International Journal of Artificial Intelligence and Mathematical Sciences

January - June 2026 Vo5. Issue No. 01
ISSN Online: 2958-0153 and Print: 2958-5228
DOI: https://doi.org/10.58921/ijaims.v5i1.186

# Real-Time Face Mask Detection Using Transfer Learning with MobileNetV2

[1*] Memona Muhammad Younus

memona.m.younus@gmail.com

[1]Faculty of Computing, Szabist University, Karachi, Pakistan.

**Received:** 01-03-2026; **Accepted:** 18-05-2026; **Published**: 01-07-2026

**Abstract**

The COVID-19 pandemic created an urgent need for automated systems capable of verifying face-mask compliance in public spaces. This paper presents a lightweight, real-time face-mask classifier built on transfer learning with the MobileNetV2 architecture. A pretrained ImageNet backbone is used as a frozen feature extractor with a compact classification head, followed by a fine-tuning phase that unfreezes the final convolutional layers. The model is trained and evaluated on the publicly available. Face Mask ~12K Images dataset, comprising approximately twelve thousand pre-cropped and class-balanced face images split into training, validation, and test partitions. Using data augmentation, two-phase training, and standard regularization, the classifier attains approximately 99% accuracy on the held-out test set with near-perfect precision and recall for both the masked and unmasked classes. The results confirm that a low-compute, mobile-oriented backbone combined with transfer learning is sufficient for accurate binary mask detection, making the approach suitable for deployment on edge devices. The proposed pipeline is a clean, reproducible, end-to-end implementation rather than a novel methodology.

**Keywords:** Face mask detection, convolutional neural networks, transfer learning, MobileNetV2, image classification, computer vision, COVID-19.

*Email: memona.m.younus@gmail.com



## 1. INTRODUCTION

The COVID-19 pandemic generated widespread interest in automated systems for verifying mask compliance in public environments such as transit hubs, hospitals, and retail spaces. Manual monitoring is labor-intensive and difficult to scale, which motivated a large body of computer-vision research aimed at detecting whether individuals are wearing face masks. Face mask detection sits at the intersection of two well-studied problems: face detection, which localizes faces within an image, and bi-nary image classification, which assigns each detected face to a masked or unmasked category.

A complete deployed system typically follows a two-stage pipeline. In the first stage, a face-detection algorithm produces bounding boxes for each face in the input frame. In the second stage, each cropped face is passed to a classifier that outputs the mask/no-mask decision. This paper focuses on the second stage: a high-accuracy binary classifier trained on pre-cropped faces. The classifier is deliberately lightweight so that it can run in real time on commodity or edge hardware.

The contribution of this work is not a new methodology but a clean, reproducible, end-to-end implementation of an established approach. We employ MobileNetV2 as a feature ex-tractor, transfer learning from ImageNet weights, and a small, fine-tuned classification head. We show that this combination achieves approximately 99% accuracy on a recognized public dataset and we describe the full training and evaluation workflow in sufficient detail to be reproduced.

The remainder of this paper is organized as follows. Section II reviews convolutional neural networks for image classification. Section III discusses transfer learning. Section IV surveys face detection, and Section V reviews prior mask-detection systems. Section VI describes the dataset. Section VII details the proposed methodology, and Section VIII reports experimental results. Section IX discusses the limitations of the study, and Section X concludes the paper and outlines directions for future work.

## 2. CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE CLASSIFICATION

Modern image classification was reshaped by Krizhevsky et al. [1] with AlexNet in 2012, which demonstrated that deep convolutional neural networks (CNNs) trained on graphics processing units could substantially outperform hand-engineered features on the ImageNet benchmark. Subsequent architectures pushed accuracy higher. VGG [2] explored very deep networks composed of small convolutional filters; ResNet [3] introduced residual skip connections that enabled stable training of networks hundreds of layers deep; and the Inception family [4] used parallel multi-scale filters within a single block to capture features at multiple receptive-field sizes.

These models achieve high accuracy but at significant computational cost, which limits their suitability for real-time or mobile applications. A standard convolutional layer that maps an input of DF × DF × M to an output of DF × DF × N using DK × DK kernels incur a cost of DK · DK · M · N · DF ·

DF multiply–accumulate operations, which grows quadratically

with the number of channels. To address this, Howard et al. [5] introduced MobileNet, which factorizes the standard convolution into a depth wise convolution that filters each input channel independently followed by a 1 × 1 pointwise convolution that combines the filtered outputs. This factorization reduces the cost to DK · DK · M · DF · DF + M · N · DF · DF , a reduction by a factor of approximately 1/N + 1/D2, which for typical 3 × 3 kernels yields an eight- to nine-fold saving with only a small

accuracy penalty.

MobileNetV2 [6] further refined this design with two key ideas. First, inverted residual blocks expand a low-dimensional input to a higher dimension with a pointwise convolution, apply a lightweight depth wise convolution, and then project back to a low-dimensional output, with a residual shortcut connecting the narrow bottleneck tensors. Second, linear bottlenecks omit the nonlinearity on the final projection, because applying a ReLU in a low-dimensional space was shown empirically to destroy information. The resulting network has roughly 3.5 million parameters, an order of magnitude fewer than VGG or ResNet-50, while remaining competitive on ImageNet. Its favorable accuracy-to-cost ratio makes it well suited to edge deployment, and it is the architecture used as the feature extractor in this work.

## 3. TRANSFER LEARNING

Training a deep CNN from scratch typically requires millions of labeled images and many hours of GPU time, resources that are unavailable for most narrowly scoped problems. Yosinski et al. [7] showed empirically that features learned on a large source dataset such as ImageNet transfer effectively to other visual tasks, particularly in the lower layers of the network, which capture generic edges, textures, and shapes.

The standard transfer-learning recipe is to freeze the pre-trained convolutional backbone, attach a small task-specific classification head, and optionally fine-tune the top layers of the backbone after the head has converged. This approach has become a default for small-data problems because it leverages generic learned features while adapting only the parameters most relevant to the target task. It is the strategy adopted in this project, where the masked/unmasked distinction can be learned from a few thousand examples by reusing ImageNet representations.

Two practical considerations govern effective fine-tuning. First, the classification head must be trained to convergence before any backbone layers are unfrozen; otherwise, the large, poorly conditioned gradients flowing from a randomly initialized head would corrupt the carefully learned pretrained weights. Second, fine-tuning should be performed with a markedly smaller learning rate than that used to train the head, so that the backbone weights are nudged rather than overwritten. Both principles are reflected in the two-phase schedule described in Section VII, which trains the head at a learning rate

of $1 \times 10{-}4$ and then fine-tunes the upper backbone layers at $1 \times 10{-}5$.

## 4. FACE DETECTION

Locating faces within an image predates mask detection by decades. The Viola–Jones detector [8] introduced Haar-feature cascades trained with AdaBoost over an integer image representation, achieving real-time frontal face detection on the hard-ware of its era; it remains popular for its speed, although it struggles with non-frontal and occluded faces a serious

limitation when much of the face is hidden by a mask. More accurate modern approaches include MTCNN [9], which uses a cascade of three CNNs, the proposal, refine, and output networks, for joint face detection and five-point landmark alignment, and single-shot detectors built on lightweight backbones such as a ResNet-10 network, as provided in the OpenCV deep-neural-network module. The latter trades a small amount of accuracy for the ability to process video frames at interactive rates on a CPU, which is attractive for deployment.

The choice of face detectors interacts with the downstream classifier: a detector that returns tight, well-aligned crops similar to the training distribution will yield higher classification accuracy than one that returns loose or skewed boxes. Although the present implementation bypasses the face-detection stage because the chosen dataset already provides pre-cropped faces this stage is essential for any system deployed on raw camera input and would precede the classifier in an end-to-end product. In such a deployment the detector and classifier together determine the achievable frame rate, so a lightweight pairing such as an SSD-style detector with the Mo-bileNetV2 classifier proposed here is a natural match.

## 5. FACE MASK DETECTION

A large body of mask-detection work appeared during 2020 and 2021, broadly divisible into two families. The first treats mask detection as a classification problem on pre-cropped faces, typically using a CNN feature extractor followed by a classifier. Loey et al. [10] exemplifies this family: they combined ResNet-50 features with classical machine-learning classifiers, including support vector machines, decision trees, and ensembles, and reported strong results across several small datasets. The second family treats mask detection as an object-detection problem, jointly localizing and classifying faces in a single network. Jiang et al. [11] released RetinaFaceMask, a one-stage detector with a feature-pyramid neck and a context-attention module trained specifically to localize and classify masked faces. Widely shared tutorial implementations, such as the PyImage-Search guide by Rosebrock [12], popularized the practical two-stage pipeline of detecting faces with an off-the-shelf detector and then classifying each crop as masked or unmasked with a MobileNet-based network.

The present work belongs to the first, classification-oriented family and adopts the established two-stage pattern, using Mo-bileNetV2 together with a fine-tuned head as the classifier component. Table 1 situates this work among representative prior systems. The aim is not to outperform these systems but to provide a clean, reproducible reference implementation whose every design choice is documented.

**Table 1.** Representative Face-Mask Detection Approaches

| Work | Backbone / Method | Characteristics |
|---|---|---|
| Loey *et al.* [10] | ResNet-50 + SVM/ensemble | Two-stage; classical classifier on deep features |
| Jiang *et al.* [11] | One-stage detector | Joint localization and classification |
| Rosebrock [12] | MobileNet + head | Tutorial two-stage pipeline |
| This work | MobileNetV2 + fine-tuned head | Two-stage; reproducible; ~99% test accuracy |

**Table 2.** Approximate Composition of the Dataset

| Partition | WithMask | WithoutMask | Total |
|---|---|---|---|
| Train | ~5000 | ~5000 | ~10,000 |
| Validation | ~400 | ~400 | ~800 |
| Test | ~480 | ~510 | ~990 |
| Total | | ~11,800 images | |

## 6. DATASET

Several public datasets emerged during the pandemic. This work uses the Face Mask ~12K Images dataset by Jangra [13], which provides a balanced and pre-split collection of approximately twelve thousand cropped faces, evenly divided between the ŸlithHask and ŸlithoutHask classes. The dataset is organized into Train, Validation, and Test partitions, whose approximate composition is summarized in Table 2; this fixed split simplifies reproducible experimentation and ensures that reported test metrics are computed on images never seen during training or model selection. Because the two classes are close to balanced, accuracy is a meaningful headline metric and is not inflated by a skewed prior, as it would be on an imbalanced dataset.

The images are cropped to the facial region and are generally well lit and close to frontal. Larger but noisier alternatives include MaskedFace-Net [14], which synthesizes masked variants of real face images by overlaying mask templates, and the Real-World Masked Face Dataset [15], which collects unconstrained images from the internet with substantial variation in pose, lighting, and resolution. The balanced, pre-cropped nature of the chosen dataset makes it well suited to isolating and evaluating the classification stage of the pipeline, at the cost of being less representative of unconstrained deployment conditions a trade-off revisited in Section IX.

## 7. PROPOSED METHODOLOGY

### A. Pipeline Overview

The proposed system follows the classification-oriented branch of the two-stage mask-detection pipeline. At inference time on raw imagery, a face detector would first produce cropped faces; in this study the dataset supplies those crops directly, so the work concentrates on the classification stage. Each crop is pre-processed, passed through the MobileNetV2 backbone to obtain a feature vector, and mapped by a small head to a two-way softmax over the masked and unmasked classes. The following subsections detail preprocessing, the network, the training schedule, the optimization callbacks, and the evaluation proto-col.

### B. Preprocessing and Data Augmentation

All images are resized to 224 × 224 pixels, matching the native input resolution of MobileNetV2, and pixel values are scaled using the standard MobileNetV2 preprocessing function, which maps inputs to the [−1, 1] range. To improve generalization, the training partition is augmented on the fly with random rotations of up to 20◦, zoom of up to 15%, width and height shifts of up to 10%, brightness variation in the range [0.8, 1.2], and horizontal flips, with empty regions filled using nearest-neighbor sampling. Each augmentation is chosen to mimic a realistic source of variation at deployment: rotation and shifts emulate imperfect face cropping and head tilt, zoom emulates varying camera distance, brightness jitter emulates changing illumination, and horizontal flipping reflects the left–right symmetry of faces. The validation and test partitions receive only the pre-processing step, with no augmentation, to provide an unbiased estimate of performance. Images are processed in mini batches of 32, and a fixed random seed of 42 is used throughout for reproducibility of the data shuffling, augmentation, and weight initialization.

### C. Model Architecture

The classifier uses MobileNetV2, pretrained on ImageNet and with its top classification layer removed, as a convolutional fea-ture extractor. The 7 × 7 × 1280 feature map produced by the backbone is reduced by a global average pooling layer, followed by a fully connected layer of 128 units with ReLU activation,

a dropout layer with a rate of 0.5 for regularization, and a final fully connected layer of two units with a softmax activation that produces the masked/unmasked probability distribution. Table 3 summarizes the classification head.

Two design choices in the head deserve comment. The global average pooling layer replaces the large, flattened feature map with a single 1280-dimensional vector, which drastically re-duces the parameter counts of the subsequent dense layer and acts as a structural regularize compared with a fully connected flattening. The dropout rate of 0.5 applied before the

output layer further combats overfitting, which is a real risk given that only a few thousand training images are used to adapt a network pretrained on more than a million ImageNet images. The head

adds on the order of 1.6 × 105 trainable parameters, a small frac-tion of the backbone, so the bulk of the model's representational

capacity comes from the transferred features.

### D. Two-Phase Training

Training proceeds in two phases. In the first phase the Mo-bileNetV2 backbone is frozen and only the newly added head is trained. The model is compiled with the Adam optimizer at a learning rate of 1 × 10−4 and a categorical cross-entropy loss and is trained for up to ten epochs. This allows the randomly initialized head to converge without corrupting the pretrained features through large early gradients.

In the second phase the backbone is partially unfrozen: the final thirty layers are made trainable while the remaining lower layers stay frozen. The model is recompiled with a reduced learning rate of 1 × 10−5 and trained for up to five additional epochs, allowing the high-level features to adapt to the mask-

detection task while preserving the generic low-level features.

**Table 3.** Classification Head Appended to MobileNetV2

| Layer | Output | Notes |
|---|---|---|
| MobileNetV2 (no top) | 7×7×1280 | ImageNet weights |
| GlobalAveragePooling2D | 1280 | spatial mean |
| Dense + ReLU | 128 | task-specific |
| Dropout | 128 | rate = 0.5 |
| Dense + Softmax | 2 | mask / no-mask |

### E. Optimization Callbacks

Three callbacks are used during both phases. Model checkpointing saves the weight that achieve the best validation accuracy. Early stopping halts training when validation accuracy fails to improve for five consecutive epochs and restores the best weights. A learning-rate scheduler reduces the learning rate by a factor of 0.5 when the validation loss plateaus for two epochs. Together these mechanisms mitigate overfitting and ensure that the final model corresponds to the best-performing checkpoint rather than the last epoch.

### F. Implementation and Reproducibility

The complete pipeline was implemented in Python using Ten-sorFlow/Keras for the model and training loop, OpenCV for im-age loading and color conversion, and scikit-learn for the classification report and confusion matrix. Data was loaded

through directory-based generators that infer class labels from the folder structure, apply the preprocessing and augmentation transformations on the fly, and yield batches in a memory-efficient streaming fashion, which avoids holding the full dataset in memory. The training and evaluation generators were configured without shuffling so that predicted labels could be aligned unambiguously with ground-truth labels for metric computation. The best model persisted to disk by the checkpoint callback and reloaded prior to final evaluation, ensuring that the reported numbers correspond exactly to the saved artifact that would be deployed. Fixing the random seed across the Python, NumPy, and TensorFlow generators makes the data ordering, augmentation draws, and weight initialization repeatable, so that an independent run of the same code reproduces equivalent results up to the nondeterminism inherent in GPU floating-point reductions.

## 8. EXPERIMENTAL RESULTS

### A. Experimental Setup

The model was implemented in TensorFlow/Keras and trained on a single GPU. Input images were 224 × 224 × 3 tensors; the optimizer was Adam; the loss was categorical cross-entropy; and the batch size was 32. The first training phase ran for up to ten epochs with the backbone frozen, and the second phase for up to five epochs with the final thirty backbone layers un-frozen, both governed by the checkpointing, early-stopping, and learning-rate-reduction callbacks described above. All random-ness was seeded for reproducibility.

### B. Quantitative Results

After the two training phases, the classifier reached approximately 99.9% validation accuracy and approximately 99% accuracy on the held-out test set. Precision and recall were near-perfect and well balanced across both classes, reflecting the balanced composition of the dataset and the discriminative power of the learned features. Table 4 summarizes the headlines of metrics.

**Table 4.** Summary of Classification Performance

| Metric | Value |
|---|---|
| Validation accuracy | ~99.9% |
| Test accuracy | ~99% |
| Precision (WithMask) | ~0.99 |
| Recall (WithMask) | ~0.99 |
| Precision (WithoutMask) | ~0.99 |
| Recall (WithoutMask) | ~0.99 |
| F1-score (macro avg.) | ~0.99 |

### C. Error Analysis and Qualitative Results

The confusion matrix computed over the test partition was strongly diagonal, with only a handful of off-diagonal entries; that is, both false positives (an unmasked face predicted as masked) and false negatives (a masked face predicted as un-

masked) were rare and roughly balanced, consistent with the symmetric precision and recall reported above. A qualitative check on ten randomly sampled test images typically yielded ten correct predictions, with the predicted-class confidence usually close to 100%. The rare errors occurred on ambiguous or partially occluded faces, for example, a hand or scarf near the mouth, an unusual head pose, or a mask whose color closely matched the skin tone—which indicates the regions where the classifier remains uncertain. The combination of high quantitative accuracy and consistent qualitative behavior confirms that a lightweight, mobile-oriented backbone with transfer learning is sufficient for accurate binary mask detection.

### D. Discussion

The two-phase schedule contributed materially to the final accuracy: freezing the backbone first let the head converge on stable features, and the subsequent low-learning-rate fine-tuning of the upper layers adapted the high-level representations to mask-specific cues without destroying the transferred low-level filters. Dropout and the augmentation pipeline kept the gap between training and validation accuracy small, indicating that overfit ting was controlled despite the modest size of the training set relative to the capacity of the backbone. It is worth noting that the very high accuracy partly reflects the curated nature of the dataset, whose images are cleanly cropped and well lit. Performance on unconstrained, real-world imagery with varied lighting, pose, and partial occlusion would be expected to be lower, motivating the limitations and future-work directions described below.

## 9. LIMITATIONS

Several limitations qualify the reported results. First, the evaluation is confined to a single curated dataset of pre-cropped, mostly frontal, well-lit faces; the headline accuracy should therefore be read as an upper bound on what the classifier would achieve in the field rather than as a deployment estimate. Second, the system is a binary classifier and does not recognize incorrectly worn masks, such as a mask covering only the mouth, which is an operationally important category for compliance monitoring. Third, the pipeline assumes that a separate face-detection stage supplies clean crops; detector errors and mis-aligned boxes would propagate to the classifier and were not measured here. Fourth, no formal latency or throughput benchmark was conducted, so the real-time claim rests on the known efficiency of MobileNetV2 rather than on measured frame rates on target hardware. Finally, the dataset does not document the demographic composition of its subjects, so potential disparities in accuracy across skin tones, ages, or mask types remain uncharacterized and would need to be audited before any real-world use.

## 10. CONCLUSION AND FUTURE WORK

This paper presented a lightweight face-mask classifier based on transfer learning with MobileNetV2. Using a frozen ImageNet backbone, a compact classification head, two-phase training with fine-tuning, and standard data augmentation, the model achieved approximately 99% test accuracy on a balanced public dataset of roughly twelve thousand cropped faces. The techniques used—MobileNetV2 as a feature extractor, transfer learning from ImageNet weights, and a lightweight binary classification head—are standard, well-validated choices for a low-compute mask classifier, and the contribution of this work is a clean and reproducible end-to-end implementation.

Several directions remain for future work. Integrating a face-detection front end, such as an SSD-based detector or MTCNN, would extend the classifier into a complete system capable of processing raw camera frames, and an end-to-end latency benchmark on representative edge hardware would substantiate the real-time claim with measured throughput. Evaluating the model on noisier, unconstrained datasets such as the Real-World Masked Face Dataset would provide a more realistic estimate of deployment performance and expose failure modes hidden by the curated benchmark. Extending the binary formulation to a three-way distinction among correctly masked, incorrectly masked, and unmasked faces would increase the practical value of the system in real compliance-monitoring scenarios. Finally, a fairness audit across skin tones, ages, and mask types, together with model-compression techniques such as quantization or pruning, would help ensure that the deployed system is both equitable and efficient on resource-constrained devices.

## References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classi-fication with deep convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2012.

[2] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv:1409.1556, 2014.

[3] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 770–778.

[4] C. Szegedy et al., “Going deeper with convolutions,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2015, pp. 1–
9.

[5] A. G. Howard et al., “MobileNets: Efficient convolutional neu-ral networks for mobile vision applications,” arXiv:1704.04861, 2017.

[6] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2018,
pp. 4510–4520.

[7] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transfer-able are features in deep neural networks?” in Proc.

Adv. Neural Inf. Process. Syst. (NeurIPS), 2014.

[8] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2001.

[9] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint face detec-tion and alignment using multitask cascaded convolutional net-works,” IEEE Signal Process. Lett., vol. 23, no. 10, pp. 1499–1503, 2016.

[10] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, “A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic,” Measurement, vol. 167, art. 108288, 2021.

[11] M. Jiang, X. Fan, and H. Yan, “RetinaFaceMask: A face mask detector,” arXiv:2005.03950, 2020.

[12] A. Rosebrock, “COVID-19: Face mask detector with OpenCV, Keras/TensorFlow, and deep learning,” PyImageSearch, 2020. [Online]. Available: https: //pyinagesearch. con/

[13] A. Jangra, “Face Mask ∼12K images dataset,” Kaggle, 2020. [Online]. Available: https: //vvv. kaggle. con/datasets/ ashishj angra27/face- nask- 12k- inages- dataset

[14] A. Cabani, K. Hammoudi, H. Benhabiles, and M. Melkemi, “MaskedFace-Net—A dataset of correctly/incorrectly masked face images in the context of COVID-19,” Smart Health, vol. 19, art. 100144, 2021.

[15] Z. Wang et al., “Masked face recognition dataset and applica-tion,” arXiv:2003.09093, 2020.