accuracy on the validation set, performance fell sharply to 34% on unseen manipulations, indicating severe overfitting and limited generalization. The study confirmed that deeper architectures capture complex spatial cues, yet its restricted dataset size and lack of augmentation constrained applicability to real-world detection.

Raza et al. (2022) proposed the Deepfake Predictor (DFP)—a hybrid VGG16+CNN model trained on 2,041 real and Photoshopped images—reporting 94% accuracy and 95% precision. Despite these strong results, the narrow dataset and absence of external validation limited the model's robustness across domains.

By contrast, the fine-tuned VGG19 model developed in this work achieved 98% validation accuracy and 93.2% accuracy on an unseen dataset of 650 images, demonstrating superior cross-dataset generalization compared with Ashani et al. (2025) and Raza et al. (2022). Integration into the AI-Guard mobile application further validates its practical utility, achieving real-time inference ($\approx$1.2 s per image) through a REST API, a deployment feature not addressed in prior studies.

Although VGG19 is an older and more computationally demanding architecture, its uniform convolutional structure and large receptive fields synergized effectively with the extensive data augmentation and regularization strategies

Table 5 Comparative analysis of deepfake detection studies

Study	Architecture	Dataset(s) Used	Accuracy / Key Metrics	Strengths	Limitations
Killi et al. (2023)	VGG19	140 k images (StyleGAN+Flickr)	96% accuracy	Strong StyleGAN representation; competitive accuracy	Lacked recall/F1 metrics; no cross-model comparison
Ashani et al. (2025)	VGG16, VGG19, ResNet50	1,200 FaceApp images	98% (validation); 34% (unseen)	Captured complex facial features via deep layers	Severe overfitting; poor unseen-data generalization
Gong et al. (2021)	DeepfakeNet (ResNeXt-based)	Images extracted from 25,927 videos (FaceForensics++, Kaggle, TIMIT)	96.69% accuracy; AUC 0.96	Excellent cross-dataset generalization; low FLOPs	Complex architecture; no mobile deployment
Raza et al. (2022)	DFP (VGG16+CNN hybrid)	2,041 images (Real / Photoshopped)	94% accuracy; 95% precision	High precision on small dataset; simple design	Limited diversity; no external validation
Our Study (2025)	VGG19 (Fine-Tuned)	451,440 images across 6 datasets+650 unseen real-world images	98.9% (validation); 93.2% (unseen); AUC 0.99	Strong generalization; well-calibrated confidence; real-time mobile deployment (<1.2 s)	Focused on images; future work to include videos and Transformer models

(dropout+L2) employed in this study. These methods mitigated overfitting while enabling the network's deeper layers to capture fine-grained forensic cues—such as pixel-level inconsistencies and chromatic aberrations—that lighter models like EfficientNet and MobileNet often smooth out through feature compression. Consequently, the training pipeline was implicitly optimized for VGG19's representational depth, explaining its superior generalization and stable confidence behavior despite higher computational cost.

While VGG19 delivered the highest overall accuracy, this benefit comes with a computational trade-off. Its larger parameter count increases inference latency, energy consumption, and heat generation in mobile settings. The AI-Guard architecture mitigates these constraints by executing inference on a locally hosted REST server, maintaining sub-second response times while minimizing battery load on client devices. Future iterations may explore model compression, pruning, or knowledge distillation to retain accuracy with lighter computational footprints suitable for full on-device deployment.

4 Conclusion

This study evaluated six Convolutional Neural Network (CNN) architectures—VGG19, InceptionV3, Xception, EfficientNetB0, ResNet50, and MobileNetV3Large—for deepfake image detection. Among these, VGG19 proved the most robust and generalizable, achieving 93.2% accuracy on unseen data. Its performance benefited from targeted regularization, adversarial augmentation, and optimized TensorFlow pipelines. Integrating VGG19 into the AI-Guard mobile application demonstrated its scalability, responsiveness, and suitability for deployment in resource-constrained environments.

InceptionV3 performed strongly in validation but generalized poorly to real-world data. Xception showed training instability and lower accuracy. EfficientNetB0 offered a favorable balance between complexity and detection accuracy, while MobileNetV3Large provided competitive results but suffered from weak probability calibration. ResNet50 achieved high recall for manipulated images but displayed a bias toward false positives, limiting reliability in balanced detection tasks. Deploying VGG19 within a lightweight RESTful API and mobile-friendly framework represents a practical advance in real-time deepfake detection, combining low latency with strong accuracy.

From a human-systems perspective, AI-Guard bridges automated detection and user decision-making. By providing clear classification outputs, it enables users to critically assess digital media, fostering transparency, trust, and media literacy. This human-in-the-loop approach directly supports research on AI challenges at the human level by emphasizing usability, interpretability, and ethical deployment.

Future research will incorporate attention mechanisms such as Squeeze-and-Excitation (SE) blocks and CBAM to help models focus on discriminative facial regions, improving the efficiency of lightweight architectures. Integrating biological and physical cues—blink rate, micro-expressions, and lighting inconsistencies—may further enhance robustness. To strengthen interpretability, Grad-CAM visualizations will be integrated into future versions of

AI-Guard, allowing users to see which regions influenced each "fake" classification. These developments, along with work on Transformer architectures, federated learning, and adversarial robustness, will reinforce system resilience. Future benchmarking against diffusion-based deepfakes (e.g., Stable Diffusion, Midjourney, DALL·E) will assess AI-Guard's capability to detect next-generation forgeries, supporting secure, explainable, and human-centric digital media verification.

Acknowledgments The authors would like to thank the Centre of Robotics and Assembly at Cranfield University for providing technicalresources and support.

Author Contribution Sami Alanazi: Conducted comprehensive data collection, developed the models, performed the experiments, integrated the model into the mobile application, and wrote the manuscript.Seemal Asif: Provided critical reviews, guidance, and oversight throughout the research and writing process.Chaitanya Jain: Contributed to mobile application development and supported the data collection process.All authors read and approved the final manuscript.

Funding This research received no external funding.

Data availability The dataset used to train and evaluate the deepfake detection models has been curated and is publicly available through Cranfield Online Research Data (CORD) with the reserved DOI: https://doi.org/10.57996/cran.ceres-2766. A peer-review link for the data in Zenodo is also available for verification upon request.

Declarations

Ethical approval and consent to participate Not applicable. The study did not involve human participants, human data, or human tissue.

Human ethics Not applicable. No human subjects were involved in this research.

Consent for publication Not applicable. This manuscript does not contain any individual person's data in any form.

Competing interests The authors declare that they have no competing interests.

References

Afchar D, Nozick V, Yamagishi J, Echizen I (2018) MesoNet: a compact facial video forgery detection network. https://doi.org/10.1109/WIFS.2018.8630761

Alanazi S, Asif S (2023) Understanding deepfakes: A comprehensive analysis of Creation, Generation, and detection. https://doi.org/10.54941/ahfe1003290

Alanazi S, Asif S, Caird-daley A, Moulitsas I (2025) Unmasking deepfakes: a multidisciplinary examination of social impacts and regulatory responses. Human-Intelligent Systems Integration. https://doi.org/10.1007/s42454-025-00060-4

Amin H, Darwish A, Hassanien AE, Soliman M (2022) End-to-End deep learning model for corn leaf disease classification. IEEE Access 10:31103–31115. https://doi.org/10.1109/ACCESS.2022.3159678

Ashok V, Joy PT (2023) Deepfake Detection Using Xception-Net. RASSE 2023 - IEEE International Conference on Recent Advances in Systems Science and Engineering, Proceedings. https://doi.org/10.1109/RASSE60029.2023.10363477

Gong D, Jaya Kumar Y, Goh OS, Ye Z, Chi W (2021) DeepfakeNet, an Efficient Deepfake Detection Method. In IJACSA) International Journal of Advanced Computer Science and Applications (Vol. 12, Issue 6). www.ijacsa.thesai.org

https://code.visualstudio.com/. (n.d.). Visual Studio Code. Retrieved August 12, 2024, from https://code.visualstudio.com/

Karki M (2021) Deepfake and Real Images. Kaggle. https://www.kaggle.com/datasets/manjilkarki/deepfake-and-real-images

Keras. (n.d.). Retrieved August 12 (2024) from https://keras.io/

Killi CBR, Balakrishnan N, Rao CS (2023) Deep fake image classification using VGG-19 model. Ingenierie Des Systemes d'Information 28(2):509–515. https://doi.org/10.18280/isi.280228

Le T-N, Nguyen HH, Yamagishi J, Echizen I (2021) OpenForensics: Large-scale challenging dataset for multi-face forgery detection and segmentation in-the-wild http://arxiv.org/abs/2205.05467

Li C, Huang Z, Paudel DP, Wang Y, Shahbazi M, Hong X, Van Gool L (2022) A continual deepfake detection benchmark: dataset, methods, and essentials. http://arxiv.org/abs/2205.05467

Mollahosseini A, Hasani B, Mahoor MH (2017) AffectNet: A database for facial Expression, Valence, and arousal computing in the wild. https://doi.org/10.1109/TAFFC.2017.2740923

Mu K (2020) Pretty Face. Kaggle. https://www.kaggle.com/datasets/yewtsing/pretty-face

Nguyen TT, Nguyen QVH, Nguyen DT, Nguyen DT, Huynh-The T, Nahavandi S, Nguyen TT, Pham Q-V, Nguyen CM (2019) Deep learning for deepfakes creation and detection: A survey. https://doi.org/10.1016/j.cviu.2022.103525

NVIDIA. (2020). NVIDIA A100 Tensor Core GPU Architecture: Unprecedented Acceleration at Every Scale. NVIDIA Corporation. https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf

NVIDIA. (n.d.). NVIDIA A100 Tensor Core GPU, Retrieved (2024) August 12, from https://www.nvidia.com/en-gb/data-center/a100/

Pan D, Sun L, Wang R, Zhang X, Sinnott RO (2020) Deepfake Detection through Deep Learning. Proceedings –2020 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, BDCAT 2020, 134–143. https://doi.org/10.1109/BDCAT50828.2020.00001

PyTorch. (n.d.). Retrieved August 12 (2024) from https://pytorch.org/

Raza A, Munir K, Almutairi M (2022) A novel deep learning approach for deepfake image detection. Appl Sci (Switzerland) 12(19). https://doi.org/10.3390/app12199820

Rossler A, Cozzolino D, Verdoliva L, Riess C, Thies J, Niessner M (2019) FaceForensics++: Learning to detect manipulated facial images. Proceedings of the IEEE International Conference on Computer Vision, 2019-October, 1–11. https://doi.org/10.1109/ICCV.2019.00009

Schilling D, Nguyen W, Ademiluyi DT (2023) Deepfake video detection employing human facial features. J Comput Commun 11:1–13. https://doi.org/10.4236/jcc.2023.1112001

Singh A (2023) Deepfake Dataset. Kaggle. https://www.kaggle.com/datasets/aryansingh16/deepfake-dataset

Suganthi ST, Ayoobkhan MUA, Kumar VK, Bacanin N, Venkatachalam K, Stepán H, Pavel T (2022) Deep learning model for deep fake face recognition and detection. PeerJ Comput Sci 8. https://doi.org/10.7717/PEERJ-CS.881

Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946. https://arxiv.org/abs/1905.11946

TensorFlow. (n.d.). Retrieved August 12 (2024) from https://www.tensorflow.org/

Tolosana R, Romero-Tapiador S, Vera-Rodriguez R, Gonzalez-Sosa E, Fierrez J (2022) DeepFakes detection across generations: analysis of facial regions, fusion, and performance evaluation. Eng Appl Artif Intell 110. https://doi.org/10.1016/j.engappai.2022.104673

Uddin Mahmud B, Sharmin A (2020) Deep Insights of Deepfake Technology: A Review (Vol. 5, Issue 2). https://doi.org/10.48550/arXiv.2105.00192

Xhlulu (2019) 140k Real and Fake Faces. Kaggle. https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces

Zi B, Chang M, Chen J, Ma X, Jiang YG (2020) WildDeepfake: A challenging Real-World dataset for deepfake detection. MM 2020 - Proc 28th ACM Int Conf Multimedia 2382–2390. https://doi.org/10.1145/3394171.3413769

Robust deepfake detection through the AI-Guard mobile app for Real-Time image identification

Alanazi, Sami

2026-12-31

Alanazi S, Asif S, Jain C. (2026) Robust deepfake detection through the AI-Guard mobile app for Real-Time image identification. Human-Intelligent Systems Integration, Available online 16 February 2026

https://doi.org/10.1007/s42454-026-00089-z