Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A TinyML Model for Real-Time Mask Detection on Embedded Machine Learning Devices

Ismail Lamaakal
Mohamed I University
Yassine Maleh
yassine.maleh@ieee.org

Mohamed I University
Khalid El Makkaoui
Mohamed I University
Ibrahim Ouahbi
Mohamed I University

Research Article

Keywords: TinyML, Mask Detection, MobileNet, Depthwise Separable Convolutions, Transfer Learning, Healthcare 4.0, Internet of Things (IoT)

Posted Date: April 29th, 2025

DOI: https://doi.org/10.21203/rs.3.rs-6533848/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

A TinyML Model for Real-Time Mask Detection on Embedded Machine Learning Devices

Ismail Lamaakal[1], Yassine Maleh[2*], Khalid El Makkaoui[1] and Ibrahim Ouahbi[1]

[1]Multidisciplinary Faculty of Nador, Mohammed Premier University, Oujda 60000, Morocco.
[2]Laboratory LaSTI, ENSAK, Sultan Moulay Slimane University, Khouribga 54000, Morocco.

*Corresponding author(s). E-mail(s): yassine.maleh@ieee.org;
Contributing authors: ismail.lamaakal@ieee.org;
khalid.elmakkaoui@ieee.org; i.ouahbi@ump.ac.ma;

Abstract

Mask detection has emerged as an essential technology for ensuring safety and compliance in both public and private settings. This paper presents an end-to-end application utilizing Vision Wake Words (VWW) within the realm of Tiny Machine Learning (TinyML) to detect mask adherence in real-time. Capitalizing on the efficient MobileNetV2 architecture with Depthwise Separable Convolutions, our model minimizes memory footprint and computational demands on Microcontroller Units (MCUs). We employ Transfer Learning to expedite the adaptation to mask detection tasks, significantly reducing training time and data requirements. Sensor characteristics and data preprocessing are carefully considered to manage the high data volume inherent to the ML workflow. Our original model, sized at 8.62MB, was effectively compressed to 214KB using TensorFlow Lite with Dynamic Range Quantization. This drastic reduction allows the deployment of our TinyML model on constrained devices, offering a highly scalable and cost-effective approach to enhance public health safety. Achieving a 99.6% accuracy rate in mask detection, our model underscores the precision of TinyML applications in public settings. The paper contributes to the objective of Healthcare 4.0 by enhancing

healthcare services with smart sensors and IoT integration, suggesting that deploying TinyML for mask detection provides a promising avenue for public health safety with minimal hardware constraints.

Keywords: TinyML, Mask Detection, MobileNet, Depthwise Separable Convolutions, Transfer Learning, Healthcare 4.0, Internet of Things (IoT)

1 Introduction

The global outbreak of COVID-19 has brought unprecedented challenges to public health systems around the world (Cullen et al., 2020; Kupcova et al., 2023; Kauhanen et al., 2023; Umbetkulova et al., 2024; Causadias et al., 2024), emphasizing the critical need for effective preventive measures. Among various interventions mask wearing has been widely recommended by health organizations as a primary deterrent against the spread of the virus (Himeur et al., 2023; Gupta et al., 2023; Xia et al., 2022; Liu et al., 2020). While the concept is straightforward, ensuring widespread compliance has proven to be a complex public health endeavor. The integration of technology into public health initiatives provides a promising avenue to monitor and enforce these preventive measures more effectively and in real-time. Within the technological solutions proposed, TinyML (Rajapakse et al., 2023; Lin et al., 2023; Abadade et al., 2023; Giordano et al., 2022; Lamaakal et al., 2024) emerges as a particularly valuable tool due to its suitability for deployment in resource-constrained environments.

TinyML allows for the development of smart, embedded systems capable of performing data processing tasks directly on MCUs (Wu et al., 2023) without the need for continuous cloud connectivity. This is crucial in public settings where constant data transmission is impractical or privacy concerns are significant. The application of TinyML in public health (Sun et al., 2023; Kim et al., 2023), specifically for real-time detection tasks like mask adherence, requires models that are efficient in both computational resource usage and power consumption. This efficiency must be achieved without sacrificing the accuracy or responsiveness of the system, as public health decisions depend on reliable and timely data.

Despite significant advancements in ML and its applications in healthcare (Kasula et al., 2024; Kolasa et al., 2024; Sarker et al., 2024), the use of TinyML for real-time public health monitoring under pandemic conditions is relatively unexplored (Habibzadeh et al., 2020; Subrahmannian et al., 2022; Bhuiyan et al., 2021; Islam et al., 2023; Haghi et al., 2020). Existing models often struggle with the dual challenge of operating within the low-power, low-memory constraints of MCUs while requiring minimal supervised training data to achieve high accuracy. There is a significant gap in the development of TinyML models that can be effectively implemented in everyday environments, specifically for tasks as critical as mask detection during a pandemic.

This paper introduces a cutting-edge solution that leverages Vision Wake Words (VWW) technology combined with the MobileNet architecture (Howard et al., 2017), known for its Depthwise Separable Convolutions, to create a highly efficient and accurate mask detection system. By utilizing transfer learning, we have significantly reduced the model's dependency on large training datasets, thereby decreasing the training time while maintaining high performance. Our approach successfully compresses the original 8.62MB model to a mere 214KB using TensorFlow Lite with Quantization, making it feasible for deployment on the most constrained MCUs. The final model demonstrates a 99.6% accuracy rate in detecting mask usage, offering a robust tool for enhancing public health safety. These advancements contribute significantly to Healthcare 4.0, integrating smart sensors and IoT to improve healthcare service delivery and compliance with health guidelines.

This paper presents several significant advancements in the application of machine learning to public health crises. The key contributions include:

1. **Innovative Application of TinyML:** Introduction of a novel application of TinyML for real-time mask detection, integrating advanced machine learning algorithms with ultra-low-power embedded systems for scalable public health solutions.
2. **Enhancing Public Health Compliance:** Presentation of a technology-driven approach to enforce mask compliance efficiently during health crises, establishing a framework for real-time, non-invasive monitoring of public health measures to potentially increase compliance.
3. **Resource-Efficient Technology for Health Surveillance:** Addressing the challenge of resource scarcity by integrating the MobileNet architecture with TinyML, which provides a pathway for deploying advanced health surveillance technologies in settings where computational resources are limited.
4. **High Accuracy in Mask Detection with Low Resource Use:** Detailing the development of a high-accuracy, TinyML-based mask detection system that operates efficiently on low-power devices, suitable for widespread implementation in critical infrastructures and public spaces.

The structure of our paper is designed to provide a comprehensive overview of our research and findings. In Section II, we delve into the relevant literature, presenting a review of related work that sets the context for our novel contributions. In Section III, we present an in-depth examination of the MobileNetV2 architecture, focusing on its efficiency for mask detection. We discuss the architectural details of MobileNetV2 and highlight the modifications made to optimize its performance for our specific task. This is followed by a detailed description of our methodology in Section IV, where we outline the processes involved in preparing our dataset, data preprocessing techniques, modifications made to the MobileNetV2 architecture, the methods used for model conversion to TensorFlow Lite, and the strategies for deploying the model on constrained devices. Section V is devoted to a rigorous analysis of

our findings and results. Here, we discuss the strengths and limitations of our approach, providing insights into the accuracy and efficiency of the mask detection model. We also explore the practical implications of our research in Section VI, discussing the feasibility of implementing our model in real-world scenarios with considerations of cost-effectiveness. The paper concludes with Section VII, where we summarize our key conclusions and discuss potential avenues for future work, aiming to further enhance the scalability and impact of TinyML in public health initiatives.

2 Related Work

Deep learning models have been shown to outperform shallow models in complex object identification tasks (Yadav, 2020). One example is developing a real-time system or model capable to Detect whether or not people have used masks in public places.

The authors of (Hussain et al., 2020) employed real-time deep learning to classify and distinguish emotions, whereas VGG-16 was used to categorize seven faces. This method is particularly effective during the Covid-19 lock-down period to limit the spread of cases. Ejaz et al. (Ejaz et al., 2019) employed main component analysis to distinguish between people with masked and unmasked faces.

The authors of (Li S. et al., 2020) used CNN to track and enforce compliance with health guidelines by identifying mask wearers. The HGL approach classifies head postures using masks for faces by analyzing color texture in pictures and line portraits. Front accuracy was 93.64%, with side-to-side accuracy at 87.17%.

The authors of (Qin & Li, 2020) developed a face mask recognition design utilizing the condition identification approach. The research divided the task into four steps: preprocessing the image, clipping face areas, super-resolution operation, and estimating the final state. The study's main novelty was the use of super-resolution to enhance low-quality picture performance. The author's suggested technique, which utilized SRCNet, achieved 98.7% accuracy in detecting face masks and their positions.

The authors of (Din et al., 2020) developed a GAN-based system for detecting facemasks, synthesising missing facial components, and reconstructing areas. The suggested GAN used two discriminators: the first analyzed the structure of the face mask and the second extracted the region hidden by the mask. They trained the model using two fake datasets.

In (Li C. et al., 2020), the authors use the Darknet-53 (YOLOv3) algorithm for facial detection.

The authors of (Chen et al., 2021) proposed a mobile phone-based detection technique. Three components were derived from GLCMs of face mask microimages. A three-result recognition analysis utilizing the KNN algorithm showed an overall accuracy of 82.87%. To distinguish face masks in microphotos, the

system employed a grey level co-occurrence matrix. However, the approach is confined to smartphones and may not be appropriate for all applications.

The authors of (Venkateswarlu et al., 2020) recommended employing a pre-trained MobileNet with the global pooling block for facial recognition and detection. The pre-configured MobileNet creates a multidimensional component map from a shaded image. The suggested model's usage of an overall pooling block prevents over-fitting. The authors of (Loey et al., 2021) demonstrates an innovative hybrid model combining deep transfer learning with machine learning to efficiently detect face masks during the COVID-19 pandemic. This approach enhances accuracy and adaptability in real-world applications, offering significant improvements over traditional deep learning methods.

The authors of (Boutros et al., 2022) present a novel approach to enhance facial recognition technology under masked conditions, a pertinent topic due to the COVID-19 pandemic. Utilizing advanced deep learning algorithms, the paper explores the integration of innovative neural network architectures to improve the robustness and accuracy of mask detection systems.

The study by (Deng et al., 2021) introduces the "Self-restrained Triplet Loss" method to enhance the accuracy of masked face recognition, an essential adaptation due to the widespread use of face masks during the COVID-19 pandemic. This method innovatively modifies the conventional triplet loss approach, optimizing it to better handle the challenges posed by partially obscured facial features. The authors of (Zhong et al., 2024) tackle the challenges in Masked Face Recognition (MFR) due to mask occlusion by proposing a new method, Masked Face Data Uncertainty Learning (MaskDUF). This method adjusts optimization weights based on sample uncertainty and recognizability, achieving better class distinction and robustness. Their approach demonstrates notable improvements in accuracy compared to existing models.

The study (Faruque et al., 2021) introduces a lightweight CNN model to tackle the challenge of recognizing masked faces during the COVID-19 pandemic, using the HSTU Masked Face Dataset. Enhanced with batch normalization, dropout, and depth-wise normalization, this model surpasses traditional models like VGG16 and MobileNet with a 97% recognition accuracy.

3 MobileNetV2 Architecture: An Efficiency Overview

The MobileNetV2 network, which has been pre-loaded using the ImageNet database (ImageNet, 2014), serves as the basis for this modified model. It is capable of categorizing images into various object classes. In this adaptation, MobileNetV2 is implemented as a transfer learning model, while a Caffe model is employed for single shot detection (SSD) tasks like identification and verification. MobileNetV2, an advancement over its predecessor MobileNetV1, incorporates inverted residual blocks with linear bottlenecks and boasts a reduced number of parameters, enhancing its efficiency. The

architecture supports input sizes larger than 32x32, and performance improves with larger image dimensions. Figure 1 depicts the architectural schematic of MobileNetV2, resembling a CNN-based deep learning framework that includes layers for convolution, pooling, dropout, non-linear functions, full connectivity, and linear bottlenecks (Bhatt et al., 2021). The structure features a 1x1 convolution layer, seventeen 3x3 convolutional layers, a maximum pooling average layer, and a classification layer. For effective training and accurate mask detection, it is crucial to utilize substantial datasets. The primary objective is to enhance the accuracy of face mask detection while minimizing the number of trainable parameters. A key component of MobileNetV2 is the depth-wise separable convolution (DWSC) layer, which significantly speeds up the process. MobileNets significantly minimize the scale of neural networks used in computer vision. Let's explore further into their main innovative feature: Depthwise Separable Convolutions.

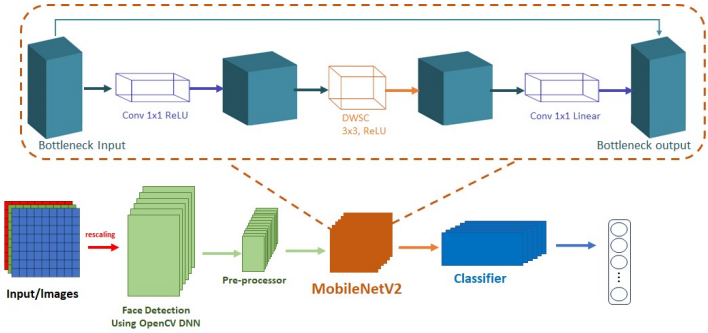


Fig. 1: MobileNetV2 Architecture.

To grasp the distinctions between depthwise separable convolutions and standard convolutions, we should first revisit and analyze standard convolutions. Specifically, we'll focus on calculating the number of multiplication operations and parameters involved in a standard convolutional operation.

Typically, our focus has been on filters used on grayscale images, such as the vertical line filter as illustrated in Figure 2

These images are also known as single-channel images because they contain only one set of pixel values. In contrast, most color images are composed of three channels—RGB—each representing the intensity levels of red, green, and blue. Consequently, the filters (often called kernels) used are more complex than a mere matrix; they are tensors that must operate across all three channels simultaneously. Below, we demonstrate this tensor operation. In the provided example, a 3x3x3 kernel convolves with a 9x9x3 image, resulting in a 7x7x1 output. The highlighted squares indicate the inputs and outputs from the final convolution operation (see Figure 3).

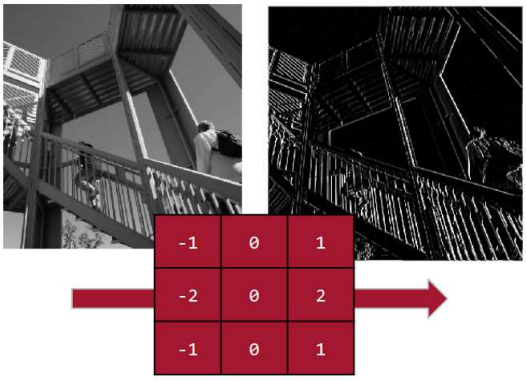


Fig. 2: Application of a Vertical Line Filter on a Grayscale Image.

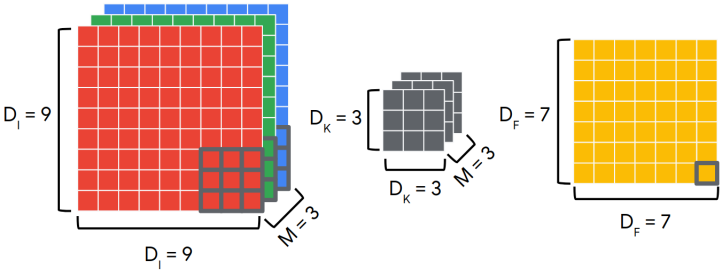


Fig. 3: Convolution Operation: 3D Kernel Applied to an RGB Image.

Typically, the convolution process necessitates $D_K \times D_K \times M$ multiplicative operations, considering the kernel's dimensions (D_K) and the count of image channels (M). To generate a singular output point, these calculations are reiterated $D_F \times D_F$ times over the output feature map. In standard practice, a set of N kernels, collectively termed as a filter, is utilized instead of a solitary kernel. A filter comprises various kernels concatenated together, with each kernel tailored to process a specific channel of the input data. Hence, the aggregate of multiplication operations required is encapsulated by the formula:

$$D_K * D_K * M * D_F * D_F * N = D_K^2 * M * D_F^2 * N \tag{1}$$

3.0.1 Depthwise Separable Convolutions

Depthwise Separable Convolutions unfold through a two-stage procedure. Initially, individual channels are processed independently, akin to solitary grayscale images, with the application of respective filters generating multiple intermediary outputs—this stage is termed as Depthwise Convolution. Subsequently, a Pointwise Convolution is executed on these intermediary outputs, utilizing a 1x1xC filter to derive the conclusive output. Illustrated below is

an example of this method where a 9x9x3 image undergoes convolution with three distinct 3x3 filters on a depthwise basis, yielding a 7x7x3 intermediary. Following this, a 1x1x3 pointwise filter is employed to generate the final 7x7x1 output as shown Figure 4

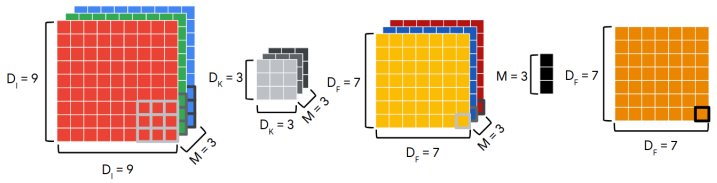


Fig. 4: Depthwise and Pointwise Convolutional Process on a 9x9x3 Image.

Each depthwise convolution demands M filters carrying out $D_K \times D_K$ multiplications. For generating a single output feature map, these operations are conducted $D_F \times D_F$ times. This results in a cumulative count of multiplications for the depthwise phase being $D_K \times D_K \times M \times D_F \times D_F$.

For pointwise convolutions, we apply a $1 \times 1 \times M$ filter over the feature map's depth $D_F \times D_F$ times. Contrary to conventional convolutions that utilize just one filter, here multiple filters are sequentially applied. In the scenario of depthwise separable convolutions, these numerous filters are utilized during the pointwise phase. Thus, if there are N pointwise filters, then $M \times D_F \times D_F \times N$ multiplications are required at this stage.

Adding up the multiplications needed for both stages, the overall required number of operations is computed as:

$$D_K \times D_K \times M \times D_F \times D_F + M \times D_F \times D_F \times N = M \times D_F^2 \times (D_K^2 + N) \quad (2)$$

Comparing the two types of convolutional computations, we determine their efficiency by analyzing the ratio of the required multiplications for each. By assigning standard convolutions to the denominator of our ratio, we derive:

$$\frac{\text{Depthwise Separable}}{\text{Standard}} = \frac{M \times D_F^2 \times (D_K^2 + N)}{D_K^2 \times M \times D_F^2 \times N} \quad (3)$$

This can be simplified to:

$$\frac{\text{Depthwise Separable}}{\text{Standard}} = \frac{D_K^2 + N}{D_K^2 \times N} \quad (4)$$

And can be succinctly represented as:

$$\frac{\text{Depthwise Separable}}{\text{Standard}} = \frac{1}{N} + \frac{1}{D_K^2} \quad (5)$$

This infers that utilizing more filters and larger kernels equates to a reduction in multiplication operations. For instance, with $D_K = 3$ and a conservative

estimate of $N = 10$ filters, the ratio becomes approximately 0.2111, illustrating that Depthwise Separable Convolutions result in a saving of nearly fivefold in multiplication operations, thus markedly improving efficiency and potentially lessening latency.

Observe that in standard convolution operations, we possess $D_K^2 \times M \times N$ learnable parameters within our assortment of filters/kernels. In contrast, Depthwise Separable convolution has a parameter quantity of $D_K^2 \times M + M \times N$. Considering the ratio of the two types:

$$\frac{\text{Depthwise Separable}}{\text{Standard}} = \frac{1}{N} + \frac{1}{D_K^2} \tag{6}$$

This highlights that Depthwise Separable convolution necessitates considerably less memory, as it has far fewer parameters to store.

Nonetheless, this efficiency comes with a trade-off. By curtailing the number of parameters to enhance latency and memory requirements, our models inherently sacrifice some degree of expressiveness. This limitation is usually acceptable for TinyML applications but should be taken into account when employing Depthwise Separable Convolutions more broadly.

4 Methodology

In this section of our research, we outline the systematic approach employed to engineer a TinyML model, designed to facilitate facial recognition tasks, including the critical ability to discern whether individuals are wearing masks. Figure 5 illustrates the transition from initial data acquisition through to the ultimate deployment of the model, ensuring a detailed walkthrough of data pre-processing, division into training and testing datasets, model training with rigorous validation, model conversion and subsequent deployment. This section is crafted to impart a transparent and detailed framework of our experimental strategy, underscoring the procedural nuances that fortify the reliability and performance of our TinyML-based facial recognition system, tailored for low-resource environments inherent to our study's focus.

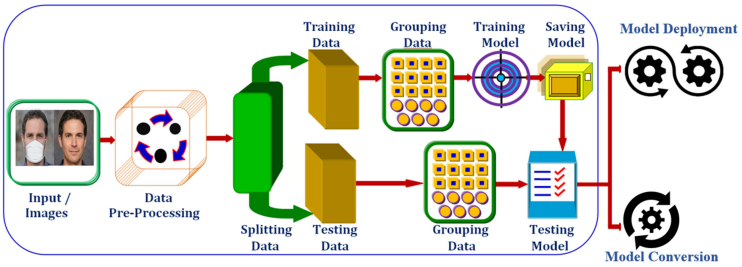


Fig. 5: Schematic Overview of Methodology

4.1 Data Description

In our research, we have utilized the Face Mask Lite Dataset (Kottarathil, 2020) created by Prasoon Kottarathil. This dataset has been instrumental in advancing our studies on face mask detection technologies, providing a robust foundation of 20,000 high-definition images, evenly split between individuals wearing masks and those without. Each image in this dataset was generated using the sophisticated StyleGAN-2 (Bermano et al., 2022), ensuring the high quality and consistency needed for effective training and evaluation of machine learning models in public health applications. The figure presents a selection of examples from the dataset, showcasing images of individuals with and without masks.

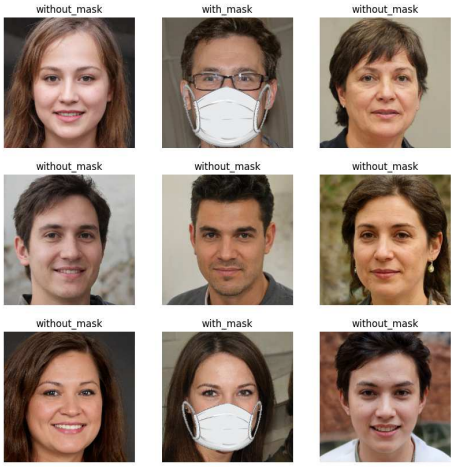


Fig. 6: Dataset Samples: Masked and Unmasked Images

4.2 Data Preprocessing

In this section of our research, we focus on the preprocessing phase, a crucial step to prepare the dataset for the subsequent machine learning tasks. This phase encompasses several key procedures including data augmentation, image resizing, normalization, and ultimately, data splitting into training and test sets.

4.2.1 Data Augmentation

In our research, we employ data augmentation (Shorten & Khoshgoftaar, 2019) to enhance the diversity of our image dataset, applying various transformations to train our model under different conditions. Here's a detailed explanation of common data augmentation techniques we use :

- **Rotation**: We rotate the images by a certain angle θ, using the rotation transformation matrix M:

$$M = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  This matrix is crucial for computing new pixel positions, ensuring images remain within the frame post-rotation.

- **Translation**: We shift images in the x and y directions by t_x and t_y pixels respectively, employing the transformation matrix T:

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

  This matrix moves each pixel to a new location, effectively shifting the image.

- **Scaling**: We change the image size by a scale factor s, using the scaling matrix S:

$$S = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  This uniformly scales the image dimensions, either enlarging or shrinking it.

- **Shearing**: We apply a shearing transformation, particularly in the x-direction, which can be represented by:

$$Sh_x = \begin{bmatrix} 1 & \tan(\phi) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  where ϕ is the shearing angle, tilting the image to create a "windblown" effect.

- **Flip**: Flipping images horizontally or vertically involves mirroring the image across its vertical or horizontal axis, respectively.

- **Color Augmentation**: We adjust the image colors, including brightness, contrast, saturation, and hue alterations. These are calculated by adjusting pixel values directly.

- **Normalization**: We normalize pixel values to a mean of zero and a standard deviation of one:

$$\text{Normalized Pixel Value} = \frac{\text{Pixel Value} - \mu}{\sigma} \tag{7}$$

where μ is the mean and σ is the standard deviation of pixel values.

By integrating these transformations randomly across different images in our dataset, we substantially improve our model's ability to generalize, crucial for robust performance on new, unseen data.
Following the data augmentation stage, our dataset expanded to include a total of 46,233 images.

4.2.2 image resizing

In this research endeavor, our focus extended to the meticulous resizing of images, ensuring their dimensions spanned across four distinct resolutions: 128x128, 160x160, 192x192, and 224x224. This deliberate variation was not arbitrary; rather, it served as a deliberate exploration into the impact of image resizing on the efficacy of our TinyML model. By subjecting the resized images to rigorous analysis, we sought to uncover nuanced insights into how alterations in resolution affect model performance across a spectrum of metrics. This comprehensive examination is poised to enrich our understanding of the interplay between image dimensions and model outcomes, thereby informing future advancements in TinyML deployment within the realm of image recognition.

4.2.3 Data Splitting

In this study, our approach to data splitting involves a meticulous partitioning of the dataset into three distinct subsets. We allocate 80% of the dataset for training, 10% for validation, and the remaining 10% for testing purposes. This strategic division ensures that our model is trained on a substantial portion of the data while reserving separate subsets for validation and testing. The training set serves as the foundation for model learning, enabling it to capture underlying patterns and relationships within the data. The validation set plays a critical role in fine-tuning model parameters and evaluating its performance during the training process. Finally, the testing set provides an unbiased assessment of the model's generalization capabilities on unseen data, ensuring robustness and reliability in real-world scenarios. Through this meticulous data splitting approach, we aim to develop a well-validated and dependable TinyML model for accurate image recognition tasks.

4.2.4 The proposed TinyML model

In developing our TinyML model for mask detection, we leverage the MobileNetV2 architecture as a feature extractor coupled with transfer learning techniques. This approach allows us to take advantage of a pre-trained

MobileNetV2 network and customize it for our specific task of classifying individuals as wearing masks or not.

Create the base model

We will construct the foundational model using Google's MobileNetV2, which has been pretrained on the vast ImageNet dataset containing 1.4 million images across 1000 classes. This extensive dataset, covering a broad spectrum of categories from jackfruit to syringe, provides a rich base for understanding and categorizing images. Such a diverse foundation will assist us in the specific task of discerning whether individuals in our dataset are wearing masks.

Initially, we must decide which layer of MobileNetV2 will serve as the repository of high-level features for our purpose. Our goal is to repurpose the model's output for a new task; hence, we focus on the final layer before the classification, the layer often referred to as the "bottleneck layer" in some literature. This layer is typically found just before the flattening process in many image processing models. Conceptually, in machine learning models, inputs are perceived at the bottom while outputs at the top, prompting us to omit the topmost classification layers. TensorFlow simplifies this process with a parameter, include_top=False, which allows us to initiate a MobileNetV2 model armed with pre-trained ImageNet weights and devoid of the classification layers at the top, making it perfectly suited for feature extraction.

After establishing our base model, we employ transfer learning (Iman et al., 2023; Zhuang et al., 2020), a technique that repurposes a model created for one task to adapt it for another, albeit related, task. By leveraging existing models, we not only conserve computational energy but also enhance the model's performance with limited data, which might otherwise lead to overfitting if the model were trained from scratch (see Figure 7). Thus, transfer learning offers three key advantages: it reduces energy consumption, accelerates model convergence, and increases the potential for higher accuracy. However, the implementation of transfer learning can be challenging and its success is not always assured.

Add a classification head

The conventional MobileNetV2 architecture was customized and refined with the integration of three additional layers. These enhancements comprised a 7x7 average pooling layer, a dropout layer with a rate of 0.2, and a dense_1 layer for the purpose of detection. Excluding these newly added layers, the rest of the architecture was employed for feature extraction. The detailed structure of each layer, including their output dimensions, shape, and the number of parameters, is described in Table 1 for more details. In our tailored MobileNetV2 model, there were 1281 parameters available for training, whereas 2257984 parameters were non-trainable, making up a total of 2259265

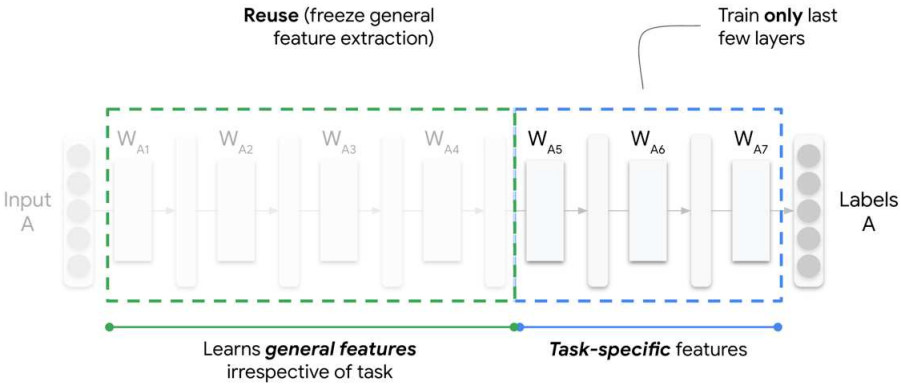


Fig. 7: The Pitfalls of Transfer Learning

parameters. In contrast, the original MobileNetV2 model was designed with 2257984 non-trainable parameters, without any parameters allocated for training, summing up to 2257984 parameters in total. The primary objective of our enhanced model was to minimize the quantity of trainable parameters relative to the original MobileNetV2. This is because it was unnecessary to train additional parameters when utilizing the pre-established layers for feature extraction, which is crucial in the identification of faces with or without masks.

Table 1: Architecture of the "model_2"

Layer (type)	Output Shape	Param #
input_6 (InputLayer)	(None, 224, 224, 3)	0
sequential_1 (Sequential)	(None, None, None, 3)	0
tf.math.truediv_2 (TFOpLambda)	(None, 224, 224, 3)	0
tf.math.subtract_2 (TFOpLambda)	(None, 224, 224, 3)	0
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2,257,984
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 1280)	0
dropout_2 (Dropout)	(None, 1280)	0
dense_1 (Dense)	(None, 1)	1281
Total params: 2,259,265 (8.62 MB)		
Trainable params: 1281 (5.00 KB)		
Non-trainable params: 2,257,984 (8.61 MB)		

4.3 Model Compression

This section focuses on the critical process of model conversion, specifically tailoring the enhanced MobileNetV2 for deployment on resource-constrained

devices typically encountered in TinyML applications. The transformation of the model into a TensorFlow Lite format is pivotal, facilitating its execution within the limited computational environments characteristic of edge devices. This conversion is achieved through two principal compression techniques: quantization to 8-bit integers and model pruning (Lamaakal et al., 2024; Gholami et al., 2022; Vadera et al., 2022; Lamaakal et al., 2024; Hu et al., 2023; Lamaakal et al., 2024). Each technique contributes uniquely to the reduction of the model's size and computational demand without significantly compromising performance, a balancing act of utmost importance in TinyML.

4.3.1 TensorFlow Lite Model Generation

In the context of deploying machine learning models on resource-constrained devices, TensorFlow Lite (Contoli et al., 2023) serves as a powerful tool that facilitates the conversion and optimization of TensorFlow models into a format that is more efficient for mobile and embedded devices. Below, we'll delve into two key techniques used in this process: Quantization and Pruning.

Dynamic Range Quantization Involves converting floating-point numbers to integers. The formula for dynamic range quantization is given by:

$$q = \text{round}\left(\frac{r}{S}\right) + Z \tag{8}$$

where:

- r is the real (floating-point) value.
- S is the scale factor.
- Z is the zero-point, representing the real number zero in the quantized space.
- q is the quantized integer value.

The reverse conversion from quantized value back to floating-point is:

$$r \approx S \cdot (q - Z) \tag{9}$$

The scale factor S is calculated as:

$$S = \frac{\text{Range of } r}{\text{Range of } q} \tag{10}$$

Pruning Reduces the model size by setting the less significant weights to zero. The pruning condition is defined as:

$$W'_{ij} = \begin{cases} 0 & \text{if W_\{ij\} } < \theta \\ W_{ij} & \text{otherwise} \end{cases} \tag{11}$$

Where:

- W is the original weight matrix.

- W' is the pruned weight matrix.
- θ is the pruning threshold.

The sparsity level of the pruned matrix is:

$$\text{Sparsity} = \frac{\text{Number of zero elements in } W'}{\text{Total number of elements in } W} \tag{12}$$

Both quantization and pruning are essential techniques for optimizing machine learning models for deployment on resource-constrained devices. Quantization effectively reduces the computational resources required by converting floating-point values to integers, which accelerates inference and reduces model size. Pruning increases model efficiency by eliminating weights that have minimal impact on the model's output, thereby enhancing computational speed and potentially improving model generalization. Together, these techniques enable the deployment of advanced machine learning models in environments where computational resources, memory, and power are limited.

4.3.2 Generate a TensorFlow Lite for Microcontrollers Model

After the initial transformation into TensorFlow Lite format, the model is further modified to work with the TensorFlow Lite Micro (TFLM) runtime (Manor & Greenberg, 2022). TFLM is tailored for use in microcontrollers and similar devices with very limited memory capacity. The adaptation involves translating the TFLite model into a C source file, which includes an array of bytes that represent the model. This conversion can be achieved using the '*xxd*' command-line utility or through a Python script provided by TensorFlow.

5 Experimental Analysis and Evaluation

This section delves into the comprehensive evaluation of our TinyML-based mask detection model. We outline the experimental setup, methodologies employed for testing, and a detailed analysis of the model's performance across various metrics. By systematically examining the results, we assess the efficacy of the MobileNetV2 architecture when adapted for real-time mask detection on constrained devices.

5.1 Experiment

The experimental setup, outlined in Table 2, was carefully designed to test the performance of the TinyML-based mask detection model within a realistic yet controlled environment. Equipped with a high-specification Dell PC and an array of sophisticated development tools including Python, TensorFlow, Keras, and OpenCV, this environment facilitated the development and testing of complex image processing algorithms. The OV7670 camera module (see Figure 8, used in conjunction with the Arduino Nano 33 BLE microcontroller, formed a robust image acquisition system essential for capturing real-time data.

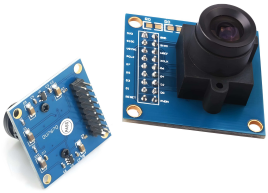


Fig. 8: The OV7670 Camera Overview.

The Arduino Nano 33 BLE was specifically utilized to deploy our TinyML model (see Figure 9). During the experiments, if the model detected that a

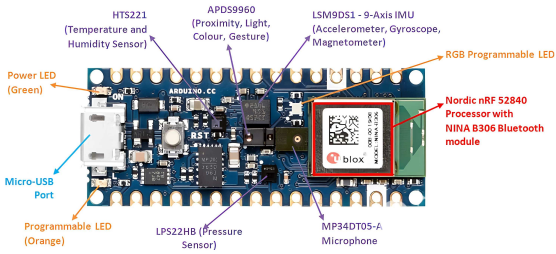


Fig. 9: Arduino Nano 33 BLE Sense Hardware Overview.

person was wearing a mask, the LED on the Arduino turned green, indicating compliance. Conversely, if no mask was detected, the LED turned red, signaling non-compliance. This setup not only enabled the effective deployment of our advanced TinyML model but also supported a thorough analysis of its performance metrics such as real-time execution capability, accuracy in mask detection, and operational efficiency on the resource-constrained Cortex-M4 processor. The experiment highlights the model's applicability in public health monitoring, particularly in scenarios that demand low power consumption while maintaining high operational performance.

5.2 Training Process

In our approach, the foundational layers of the pre-trained MobileNetV2 model were kept frozen to leverage the existing learned features, and a new set of trainable layers was introduced. These layers were specifically trained using our dataset to distinguish between individuals wearing masks and those not. This strategy allowed us to utilize the inherent strengths of the MobileNetV2 model, including its pre-tuned bias weights, without the need for additional computational resources and avoiding the problem of catastrophic forgetting. Subsequently, the model was further refined by introducing additional training parameters: a learning rate of 0.001, a training duration of 10 epochs, and a batch size of 32. For optimizing the training process, we employed the Adam

Table 2: Execution environment specifications.

Component	Specification
PC	Brand: Dell
	CPU: Intel(R) Core(TM) i5-8400H CPU @ 2.50GHz
	RAM: 8GB
	Storage: 500GB SSD
	OS: Windows 10
Development Tools	Python
	TensorFlow
	Keras
	Numpy
	Pandas
	Jupyter Notebook
	Visual Studio Code
	OpenCV
Camera	OV7670
Microcontroller	Arduino Nano 33 BLE
Processor	Cortex-M4 running at 64 MHz
Memory	Flash: 1 MB
	SRAM: 256 KB

optimizer (Kingma & Ba, 2014), renowned for its efficiency in various deep learning applications, particularly in computer vision. The Adam optimizer dynamically adjusts the learning rate for each weight in the neural network by estimating the gradients' first and second moments. This enabled effective transfer learning, allowing for precise adjustments to the neural network's weights with each batch processed. After completing the training, the neural network was saved, enabling us to make accurate predictions on new images from the dataset. This method ensured that our model was both robust and efficient, ready for deployment in real-world scenarios requiring mask detection.

5.3 Evaluation Metrics

In our research, we've used several key evaluation metrics that are essential for assessing the performance of predictive models.

5.3.1 Accuracy

We calculate the proportion of real outcomes (including true positives and true negatives) in the total number of instances studied as shown equation 13.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{13}$$

5.3.2 Precision

We assess the accuracy of positive predictions, focusing on the proportion of positive identifications that were actually correct (see equation 14).

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{14}$$

5.3.3 Recall (Sensitivity)

We measure the ability to find all the relevant cases within a dataset, reflecting the proportion of actual positives correctly identified (see equation 15).

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{15}$$

5.3.4 F1-Score

We use the harmonic mean of precision and recall to provide a single metric that balances both (see equation 16).

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{16}$$

5.3.5 Error Rate

We measure the proportion of predictions that were incorrect, highlighting the overall rate of failure in the predictions (see equation 17).

$$\text{Error Rate} = \frac{\text{Number of Incorrect Predictions}}{\text{Total Number of Predictions}} \tag{17}$$

5.4 Results and Discussion

In this section, we detail the performance evaluation of our proposed TinyML model designed for real-time mask detection. After training our TinyML model, we undertook a comprehensive evaluation that yielded encouraging outcomes. The model exhibited outstanding performance, achieving an accuracy of 99.6% in the training phase. Remarkably, it reached a perfect accuracy of 100% during the validation phase. These results are graphically represented in Figure 10.

Then , we employed confusion matrices to thoroughly examine our model's ability to accurately distinguish between images of individuals wearing masks and those not wearing them. These matrices offer a clear visual depiction of our model's classification precision. Figure 11 displays both the generalized confusion matrices and the specific confusion matrix for our proposed model's classifications. We evaluated the proposed Tiny MobileNetV2 model using several metrics, including accuracy, precision, recall (sensitivity), F1-score, and error rate. As illustrated in Figure 11, the model demonstrated exceptional

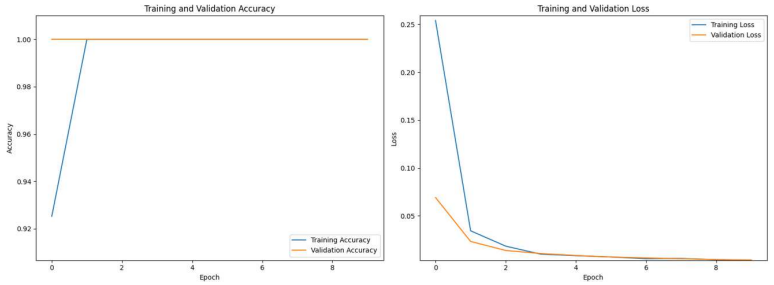
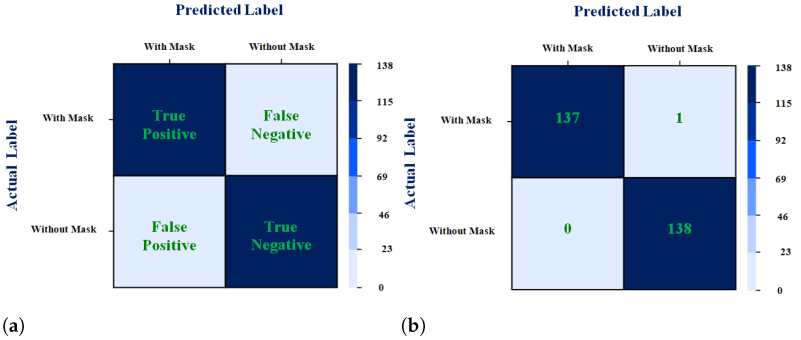


Fig. 10: (a) Accuracy curve for training and validation, (b) Loss curve for training and validation.

performance with a true positive (TP) count of 137, a true negative (TN) count of 138, no false positives (FP), and only one false negative (FN).



(a) (b)

Fig. 11: (a) Standard Form Confusion Matrix, (b) Confusion matrix of Tiny MobileNetV2 model.

In this study, we utilized an image dimension of 224x224 pixels for photo analysis, which significantly enhanced the model's performance metrics. To assess the impact of image dimensions on model effectiveness, we conducted a series of tests. The 224x224 dimension yielded the highest accuracy at 99.6%. Additionally, the specificity metric, which measures the model's ability to correctly identify negatives, also showed remarkable improvement at this dimension. The detailed performance metrics for various image dimensions, including accuracy, precision, recall, F1-score and error rate are compared in the table 3.

After evaluating all metrics for our Tiny MobileNetV2, we converted the model into a format suitable for deployment on resource-constrained devices. Tables 4 and 5 detail the size reductions achieved through the application of different optimization techniques. The original TensorFlow model size of 8,620,000 bytes was significantly reduced when converted to TensorFlow Lite, showing a 63.99% decrease in size. Further reductions were achieved with

Table 3: Comparison of different dimension images of our Tiny MobileNetV2 model

Sl.No.	Model Version (Dimension)	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Error Rate (%)
1	MobileNetV2 (128 x 128)	97.82	98.53	97.10	97.80	2.18
2	MobileNetV2 (160 x 160)	98.19	98.54	97.82	98.18	1.81
3	MobileNetV2 (192 x 192)	98.55	98.55	98.55	98.55	1.45
4	MobileNetV2 (224 x 224)	99.6	100	99.2	99.64	0.4

TensorFlow Lite Quantized, which demonstrated a dramatic size reduction of 97.52%. Similarly, the use of pruning techniques also resulted in substantial size reductions. Conversion to TensorFlow Lite with pruning yielded a model that is 69.33% smaller than the original TensorFlow Lite model, which itself was 40.79% smaller than the original TensorFlow version. We can conclude that the quantization method achieves a higher degree of size reduction compared to the pruning technique, making it especially effective for deployment in environments with severe size constraints.

Table 4: Comparison of model sizes and their respective reductions in size after conversion using quantization.

Model	Size (bytes)	Size Reduction (%)
TensorFlow	8620000	-
TensorFlow Lite	3103632	63.99
TensorFlow Lite Quantized	214120	97.52

Table 5: Comparison of model sizes and their respective reductions in size after conversion using pruning technique.

Model	Size (bytes)	Size Reduction (%)
TensorFlow	8620000	-
TensorFlow Lite	5103632	40.79
TensorFlow Lite with pruning	2644120	69.33

The conversion techniques applied to the Tiny MobileNetV2 model, quantization and pruning, have distinct impacts on testing accuracy. The table 6 reveals that the model maintains its original testing accuracy of 99.8% across all versions post-quantization, including the TensorFlow original, TensorFlow Lite, and TensorFlow Lite with Quantization. This consistency underscores the effectiveness of quantization in reducing model size without sacrificing accuracy. In contrast, the table 7 shows a different outcome when pruning techniques are applied. While the original TensorFlow model retains an accuracy of 99.8%, the TensorFlow Lite version experiences a noticeable drop to

Table 6: Testing the accuracy of models post-conversion using quantization.

Model	Testing Accuracy
TensorFlow	99.8%
TensorFlow Lite	99.8%
TensorFlow Lite with Quantization	99.8%

96.4%. Furthermore, the TensorFlow Lite with Pruning version sees a more significant decrease to 92.2%. This suggests that while pruning effectively reduces the model size, it does so at the cost of a marked reduction in testing accuracy. From these observations, we can conclude that quantization is a superior method for model size reduction when the preservation of accuracy is critical, especially in resource-constrained deployment scenarios. On the other hand, while pruning offers substantial size reduction, it requires careful consideration due to its impact on model accuracy.

Table 7: Testing the accuracy of models post-conversion using pruning.

Model	Testing Accuracy
TensorFlow	99.8%
TensorFlow Lite	96.4%
TensorFlow Lite with Quantization	92.2%

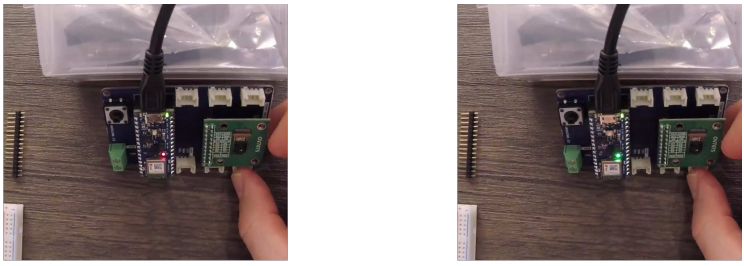
Figure 12 demonstrates that our model effectively and accurately recognizes images of individuals whether they are wearing masks or not.



Fig. 12: Evaluating Model Performance on the Test Dataset

After testing and verifying the satisfactory performance of our model post-conversion, we opted to deploy the quantized TensorFlow Lite model on the

Arduino Nano 33 BLE. As previously mentioned, the TensorFlow Lite Quantized model has a compact size of 214KB, which is compatible with the specifications of the Arduino Nano 33 BLE—featuring a Cortex-M4 processor running at 64 MHz, 1 MB of flash memory, and 256 KB of SRAM. Following deployment on this resource-constrained device, we configured the system such that if the camera detects a person wearing a mask, the LED turns red; conversely, if a person is not wearing a mask, the LED turns green, as demonstrated in Figure 13.



(a) (b)

Fig. 13: a) Arduino Nano 33 BLE Showing No Mask Detected with Red LED, b) Arduino Nano 33 BLE Indicating Mask Detected with Green LED

The findings from our proposed Tiny MobileNetV2 model, detailed in Table 8, underscore its remarkable performance in terms of accuracy while utilizing fewer parameters compared to various other models.

Our results unequivocally demonstrate that the proposed model outperforms existing counterparts.

5.4.1 Limitations of Our Proposed Model

While our study on real-time mask detection using TinyML showcases significant advancements, it also encounters certain limitations:

1. **Dataset Limitations**: The model performs exceptionally well with a high-quality, balanced dataset. However, real-world applications often present more diverse and challenging data, which might affect the model's performance. Exploring the model's effectiveness across varied datasets could enhance its robustness and applicability in less controlled environments.
2. **Hardware Constraints**: The study focuses on optimizing model size and computational demands to fit resource-constrained microcontrollers. However, the ongoing advancements in microcontroller capabilities could allow for more complex models without compromising performance, suggesting a need to continually update and optimize model architecture as hardware evolves.

Table 8: Comparison of various models' accuracy.

Sl. No.	Model Name	Accuracy (%)
1	VGG-Face (Chandra & Reddy, 2020)	68.17%
2	OpenFace (Chandra & Reddy, 2020)	63.18%
3	DeepFace (Chandra & Reddy, 2020)	63.78%
4	MTCNN+FaceNet (Hong et al., 2020)	64.23%
5	FaceNet (Chandra & Reddy, 2020)	67.48%
6	IAMGAN (Geng et al., 2020)	86.50%
7	DSA-Face (Wang Q & Guo G, 2021)	91.20%
8	SSDMNV2 (Nagrath et al., 2021)	92.64%
9	CBAM (Li Y. et al., 2021)	92.61%
10	CNNs (Hariri, 2022)	91.30%
11	LW-CNN (Farman et al., 2022)	98.47%
12	FaceMaskNet21 (Golwalkar & Mehendale, 2022)	88.92%
13	GANs (Li C. et al., 2020)	94.10%
14	MGL (Wang Z. et al., 2023)	95.00%
15	FaceNet (Anwar & Raychowdhury, 2020)	97.00%
16	LPD (Ding et al., 2020)	97.94%
17	CNN (Goyal et al., 2022)	98.00%
18	ResNet50 (Sethi et al., 2021)	98.20%
19	**Tiny MobileNetV2 (Proposed Model)**	**99.6%**

3. **Impact of Quantization and Pruning**: The use of quantization and pruning significantly reduces the model size but may also impact its accuracy and efficiency. A deeper exploration into the trade-offs between model compression and performance is necessary to ensure optimal deployment in real-time applications.

These points highlight the need for ongoing research to address these limitations and improve the technology's efficacy and scope of deployment.

6 Feasibility and Cost Analysis of Model Implementation

Implementing the most effective TinyML model for real-time mask detection involves assessing both technical feasibility and cost-effectiveness. This section explores these aspects, aiming to determine a practical approach for widespread deployment:

6.1 Model Efficiency and Hardware Costs

The optimized TinyML model, while compact, must be deployed on hardware capable of supporting real-time data processing (Sabovic et al., 2023). The use of microcontrollers like the Arduino Nano 33 BLE, which balances cost and computational capability, presents a cost-effective solution. Advances in hardware may further reduce costs, allowing for the deployment of more sophisticated models without significant financial burden.

6.2 Scalability

For large-scale implementation, such as in public transportation systems or retail environments, the scalability of the solution is crucial. The cost per unit decreases with the number of units produced and deployed, making the project more feasible as it scales.

6.3 Maintenance and Updates

Continuous operation in diverse environments may require periodic updates to the model and hardware maintenance. Setting up a framework for remote updating of the models and routine maintenance checks can ensure long-term sustainability with minimal additional cost.

6.4 Integration with Existing Systems

Integrating this technology with existing surveillance and monitoring systems can reduce implementation costs. By leveraging existing camera systems and infrastructure, the need for extensive new installations is minimized, enhancing the cost-effectiveness of the project.

6.5 Government and Institutional Support

Securing support from governmental or institutional bodies can significantly reduce financial constraints. Subsidies, grants, or public-private partnerships could provide the necessary funding to cover initial costs, making the deployment feasible in public interest areas.

The combination of technical efficiency, scalable deployment options, and potential institutional support ensures that implementing the optimized TinyML mask detection model can be both feasible and cost-effective, particularly in critical public health contexts.

7 Conclusion and Future Work

This work successfully demonstrates the utility of TinyML in deploying real-time mask detection systems that are efficient and effective, utilizing a highly compressed MobileNetV2 model to achieve an outstanding 99.6% accuracy in mask detection. This approach addresses the crucial need for continuous monitoring of public health safety measures, particularly in managing infectious diseases, and offers a scalable and cost-effective solution to enhance compliance with public health protocols. Our findings not only showcase the capability of TinyML models to operate under constrained resources but also underscore the potential of intelligent systems in public health surveillance. To expand upon this research, future work could focus on several key areas: investigating the model's performance across more diverse environments and datasets to improve its robustness and adaptability to different scenarios, such as varying light conditions and crowds; further testing in real-world settings such as

public transport and retail spaces to assess practical deployment challenges and gather user feedback for model refinement; exploring the use of newer and more capable microcontrollers to possibly enhance model performance and efficiency without significant cost increases; studying the integration of the mask detection model with other IoT devices and systems for comprehensive health monitoring solutions; and applying novel machine learning techniques like federated learning for data privacy and model training efficiency. These directions not only promise to enhance the efficacy of public health monitoring systems but also contribute to the broader application of TinyML in healthcare and other critical sectors.

Data Availability. Data sources are included in the manuscript.

Author Contributions. All authors wrote and reviewed the manuscript.

Declaration

Conflict of interest. There are no conflicts of interest to declare.

References

Abadade, Y., Temouden, A., Bamoumen, H., Benamar, N., Chtouki, Y., & Hafid, A. S. (2023). A comprehensive survey on TinyML. *IEEE Access*, 11, 96892–96922.

Anwar, A., & Raychowdhury, A. (2020). Masked face recognition for secure authentication. *arXiv preprint arXiv:2008.11104*.

Bermano, A. H., Gal, R., Alaluf, Y., Mokady, R., Nitzan, Y., Tov, O., et al. (2022). State-of-the-art in the architecture, methods, and applications of StyleGAN. *Computer Graphics Forum*, 41(2), 591–611.

Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., et al. (2021). CNN variants for computer vision: History, architecture, application, challenges and future scope. *Electronics*, 10(20), 2470.

Bhuiyan, M. N., Rahman, M. M., Billah, M. M., & Saha, D. (2021). Internet of Things (IoT): A review of its enabling technologies in healthcare applications, standards protocols, security, and market opportunities. *IEEE Internet of Things Journal*, 8(13), 10474–10498.

Boutros, F., Damer, N., Kirchbuchner, F., & Kuijper, A. (2022). Self-restrained triplet loss for accurate masked face recognition. *Pattern Recognition*, 124, 108473.

Causadias, J. M., & Neblett, E. W. Jr. (2024). Understanding the impact of the COVID-19 pandemic on the mental health of Latinx children, youth, and families: Clinical challenges and opportunities. *Journal of Clinical Child & Adolescent Psychology*, 53(1), 1–9.

Chandra, Y. B., & Reddy, G. K. (2020). A comparative analysis of face recognition models on masked faces. *International Journal of Scientific & Technology Research*, 9(10), 30175–0.

Chen, Y., Hu, M., Hua, C., Zhai, G., Zhang, J., Li, Q., & Yang, S. X. (2021). Face mask assistant: Detection of face mask service stage based on mobile phone. *IEEE Sensors Journal*, 21(9), 11084–11093.

Contoli, C., & Lattanzi, E. (2023). A study on the application of TensorFlow compression techniques to human activity recognition. *IEEE Access*, 11, 48046–48058.

Cullen, W., Gulati, G., & Kelly, B. D. (2020). Mental health in the COVID-19 pandemic. *QJM*, 113(5), 311–312.

Deng, H., Feng, Z., Qian, G., Lv, X., Li, H., & Li, G. (2021). MFCosface: A masked-face recognition algorithm based on large margin cosine loss. *Applied Sciences*, 11(16), 7310.

Ding, F., Peng, P., Huang, Y., Geng, M., & Tian, Y. (2020). Masked face recognition with latent part detection. In *Proceedings of the 28th ACM International Conference on Multimedia* (pp. 2281–2289).

Din, N. U., Javed, K., Bae, S., & Yi, J. (2020). A novel GAN-based network for unmasking of masked face. *IEEE Access*, 8, 44276–44287.

Ejaz, M. S., Islam, M. R., Sifatullah, M., & Sarker, A. (2019). Implementation of principal component analysis on masked and non-masked face recognition. In *Proceedings of the 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, 1–5.

Farman, H., Khan, T., Khan, Z., Habib, S., Islam, M., & Ammar, A. (2022). Real-time face mask detection to ensure COVID-19 precautionary measures in the developing countries. *Applied Sciences*, 12(8), 3879.

Faruque, M. O., Islam, M. R., & Islam, M. T. (2021). Advanced masked face recognition using robust and lightweight deep learning model. *International Journal of Computer Applications*, 975, 8887.

Geng, M., Peng, P., Huang, Y., & Tian, Y. (2020). Masked face recognition with generative data augmentation and domain constrained ranking. In *Proceedings of the 28th ACM International Conference on Multimedia* (pp. 2246–2254).

Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., & Keutzer, K. (2022). A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, 291–326.

Giordano, M., Piccinelli, L., & Magno, M. (2022). Survey and comparison of milliwatts microcontrollers for tiny machine learning at the edge. In *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*.

Golwalkar, R., & Mehendale, N. (2022). Masked-face recognition using deep metric learning and FaceMaskNet-21. *Applied Intelligence*, 52(11), 13268–13279.

Goyal, H., Sidana, K., Singh, C., Jain, A., & Jindal, S. (2022). A real-time face mask detection system using convolutional neural network. *Multimedia*

Tools and Applications, 81(11), 14999–15015.

Gupta, S., Sreenivasu, S. V. N., Chouhan, K., Shrivastava, A., Sahu, B., & Potdar, R. M. (2023). Novel face mask detection technique using machine learning to control COVID'19 pandemic. *Materials Today: Proceedings*, 80, 3714–3718.

Haghi, M., et al. (2020). A flexible and pervasive IoT-based healthcare platform for physiological and environmental parameters monitoring. *IEEE Internet of Things Journal*, 7(6), 5628–5647.

Habibzadeh, H., Dinesh, K., Rajabi Shishvan, O., Boggio-Dandry, A., Sharma, G., & Soyata, T. (2020). A survey of healthcare Internet of Things (HIoT): A clinical perspective. *IEEE Internet of Things Journal*, 7(1), 53–71.

Hariri, W. (2022). Efficient masked face recognition method during the COVID-19 pandemic. *Signal, Image and Video Processing*, 16(3), 605–612.

Himeur, Y., Al-Maadeed, S., Varlamis, I., Al-Maadeed, N., Abualsaud, K., & Mohamed, A. (2023). Face mask detection in smart cities using deep and transfer learning: Lessons learned from the COVID-19 pandemic. *Systems*, 11(2), 107.

Hong, Q., Wang, Z., He, Z., Wang, N., Tian, X., & Lu, T. (2020). Masked face recognition with identification association. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)* (pp. 731–735).

Howard, A. G., et al. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Hu, J., et al. (2023). A dynamic pruning method on multiple sparse structures in deep neural networks. *IEEE Access*, 11, 38448–38457.

Hussain, S. A., & Al Balushi, A. S. A. (2020). A real-time face emotion classification and recognition using deep learning model. *Journal of Physics: Conference Series*, 1432(1), 012087.

ImageNet. Available online: http://www.image-net.org (Accessed on 11 November 2024).

Iman, M., Arabnia, H. R., & Rasheed, K. (2023). A review of deep transfer learning and recent advancements. *Technologies*, 11(2), 40.

Islam, M. M., Nooruddin, S., Karray, F., & Muhammad, G. (2023). Internet of Things: Device capabilities, architectures, protocols, and smart applications in healthcare domain. *IEEE Internet of Things Journal*, 10(4), 3611–3641.

Kasula, B. Y. (2024). Optimizing healthcare delivery: Machine learning applications and innovations for enhanced patient outcomes. *International Journal of Creative Research in Computing*, 6(6), 1–7.

Kauhanen, L., Wan Mohd Yunus, W. M. A., Lempinen, L., Peltonen, K., Gyllenberg, D., Mishina, K., & Sourander, A. (2023). A systematic review of the mental health changes of children and young people before and during the COVID-19 pandemic. *European Child & Adolescent Psychiatry*, 32(6), 995–1013.

Kim, E., Kim, J., Park, J., Ko, H., & Kyung, Y. (2023). TinyML-based classification in an ECG monitoring embedded system. *Computers, Materials and Continua*, 75(1), 1751–1764.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kolasa, K., Admassu, B., Hołownia-Voloskova, M., Kędzior, K. J., Poirrier, J. E., & Perni, S. (2024). Systematic reviews of machine learning in healthcare: A literature review. *Expert Review of Pharmacoeconomics & Outcomes Research*, 24(1), 63–115.

Kottarathil, P. (2020). Face Mask Lite Dataset. *Kaggle Data*.

Kupcova, I., Danisovic, L., Klein, M., & Harsanyi, S. (2023). Effects of the COVID-19 pandemic on mental health, anxiety, and depression. *BMC Psychology*, 11(1), 108.

Lamaakal, I., Ouahbi, I., El Makkaoui, K., Maleh, Y., Pławiak, P., & Alblehai, F. (2024). A TinyDL model for gesture-based air handwriting Arabic numbers and simple Arabic letters recognition. *IEEE Access*.

Lamaakal, I., El Makkaoui, K., Ouahbi, I., & Maleh, Y. (2024). A TinyML model for gesture-based air handwriting Arabic numbers recognition. *Procedia Computer Science*, 236, 589–596.

Lamaakal, I., Maleh, Y., Ouahbi, I., El Makkaoui, K., & Abd El-Latif, A. A. (2024). A deep learning-powered TinyML model for gesture-based air handwriting simple Arabic letters recognition. In *Proceedings of the International Conference on Digital Technologies and Applications* (pp. 32–42). Springer Nature Switzerland.

Lamaakal, I., El Mourabit, N., El Makkaoui, K., Ouahbi, I., & Maleh, Y. (2024). Efficient gesture-based recognition of Tifinagh characters in air handwriting with a TinyDL model. In *Proceedings of the 2024 Sixth International Conference on Intelligent Computing in Data Sciences (ICDS)* (pp. 1–8). IEEE.

Li, C., Wang, R., Li, J., & Fei, L. (2020). Face detection based on YOLOv3. In *Recent Trends in Intelligent Computing, Communication and Devices: Proceedings of ICCD 2018*, 277–284.

Li, C., Ge, S., Zhang, D., & Li, J. (2020). Look through masks: Towards masked face recognition with de-occlusion distillation. In *Proceedings of the 28th ACM International Conference on Multimedia* (pp. 3016–3024).

Lin, J., Zhu, L., Chen, W.-M., Wang, W.-C., & Han, S. (2023). Tiny machine learning: Progress and futures [Feature]. *IEEE Circuits and Systems Magazine*, 23(3), 8–34.

Li, S., Ning, X., Yu, L., Zhang, L., Dong, X., Shi, Y., & He, W. (2020). Multi-angle head pose classification when wearing the mask for face recognition under the COVID-19 coronavirus epidemic. In *2020 International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS)*, 1–5.

Liu, Y., Zhang, X., Lin, Y., & Wang, H. (2020). Facial expression recognition via deep action units graph network based on psychological mechanism. *IEEE Transactions on Cognitive and Developmental Systems*, 12(2), 311–322.

Li, Y., Guo, K., Lu, Y., & Liu, L. (2021). Cropping and attention-based approach for masked face recognition. *Applied Intelligence*, 51, 3012–3025.

Loey, M., Manogaran, G., Taha, M. H. N., & Khalifa, N. E. M. (2021). A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic. *Measurement*, 167, 108288.

Manor, E., & Greenberg, S. (2022). Custom hardware inference accelerator for TensorFlow Lite for microcontrollers. *IEEE Access*, 10, 73484–73493.

Nagrath, P., Jain, R., Madan, A., Arora, R., Kataria, P., & Hemanth, J. (2021). SSDMNV2: A real-time DNN-based face mask detection system using single-shot multibox detector and MobileNetV2. *Sustainable Cities and Society*, 66, 102692.

Qin, B., & Li, D. (2020). Identifying facemask-wearing condition using image super-resolution with classification network to prevent COVID-19. *Sensors*, 20(18), 5236.

Rajapakse, V., Karunanayake, I., & Ahmed, N. (2023). Intelligence at the extreme edge: A survey on reformable TinyML. *ACM Computing Surveys*, 55(13s), 1–30.

Sabovic, A., Aernouts, M., Subotic, D., Fontaine, J., De Poorter, E., & Famaey, J. (2023). Towards energy-aware TinyML on battery-less IoT devices. *Internet of Things*, 22, 100736.

Sarker, M. (2024). Revolutionizing healthcare: The role of machine learning in the health sector. *Journal of Artificial Intelligence General Science*, 2(1), 35–48.

Sethi, S., Kathuria, M., & Kaushik, T. (2021). Face mask detection using deep learning: An approach to reduce risk of Coronavirus spread. *Journal of Biomedical Informatics*, 120, 103848.

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1), 1–48.

Subrahmannian, A., & Behera, S. K. (2022). Chipless RFID sensors for IoT-based healthcare applications: A review of state of the art. *IEEE Transactions on Instrumentation and Measurement*, 71, 1–20.

Sun, B., Bayes, S., Abotaleb, A. M., & Hassan, M. (2023). The case for tinyML in healthcare: CNNs for real-time on-edge blood pressure estimation. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, 629–638.

Umbetkulova, S., Kanderzhanova, A., Foster, F., Stolyarova, V., & Cobb-Zygadlo, D. (2024). Mental health changes in healthcare workers during the COVID-19 pandemic: A systematic review of longitudinal studies. *Evaluation & the Health Professions*, 47(1), 11–20.

Vadera, S., & Ameen, S. (2022). Methods for pruning deep neural networks. *IEEE Access*, 10, 63280–63300.

Venkateswarlu, I. B., Kakarla, J., & Prakash, S. (2020). Face mask detection using MobileNet and global pooling block. In *2020 IEEE 4th Conference on Information & Communication Technology (CICT)*, 1–5.

Wang, Q., & Guo, G. (2021). DSA-Face: Diverse and sparse attentions for face recognition robust to pose variation and occlusion. *IEEE Transactions on Information Forensics and Security*, 16, 4534–4543.

Wang, Z., Huang, B., Wang, G., Yi, P., & Jiang, K. (2023). Masked face recognition dataset and application. *IEEE Transactions on Biometrics, Behavior, and Identity Science*.

Wu, C. K., Cheng, C.-T., Uwate, Y., Chen, G., Mumtaz, S., & Tsang, K. F. (2023). State-of-the-art and research opportunities for next-generation consumer electronics. *IEEE Transactions on Consumer Electronics*, 69(4), 937–948.

Xia, Y., Yu, H., Wang, X., Jian, M., & Wang, F.-Y. (2022). Relation-aware facial expression recognition. *IEEE Transactions on Cognitive and Developmental Systems*, 14(3), 1143–1154.

Yadav, S. (2020). Deep learning-based safe social distancing and face mask detection in public areas for COVID-19 safety guidelines adherence. *International Journal of Research in Applied Science and Engineering Technology*, 8(7), 1368–1375. https://doi.org/10.22214/ijraset.2020.30560

Zhong, M., Xiong, W., Li, D., Chen, K., & Zhang, L. (2024). MaskDUF: Data uncertainty learning in masked face recognition with mask uncertainty fluctuation. *Expert Systems with Applications*, 238, 121995.

Zhuang, F., et al. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1), 43–76.