



RESEARCH ARTICLE

Classification of Real and AI-Generated Face Images Using a Frozen MobileNetV2 Transfer Learning Model with a Web Demonstration

Annur Farida^{1,*}, Ratri Enggar Pawening¹, Wahab Sya'roni¹

¹Department of Informatics Engineering, Faculty of Engineering, Universitas Nurul Jadid, Probolinggo, Indonesia

*Corresponding author: annurfarida42@gmail.com

ARTICLE INFO

Article history:

Received: 30 October 2026

Revised: 16 November 2026

Accepted: 26 November 2026

Published: 10 December 2026

Keywords:

Deepfake detection

Synthetic image detection

Lightweight deep learning

Image classification

Flask

ABSTRACT

Image generators based on diffusion models can now produce faces that people can barely distinguish from photographs, which creates risks of impersonation and misinformation. This study examines how far a frozen, ImageNet-pretrained MobileNetV2 backbone with a small trainable classification head can separate real faces from faces generated by artificial intelligence, and it wraps the trained model in a Flask web application as a demonstration. The dataset contains 2,200 face images (1,050 real and 1,150 synthetic, produced by SDXL, FLUX.1 Pro, FLUX.1 Dev, and Gemini), split 70:15:15 into training, validation, and test subsets. The model has 2.42 million parameters, of which 164,353 were trained. On a held-out test set of 331 images drawn from the same pool, it reached an accuracy of 95.47% (95% confidence interval 92.7% to 97.2%), with F1-scores of 0.956 for synthetic and 0.953 for real faces; the 15 errors comprised 10 synthetic faces labelled real and 5 real faces labelled synthetic. Because the real images come mostly from one photographic collection and most synthetic images from one generated collection, the study cannot separate the detection of image generation from the recognition of the collection. Cross-source, per-generator, and baseline comparisons were not carried out, and training was run once. The results are therefore preliminary, and the main contribution is a transparent in-distribution baseline together with a working prototype.

I. INTRODUCTION

Generative models have changed what it means for an image to be “real.” Generative adversarial networks [1] and, more recently, latent diffusion models [2], [3] can synthesise human faces whose visual quality approaches that of photographs. As a result, a viewer can no longer rely on inspection alone to tell a photograph of a person from a face generated by artificial intelligence (AI). Such images can be used for identity fraud, the spread of false information, and the manipulation of opinion [4], [5], and the number and diversity of available generators keeps growing, as the scale of recent benchmarks for AI-generated images illustrates [6].

Convolutional neural networks (CNNs) are the most common basis for detecting manipulated or synthetic faces. Earlier studies applied CNNs to real and fake face images with different pre-processing pipelines [7], designed deeper CNN architectures trained on several data sources [8], and, more recently, compared CNNs with vision transformers [9], [10]. These studies show that learned detectors can be accurate, but they typically concentrate on benchmark accuracy, often with large networks, and the studies reviewed here rarely describe how the trained model reaches a person who simply wants to check an image. Lightweight backbones such as MobileNetV2 [11] were designed for settings with limited computational resources, and transfer learning from ImageNet [12] lets such a backbone be reused with little data and short training time. MobileNetV2 has already been applied to other face-related classification tasks, for example face-mask detection [13]. A recurring difficulty is that detectors trained on one set of generators or one image collection often transfer poorly to others [14], [15]. In the literature we reviewed (a non-systematic review), we also found little work that combines a MobileNetV2 transfer-learning classifier for real versus AI-generated faces with an interactive web application.

This paper addresses that gap. Its objective is to design, implement, and evaluate a system that classifies a face image as real or AI-generated using a CNN with MobileNetV2 transfer learning, and that makes the trained model usable through a web interface. The study is guided by one research question: how well can a frozen, ImageNet-pretrained MobileNetV2 backbone with a small classification head separate real from AI-generated faces on held-out data, and what does it take to expose the resulting model as a web application? The study is explicitly a first, in-distribution baseline; it does not test generalisation to other sources or generators.

The main contributions of this paper are as follows:

- A compact binary classifier that keeps the MobileNetV2 backbone frozen and trains only a head of 164,353 parameters, reaching 95.47% accuracy on 331 held-out test images from the same pool, reported as an in-distribution baseline.
- A 2,200-image dataset that combines real faces from the FFHQ-based part of a public Kaggle dataset, synthetic faces from SDXL, FLUX.1 Pro, and FLUX.1 Dev, and manually collected images, including Gemini-generated faces.
- A Flask web application that returns the predicted class, a confidence score, and the raw model output, and that rejects images without exactly one face.
- An error analysis based on the confusion matrix and on example predictions, showing that wrong predictions can carry high confidence.

II. RELATED WORK

CNN-based detection of fake faces. Abdul kareem and Altaei [7] classified real and fake faces from a public Kaggle collection with a CNN, testing a pre-processing chain (colour conversion from RGB to YCbCr, gamma correction, and Canny edge extraction) and principal component analysis (PCA). They found that the CNN without PCA performed better than the CNN with PCA, and their highest reported accuracy, 98.04%, was obtained with more training and test samples. Patel *et al.* [8] proposed a dense CNN for deepfake image detection and trained it on seven datasets to improve generalisation; they reported accuracies from 94.67% (StyleGAN2) to 99.33% (GDWCT), depending on the generator. Mu *et al.* [4] applied a CNN to identify real and deepfake faces and reported 91% accuracy.

Transformers and hybrid models. Arshed *et al.* [9] treated AI-generated face detection as a four-class problem (real faces, GAN-generated faces, Stable Diffusion faces, and StyleGAN2-encoded Stable Diffusion faces) and fine-tuned a vision transformer on image patches. It reached 99.90% accuracy, compared with 0.77 for a fine-tuned ResNet-50 and 0.94 for a fine-tuned VGG-16. Soudy *et al.* [10] trained a CNN, a CNN–vision-transformer hybrid, and majority voting on FaceForensics++ and DFDC; their abstract reports 97% accuracy for the CNN and 85% for the hybrid on FaceForensics++.

Lightweight CNNs for face analysis. Naufal and Kusuma [13] compared MobileNetV2, VGG16, DenseNet201, and Xception for face-mask classification with 5-fold cross-validation. Xception obtained the highest accuracy (0.988), while MobileNetV2 was almost as accurate (0.981) and the fastest, needing 4,081 s for training and testing against 18,274 s for Xception. This trade-off between accuracy and computation motivates the use of MobileNetV2 here.

Generalisation across generators. A recurring finding is that detectors trained on one family of generators do not automatically work on others. Wang *et al.* [16] trained a classifier on images from a single GAN and found that, with suitable pre- and post-processing and augmentation, it generalised to several unseen CNN-based generators. Ojha *et al.* [14] reported that classifiers trained only on GAN images fail on newer generative models and proposed using features of a large pretrained vision–language model instead of a trained real-versus-fake classifier. Corvi *et al.* [15] asked whether GAN-oriented detectors still work on diffusion-model images and tested them under conditions typical of social networks, such as compression and resizing. The GenImage benchmark [6] and the FaceForensics++ benchmark [17] provide large collections for cross-generator and cross-manipulation tests. The present study does not perform such tests, which is its main limitation (Section V).

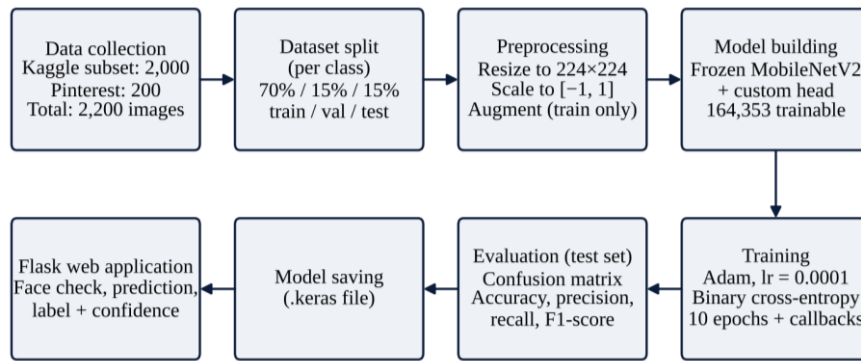
Table 1 summarises these studies. The studies used different datasets and protocols, so their numbers cannot be ranked against each other or against ours. What distinguishes the present work is the combination of a lightweight, mostly frozen backbone, a binary real-versus-AI face task with images from several recent generators, and integration into a web application.

Table 1. Summary of related studies and their difference from this study

Study	Task and data	Method	Reported result	Difference from this study
Abdul kareem & Altaei [7]	Real vs. fake faces (Kaggle)	CNN, with/without pre-processing and PCA	Up to 98.04% accuracy	CNN from scratch; no web application
Patel <i>et al.</i> [8]	Deepfake images from several sources	Dense CNN	94.67% to 99.33%, by generator	Larger architecture; no web application
Mu <i>et al.</i> [4]	Real vs. deepfake faces	CNN	91% accuracy	No transfer learning with a lightweight backbone
Arshed <i>et al.</i> [9]	Four classes, 10,000 images each (real, GAN, diffusion)	Patch-wise vision transformer	99.90% accuracy (ResNet-50: 0.77; VGG-16: 0.94)	Transformer; multiclass; no web application
Soudy <i>et al.</i> [10]	FaceForensics++ and DFDC	CNN, CNN-ViT hybrid, voting	CNN 97%; hybrid 85% (FaceForensics++)	Video-oriented; no web application
Naufal & Kusuma [13]	Face masks vs. no masks	MobileNetV2, VGG16, DenseNet201, Xception	Xception 0.988; MobileNetV2 0.981	Different task (mask detection)
This study	2,200 real and AI-generated faces	Frozen MobileNetV2 + custom head	95.47% accuracy (331 test images)	Real vs. AI faces; integrated Flask web application

III. MATERIALS AND METHODS

This study follows a quantitative, experimental design. Fig. 1 gives an overview: data are collected and split, pre-processed, used to train a MobileNetV2-based classifier, evaluated on a held-out test set, and finally served through a web application.

**Fig. 1.** Overview of the research workflow, from data collection to the Flask web application.

A. Dataset

The dataset contains 2,200 face images in two classes, REAL and FAKE (Table 2). The main source is a public Kaggle collection of real and synthetic faces [18]. In that collection, the real images come from Flickr-Faces-HQ (FFHQ) [19], and the synthetic images were generated with SDXL [3], FLUX.1 Pro, and FLUX.1 Dev. A subset of 2,000 images was taken from it: 1,000 real images and 1,000 synthetic images (333 SDXL, 333 FLUX.1 Pro, and 334 FLUX.1 Dev). The sampling procedure used for this subset was not recorded. To add variety, 200 further images were collected manually from Pinterest: 150 synthetic faces found with the search keywords “AI face Gemini” and 50 photographs of real people. Every image was inspected visually to confirm that it matched its label. No image was captured by the authors, and no identity information was used. Class and source are strongly aligned: 1,000 of the 1,050 REAL images come from FFHQ, and 1,000 of the 1,150 FAKE images come from the three generators in the Kaggle collection (Table 2). This alignment limits what the test results can show (Section V).

Table 2. Composition of the dataset

Class	Source	Generator / origin	Images
REAL	Kaggle [18]	FFHQ photographs [19]	1,000
REAL	Pinterest	Photographs of real people	50
FAKE	Kaggle [18]	SDXL	333
FAKE	Kaggle [18]	FLUX.1 Pro	333
FAKE	Kaggle [18]	FLUX.1 Dev	334
FAKE	Pinterest	Gemini	150
Total		1,050 REAL + 1,150 FAKE	2,200

The dataset was divided per class in a 70:15:15 ratio into training, validation, and test subsets (Table 3). The validation subset was used to monitor training, and the test subset was not used for training or validation. Labels were derived from the folder structure of each subset: FAKE was assigned the numeric label 0 and REAL the label 1.

Table 3. Number of images per class in each subset

Class	Training	Validation	Test	Total
FAKE	805	172	173	1,150
REAL	735	157	158	1,050
Total	1,540	329	331	2,200

B. Proposed Method

1) Pre-processing and augmentation:

All images were resized to 224×224 pixels, the input size of MobileNetV2. Pixel values were rescaled from $[0, 255]$ to $[-1, 1]$ with the standard MobileNetV2 pre-processing function, which implements (1).

$$x' = \frac{x}{127.5} - 1 \quad (1)$$

where x is a pixel value of one colour channel and x' is the rescaled value. Data augmentation was applied to the training subset only and consisted of random rotation of up to $\pm 10^\circ$, random zoom of up to 10%, and random horizontal flipping. The validation and test subsets were only resized and rescaled.

2) Model architecture:

The classifier (Fig. 2) uses MobileNetV2 [11] without its original classification layers (*include_top* = false) as a feature extractor, which is the standard transfer-learning setting [20], initialised with ImageNet weights [12] and frozen so that these weights are not updated during training. The $7 \times 7 \times 1280$ feature map of the backbone is passed to a custom head. Global average pooling [21] reduces each feature map to its mean, as in (2),

$$\text{GAP}_c = \frac{1}{HW} \sum_{i=1}^H \sum_{j=1}^W X_{i,j,c} \quad (2)$$

where $X_{i,j,c}$ is the activation at position (i, j) of feature map c , and H and W are the height and width of the feature map, giving a 1,280-dimensional vector. This vector feeds a dense layer with 128 units and ReLU activation, followed by batch normalisation [22] and dropout [23] with a rate of 0.2. The output layer is a single dense unit with a sigmoid activation, (3),

$$p = \sigma(z) = \frac{1}{1 + e^{-z}} \quad (3)$$

where z is the output of the final linear transformation and p is interpreted as the probability that the image is REAL. The model has 2,422,593 parameters in total (9.24 MB), of which 164,353 are trainable and 2,258,240 are non-trainable; the non-trainable part consists of the frozen backbone and the moving statistics of the batch-normalisation layer.

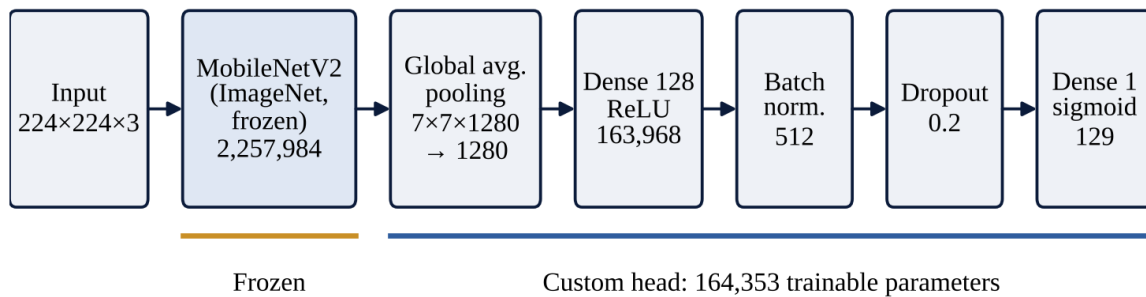


Fig. 2. Architecture of the classifier. Numbers below the layer names are parameter counts; the sigmoid output is the probability of the REAL class.

3) Inference and web application:

The trained model was saved in the Keras format and loaded by a Flask web application. The application has an input page for uploading an image, a prediction step, and an output page. After upload, a face-detection step checks that the image contains exactly one human face; if not, the image is rejected and the user is asked to upload a different one. Face

detection uses the MTCNN detector (multi-task cascaded convolutional network) from the Python package mtcnn with its default settings: the image is read with OpenCV, converted from BGR to RGB, and passed to the `detect_faces` function, and the number of detections is counted. Zero detections or more than one detection causes the image to be rejected with a different message in each case, and no extra confidence or size filter is applied. The detector was not evaluated separately. Accepted images are resized and rescaled as above and passed to the model. The model output p (the raw prediction) is compared with a threshold of 0.5: the image is labelled REAL if $p \geq 0.5$ and FAKE otherwise. The confidence shown to the user is the probability of the predicted class, (4).

$$\text{conf} = \begin{cases} 100 p, & p \geq 0.5 \\ 100 (1 - p), & p < 0.5 \end{cases} \quad (4)$$

The application shows the uploaded image, the label, the confidence, and the raw prediction. During development, it was run on a local server and accessed through a web browser.

C. Experimental Setup

The model was implemented in Python with TensorFlow/Keras and trained in Google Colab. It was compiled with the Adam optimiser [24] (learning rate 0.0001) and the binary cross-entropy loss, (5),

$$\mathcal{L} = -\frac{1}{N} \sum_{k=1}^N [y_k \log p_k + (1 - y_k) \log(1 - p_k)] \quad (5)$$

where N is the number of images in a batch, $y_k \in \{0, 1\}$ is the label of image k , and p_k is the predicted probability of the REAL class. Training ran for 10 epochs with a batch size of 32 (49 steps per epoch). Two callbacks monitored the validation loss: early stopping, which halts training after four epochs without improvement and restores the best weights, and a learning-rate reduction that multiplies the learning rate by 0.3 after two epochs without improvement, down to a minimum of 10^{-6} . Table 4 lists the settings. Each epoch took roughly two minutes (119–131 s in the training log). The training images were shuffled in each epoch, whereas the validation and test images were not. The code sets no random seed, and the accelerator type and library versions were not recorded; the model was trained once.

Table 4. Training configuration

Setting	Value
Input size / batch size	$224 \times 224 \times 3 / 32$
Backbone	MobileNetV2, ImageNet weights, all layers frozen
Classification head	Global average pooling, Dense (128, ReLU), batch normalisation, dropout (0.2), Dense (1, sigmoid)
Optimiser / learning rate	Adam / 0.0001
Loss function	Binary cross-entropy
Epochs	10
Early stopping	Monitor: validation loss; patience: 4; restore best weights
Learning-rate reduction	Monitor: validation loss; factor: 0.3; patience: 2; minimum: 10^{-6}
Augmentation (training only)	Rotation $\pm 10^\circ$, zoom 10%, horizontal flip
Decision threshold	0.5 on the sigmoid output (REAL if $p \geq 0.5$)

D. Evaluation Metrics

The final model was evaluated on the 331 test images. We report the confusion matrix and the accuracy, precision, recall, and F1-score [25], [26]. Following the definition used in this study, FAKE is treated as the positive class when computing per-class metrics, and the REAL class is evaluated in the same way with the roles reversed. The model outputs the probability of REAL, so the FAKE-positive quantities are derived from the FAKE row and the FAKE column of the confusion matrix. With true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), the metrics are given by (6)–(9):

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (6)$$

$$\text{Precision} = \frac{TP}{TP+FP} \quad (7)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (8)$$

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (9)$$

Macro averages are the unweighted mean of the two per-class values. Because the test set is small, we also report a 95% Wilson score interval for the accuracy, which we computed from the counts of correct and incorrect predictions. All metrics come from a single training run and a single data split.

IV. RESULTS

A. Training Behaviour

Fig. 3 shows the accuracy and loss over the 10 epochs. Training accuracy rose from 73.25% to 96.88%, and validation accuracy rose from 85.11% to 96.66%. Training loss fell from 0.5349 to 0.0876 and validation loss from 0.3650 to 0.1039. The validation loss decreased in every epoch, so its lowest value was reached at epoch 10, whereas the highest validation accuracy (96.96%) occurred at epoch 4 and the validation accuracy stayed between 96.35% and 96.96% from epoch 4 onward. Because the validation loss kept improving, neither callback was triggered: training ran for the full 10 epochs and the learning rate stayed at 0.0001. At epoch 10 the gap between training and validation accuracy was 0.22 percentage points. Training accuracy was still below 97% and the validation loss was still falling at epoch 10, so the model may be under-trained; the 10-epoch budget was not tuned.

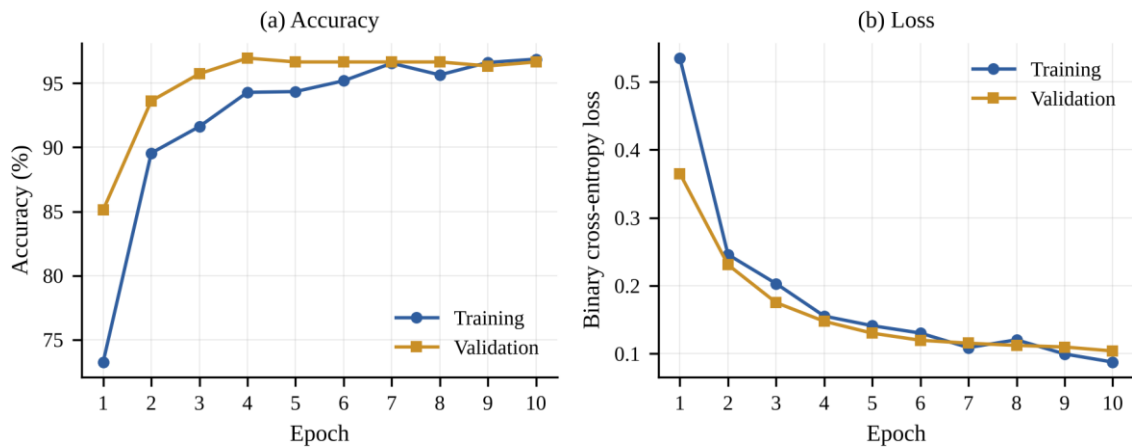


Fig. 3. Training and validation (a) accuracy and (b) loss over 10 epochs. Values are taken from the training log.

B. Test-Set Performance

On the test set, the model classified 316 of 331 images correctly, an accuracy of 95.47% (95% Wilson interval: 92.7% to 97.2%). Fig. 4 shows the confusion matrix: 163 of 173 FAKE images and 153 of 158 REAL images were classified correctly. The 15 errors consist of 10 FAKE images labelled REAL and 5 REAL images labelled FAKE. Table 5 gives the per-class metrics.

Actual class	FAKE	163 (94.2%)	10 (5.8%)
	REAL	5 (3.2%)	153 (96.8%)
		FAKE	REAL
		Predicted class	

Fig. 4. Confusion matrix on the test set (331 images). Percentages are row-wise.

Table 5. Per-class and average metrics on the test set

Class (support)	Precision	Recall	F1-score
FAKE (173)	0.970	0.942	0.956
REAL (158)	0.939	0.968	0.953
Macro average (331)	0.954	0.955	0.955

Note: accuracy is 0.9547 (316/331). Values were computed from the confusion matrix in Fig. 4.

Precision for FAKE (0.970) is higher than its recall (0.942), and the reverse holds for REAL (precision 0.939, recall 0.968): the model misses more synthetic faces than it falsely flags real ones. Table 6 lists the first and last example of each confusion-matrix category. Both misclassified FAKE examples were assigned to REAL with moderate confidence (60.04% and 77.35%), while one of the misclassified REAL examples was labelled FAKE with a confidence of 99.46%, which shows that high confidence does not guarantee a correct prediction.

Table 6. Example predictions from each confusion-matrix category (first and last case per category; not a random sample)

Category	Actual	Predicted	Raw output (p)	Confidence (%)
TP (1st)	FAKE	FAKE	0.0173	98.27
TP (163rd)	FAKE	FAKE	0.0547	94.53
FN (1st)	FAKE	REAL	0.6004	60.04
FN (10th)	FAKE	REAL	0.7735	77.35
FP (1st)	REAL	FAKE	0.1749	82.51
FP (5th)	REAL	FAKE	0.0054	99.46
TN (1st)	REAL	REAL	0.9998	99.98
TN (153rd)	REAL	REAL	0.9888	98.88

Note: the raw output p is the sigmoid output, interpreted as the probability of REAL; confidence follows (4). Because FAKE is the positive class, TP/FN refer to FAKE images and TN/FP to REAL images.

C. Web Application

Fig. 5 shows the web application. The input page offers a file chooser and an “Analyze Image” button. For a REAL test image, the application returned the label REAL with a raw prediction of 0.999325 and a confidence of 99.93%. For a FAKE image, the raw prediction was 0.098941, which gives a confidence of $(1 - 0.098941) \times 100\% = 90.11\%$. When a photograph of a cat was uploaded, the face-detection step found no human face, the model was not called, and the application displayed an “input not appropriate” message asking for an image with exactly one face. These cases demonstrate the behaviour of the application qualitatively; no response-time or user study was conducted.

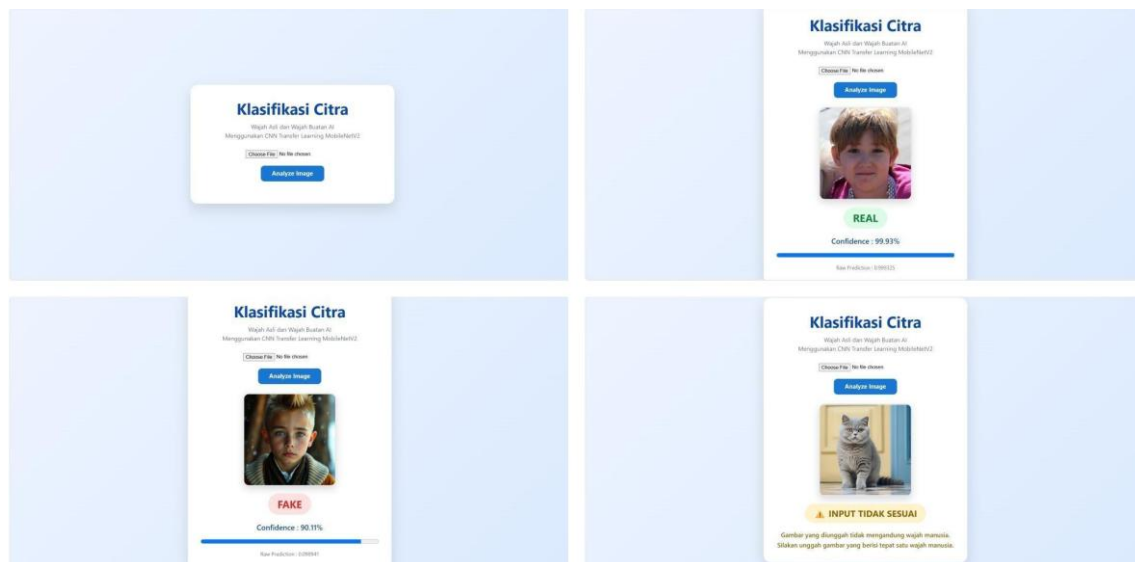


Fig. 5. Web application screenshots: (top left) input page; (top right) prediction REAL, confidence 99.93%; (bottom left) prediction FAKE, confidence 90.11%; (bottom right) rejection of an image without a human face.

V. DISCUSSION

The central result is that a frozen MobileNetV2 backbone with a 164,353-parameter head separates real from AI-generated faces with 95.47% accuracy on held-out images. The learning curves in Fig. 3 and the small difference between

training accuracy (96.88%), validation accuracy (96.66%), and test accuracy (95.47%) show no sign of overfitting in the usual sense, that is, a gap between training and held-out accuracy. This does not rule out shortcut learning, because a model that has learnt a property of the collections would score equally well on training, validation, and test images drawn from the same pool; and the still-falling validation loss suggests that the model may be under-trained rather than well fitted. Freezing the backbone and using augmentation, dropout, and batch normalisation limits the capacity that can be fitted to a training set of only 1,540 images. The validation subset holds 329 images, so one image corresponds to about 0.3 percentage points; the fluctuations of the validation accuracy after epoch 4 are therefore within what a single image or two can cause, and should not be read as a trend.

The most important limitation concerns what the result means. In this dataset, class is largely confounded with source: the real images are almost all curated FFHQ portraits, and most synthetic images come from a single collection of generated portraits (Table 2). A frozen network pretrained for object recognition can separate such collections using stylistic and photographic differences (for example lighting, depth of field, colour grading, resolution, or compression history) that have nothing to do with the process that created the image. The 95.47% accuracy should therefore be read as performance on this pool, not as evidence that the model detects AI generation in general. The 200 Pinterest images are the only material from outside the main collection; they were not evaluated separately, and no per-generator or cross-source results are available. These analyses, together with comparisons against a fine-tuned backbone, other lightweight models, and simple non-learned baselines, are the necessary next step.

The error pattern is asymmetric. The model failed to recognise 10 of 173 synthetic faces (5.8%) but mislabelled only 5 of 158 real faces (3.2%). For a screening tool, missing a synthetic face is usually the costlier error, so a different decision threshold could be preferable; this was not explored here. The examples in Table 6 add a caution for users of the application: the confidence value is the transformed sigmoid output, not a calibrated probability, and one wrongly classified real face received 99.46% confidence. The interface should therefore present the confidence as an indication of the model's certainty and not as evidence that the prediction is correct. The errors were not broken down by generator, so we cannot say which generator is hardest to detect.

Responsible use needs more than accuracy. The base rate matters: with the sensitivity for FAKE of 94.2% and the false-positive rate of 3.2% observed here (5 of 158 real faces), a stream of images in which 1% are synthetic would give a positive predictive value of only about 23%, and a stream with 10% synthetic images about 77% (an illustrative calculation that assumes these rates hold outside the test set). A FAKE label attached to a real person's photograph can also cause harm, so the output should be presented as an indication that calls for human verification, never as proof. Uploaded face images are personal data; a public deployment should state whether images are stored, for how long, and under what policy. We did not analyse performance by age, skin tone, or gender, and the FFHQ and web images are not demographically representative of any particular population, including Indonesia; a local, described test set would be a valuable extension.

Comparison with the literature requires care. The accuracy obtained here is lower than the 99.90% reported for a vision transformer on a four-class dataset [9] and within the 94.67% to 99.33% range reported for a CNN detector on other generators [8], though below the 97% to 98% reported elsewhere [7], [10], but the datasets, class definitions, and protocols differ, and no baseline was trained on our data. The comparison with [13], where MobileNetV2 was almost as accurate as a much slower network, supports the choice of a lightweight backbone, but we did not measure inference time or memory use ourselves. What the present study adds is a worked example of a small frozen model, an end-to-end evaluation protocol with a held-out test set, and a working web interface.

The practical implication is limited. A model with 2.4 million parameters is cheap to host, and freezing the backbone makes retraining cheap, but freezing reduces training cost only: inference cost is that of MobileNetV2 itself, and we did not measure it. The model should not be taken as a general-purpose detector. Several limitations restrict the conclusions. (i) The dataset is small and was split once, and the model was trained once, so the reported accuracy carries sampling uncertainty (the 95% interval spans about 4.5 percentage points) and the effect of random initialisation and of the split is unknown. (ii) The synthetic images come from only four generators, and the Gemini images (150) were collected from a single platform. The test set is drawn from the same pool as the training set, so we have no evidence about generators that were not seen in training, or about images that were re-compressed, resized, or edited, as happens on social media. (iii) Most real images come from FFHQ, which consists of curated, aligned photographs, so performance on casual photographs may differ. (iv) We did not check whether differences in resolution, file format, or background between the sources could act as shortcuts for the classifier. (v) No other backbone or fine-tuning strategy was compared, so we cannot tell how much accuracy was traded for a frozen backbone. (vi) The web application was tested locally with a few examples, the face-detection step was not evaluated separately, and the application has not been deployed or user-tested. (vii) The 10-epoch training budget was fixed in advance and no hyper-parameter search was done.

VI. CONCLUSION

This study designed, trained, and evaluated a classifier that separates real faces from AI-generated faces using a frozen MobileNetV2 backbone pretrained on ImageNet and a small trainable head, and integrated it into a Flask web application. On a held-out test set of 331 images drawn from a 2,200-image dataset, the model reached an accuracy of 95.47% (95% interval 92.7% to 97.2%) and F1-scores of 0.956 for synthetic and 0.953 for real faces, with 10 synthetic faces labelled

as real and 5 real faces labelled as synthetic. The web application returns a label, a confidence score, and the raw model output, and it rejects images that do not contain exactly one face.

These results show that a frozen MobileNetV2 backbone with a small trainable head separates the real and synthetic faces of this dataset well. Because class is largely confounded with image source, they do not establish that the model detects AI generation in general, and the study should be read as an in-distribution baseline with a working prototype. Future work should first test the model across sources and generators, and should also evaluate the model on generators and image-processing conditions that were not seen in training, report results per generator, repeat training with several random seeds, compare the frozen backbone with fine-tuned and alternative lightweight backbones, examine threshold selection and confidence calibration, and measure inference time in a deployed version of the application.

ACKNOWLEDGMENT

The authors thank the Informatics Engineering Study Program, Faculty of Engineering, Universitas Nurul Jadid, for academic support.

AUTHOR CONTRIBUTIONS

Annur Farida: Conceptualization, Methodology, Software, Investigation, Data curation, Writing – original draft. Ratri Enggar Pawening and Wahab Sya'roni: Supervision, Validation, Writing – review & editing.

DATA AVAILABILITY

The primary dataset is publicly available on Kaggle [18]. The 200 additional images were collected manually from Pinterest and are not redistributed. The trained model and source code are available from the corresponding author upon reasonable request.

DECLARATION OF COMPETING INTEREST

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

REFERENCES

- [1] I. Goodfellow et al., "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 27, Montréal, QC, Canada, 2014, pp. 2672–2680.
- [2] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, New Orleans, LA, USA, 2022, pp. 10684–10695.
- [3] D. Podell et al., "SDXL: Improving latent diffusion models for high-resolution image synthesis," 2023, *arXiv:2307.01952*.
- [4] J. Mu, M. Adrezo, and A. N. Haikal, "Identifikasi wajah asli dan buatan deepfake menggunakan metode convolutional neural network," *Teknika*, vol. 13, no. 1, pp. 45–50, 2024, doi: 10.34148/teknika.v13i1.705.
- [5] A. Firc, K. Malinka, and P. Hanáček, "Deepfakes as a threat to a speaker and facial recognition: An overview of tools and attack vectors," *Heliyon*, vol. 9, no. 4, art. no. e15090, 2023, doi: 10.1016/j.heliyon.2023.e15090.
- [6] M. Zhu et al., "GenImage: A million-scale benchmark for detecting AI-generated image," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS) Datasets Benchmarks Track*, vol. 36, 2023.
- [7] H. S. Abdul kareem and M. S. M. Altaei, "Detection of deep fake in face images using deep learning," *Wasit J. Comput. Math. Sci.*, vol. 1, no. 4, pp. 60–71, 2022, doi: 10.31185/wjcm.92.
- [8] Y. Patel et al., "An improved dense CNN architecture for deepfake image detection," *IEEE Access*, vol. 11, pp. 22081–22095, 2023, doi: 10.1109/ACCESS.2023.3251417.
- [9] M. A. Arshed, S. Mumtaz, M. Ibrahim, C. Dewi, M. Tanveer, and S. Ahmed, "Multiclass AI-generated deepfake face detection using patch-wise deep learning model," *Computers*, vol. 13, no. 1, art. no. 31, 2024, doi: 10.3390/computers13010031.
- [10] A. H. Soudy et al., "Deepfake detection using convolutional vision transformers and convolutional neural networks," *Neural Comput. Appl.*, vol. 36, no. 31, pp. 19759–19775, 2024, doi: 10.1007/s00521-024-10181-7.
- [11] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Salt Lake City, UT, USA, 2018, pp. 4510–4520.
- [12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Miami, FL, USA, 2009, pp. 248–255.
- [13] M. F. Naufal and S. F. Kusuma, "Pendeteksi citra masker wajah menggunakan CNN dan transfer learning," *J. Teknol. Inf. Ilmu Komput.*, vol. 8, no. 6, pp. 1293–1300, 2021, doi: 10.25126/jtiik.2021865201.
- [14] U. Ojha, Y. Li, and Y. J. Lee, "Towards universal fake image detectors that generalize across generative models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2023, pp. 24480–24489.
- [15] R. Corvi, D. Cozzolino, G. Zingarini, G. Poggi, K. Nagano, and L. Verdoliva, "On the detection of synthetic images generated by diffusion models," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2023, doi: 10.1109/ICASSP49357.2023.10095167.
- [16] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, "CNN-generated images are surprisingly easy to spot... for now," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 8695–8704.
- [17] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "FaceForensics++: Learning to detect manipulated facial images," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2019, pp. 1–11, doi: 10.1109/ICCV.2019.00009.

- [18] shreyanshpate11, “130k real vs fake face,” Kaggle dataset. [Online]. Available: <https://www.kaggle.com/datasets/shreyanshpate11/130k-real-vs-fake-face>
- [19] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, 2019, pp. 4401–4410.
- [20] L. Alzubaidi *et al.*, “Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions,” *J. Big Data*, vol. 8, art. no. 53, 2021, doi: 10.1186/s40537-021-00444-8.
- [21] M. Lin, Q. Chen, and S. Yan, “Network in network,” 2013, *arXiv:1312.4400*.
- [22] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, Lille, France, 2015, pp. 448–456.
- [23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.*, vol. 15, no. 56, pp. 1929–1958, 2014.
- [24] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, 2015.
- [25] S. A. Hicks *et al.*, “On evaluation metrics for medical applications of artificial intelligence,” *Sci. Rep.*, vol. 12, art. no. 5979, 2022, doi: 10.1038/s41598-022-09954-8.
- [26] D. J. Hand, P. Christen, and N. Kirielle, “F: An interpretable transformation of the F-measure,” *Mach. Learn.**, vol. 110, no. 3, pp. 451–456, 2021, doi: 10.1007/s10994-021-05964-1.