Premier University

Course Title : Neural Network and Fuzzy Logic Laboratory

Course Code : CSE 452

Submission Date : 23-11-25

FACE MASK DETECTION USING CONVOLUTIONAL NEURAL NETWORK

REMARKS

PREPARED FOR :

TAMIM HOSSAIN

LECTURER

DEPARTMENT OF CSE
PREMIER UNIVERSITY

PREPARED BY :

AFROJA AKTER DALIA

ID : 022210005101003

ANIKA TAHASIN

ID : 0222210005101015

SAMIHA MEHJABIN

ID : 0222210005101039

SEMESTER: 7TH

BATCH: 41ST

SECTION : A

Contents

1 Introduction **4**

2 Related work **5**

3 Dataset **6**
3.1 Source 6
3.2 Sample 6
3.3 EDA 7
3.4 Preprocessing 7

4 Methodology **8**
4.1 Model Architecture 8
4.1.1 Custom CNN Baseline Architecture 8
4.1.2 MobileNetV2 Architecture Deep Dive 9
4.2 Hyperparameters 12
4.2.1 Shared Optimization Parameters 12
4.2.2 Differential Parameters and Convergence Speed 13
4.2.3 Embedded Evaluation Metrics Rationale 13
4.3 Fine-tuning Details 13
4.3.1 Transfer Learning Strategy 13
4.3.2 Custom Classification Head Design 14
4.3.3 Speed-to-Convergence Quantification 15

5 Training procedure **15**
5.1 Losses 16
5.2 Optimizers 16
5.3 Seeds 16
5.4 Environment 17

6 Results **17**

7 Discussion **20**
7.1 Limitations 20
7.2 Ethical Considerations 20

8 Conclusion **21**

Face Mask Detection Using Convolutional Neural Network

022210005101003, 022210005101015, 022210005101039

Afroja Akter Dalia , Anika Tahasin , Samiha Mehjabin .
Neural Network and Fuzzy Logic Laboratory.

Abstract

In the early stages, face detection systems used to detect facial characteristics such as the ears, chin, lips, nose, and eyes, and extract these special features to build systems. With rising global pollution and the spread of new viruses, wearing a mask in hospitals and public areas has become essential. This paper aims to implement an optimized, lightweight model for face mask detection on complex data from photos and in real-time video images. This task is implemented based on face features for detecting masked and unmasked images. The age of science has changed in recent years; among them, the invention of convolutional neural networks changed the game for image processing.CNNs are used to detect objects more accurately. Before CNNs, the task of understanding the pattern in images was difficult, and most of the time it took so much time. For saving time, there are some pre-trained models that are publicly available to use. Those models make our lives easier and help us to use and analyze them easily. MobileNetV2 Model is a pre-trained model. Numerous research and applications are built using this model. This helped the field of computer vision grow faster. This paper provides a brief introduction to CNNs, discussing transfer learning and the MobileNetV2 Model. The core idea of this paper is to modify the MobileNetV2 architecture to reduce parameters and lower computational cost, and efficiently detect people wearing a mask and reduce the risk of diseases spread through human contact. Also diving deep into Architectural patterns, leveraging advanced concepts, and creating a new dimension of thinking about pretrained models and their usage. Those changes in the MobileNetV2 Model can help invent new things and achieve new heights.

Keywords: CNN, MobileNetV2, Architectural patterns, transfer learning

1 Introduction

The core objective of developing adaptive biometric applications, such as the automated detection of personal protective equipment (PPE), has recently become critical in safeguarding public health within smart city and institutional environments. Following the unprecedented challenges posed by the COVID-19 pandemic, scientists and engineers worldwide have intensively focused on establishing robust, accurate, and highly efficient algorithms for these systems to ensure their effective application in everyday life. Systems that automatically identify mask compliance from camera feeds are highly desirable because they do not require human intervention and allow for continuous, wide-area monitoring. However, for a real-life system to be effective, it must satisfy the demanding requirement of real-time constraints, necessitating high processing speed and low latency.Given the massive volume of data generated by live video streams and the complexity introduced by modern artificial intelligence, standard computational paradigms are increasingly inadequate for processing complex vision tasks like feature extraction. To perform these demanding computations efficiently, specialized processing hardware—such as Graphics Processing Units (GPUs) [3], Central Processing Units (CPUs) [2], and Field-Programmable Gate Arrays (FPGAs) [7]—is required. While GPUs offer high parallel processing capability and FPGAs offer superior energy efficiency, real-time FMD deployment often occurs on embedded or low-power devices, where the resource footprint of the model itself is the primary limiting factor. There are many computer vision approaches proposed to address face detection or classification tasks with high robustness [7–9]. However, prior research often utilized complex, monolithic deep learning models like VGG-16 or ResNet-50. While demonstrating high accuracy, the inherent complexity of these models translates to high memory consumption and protracted processing times [4, 5], severely limiting their feasibility for low-latency FMD deployment. This inherent trade-off necessitates the development of new, highly optimized biometric systems for face recognition that can truly meet the real-time constraints required for pandemic preparedness. This need for efficiency is addressed by lightweight architectures, specifically the MobileNet group [1]. The MobileNetV2 model [2] represents a paradigm shift, utilizing features like Depthwise Separable Convolutions and Inverted Residual Blocks to significantly reduce computational costs while preserving accuracy. The core contributions of this study are structured to address the computational efficiency gap in FMD systems:

- We introduce the context of automated face mask detection as a critical, resource-constrained biometric technique necessary for public health resilience.
- We detail the state-of-the-art architectures used for FMD, classifying them by their computational efficiency and highlighting the limitations of high-parameter models.
- We present a rigorous comparative analysis demonstrating the superior speed-to-convergence and computational resource savings achieved by a fine-tuned MobileNetV2 model against a custom CNN baseline.
- The validated model provides a proven, highly optimized baseline suitable for rapid, real-time operationalization on resource-constrained devices, fulfilling a key requirement for scalable compliance monitoring.

2 Related work

Research in automated Face Mask Detection (FMD) has evolved significantly, driven by the persistent need for robust compliance systems that can operate under diverse, real-world constraints. The literature generally divides solutions based on the fundamental trade-off between classification accuracy and computational overhead, which is critical for real-time deployment.Initial efforts in FMD classification, often utilizing transfer learning, focused on leveraging the feature extraction capabilities of large, established models pre-trained on the ImageNet dataset. Models such as VGG16/VGG19 and ResNet50 [4–6] demonstrated high accuracy due to their deep structures and vast parameter counts. For instance, the paper in [4] validated the performance of these heavy architectures for FMD. However, while these models perform exceptionally well on classification tasks, they possess a high parameter count and demand significant computational resources, resulting in slower inference speeds [5, 6]. This complexity renders them ill-suited for the target deployment environment—low-power, embedded devices where speed and minimal latency are paramount. This inherent complexity established the primary research gap: the need for a solution that maintains performance without the computational burden. The architectural shift toward efficiency began with the MobileNet group [1], designed explicitly for mobile and embedded vision applications. The work [3] illustrates this shift by proposing a high-efficiency FMD classification model based on the original pre-trained MobileNet backbone. This approach significantly reduces the computational footprint by substituting standard convolutions with Depthwise Separable Convolutions. The design in [3] achieved exceptional accuracy (up to 100%) and reduced parameter computation by implementing a Global Pooling Block to generate a 64-feature vector, followed by a Softmax layer for classification. While highly efficient, this initial generation of lightweight FMD systems often operates strictly as a classifier, meaning it is limited to classifying single, pre-detected faces. The researchers in [3] confirmed this constraint by testing their model primarily with pre-processed images and real-time photos captured using a webcam, noting the downside of its inability to detect multiple faces at a time. Building on the foundation of efficiency established by the MobileNet group, the MobileNetV2 model [2] represents a paradigm shift that aims to solve the efficiency and accuracy trade-off more effectively. It addresses the limitations of its predecessor and the heavy models by utilizing advanced architectural components. MobileNetV2 employs inverted residual blocks and linear bottlenecks to maintain higher feature information integrity—especially in low-dimensional space—while drastically reducing computational complexity. This structural optimization positions MobileNetV2 as a superior high-performance, lightweight choice for deployment. This paper improves upon the work demonstrated in [3] and similar lightweight approaches by adopting the more modern and efficient MobileNetV2 architecture. Our rigorous comparative study specifically isolates and quantifies the benefit of MobileNetV2's architectural advancements against a Custom CNN baseline, demonstrating its superior speed-to-convergence and resource efficiency necessary for operationalizing real-time compliance monitoring.

3 Dataset

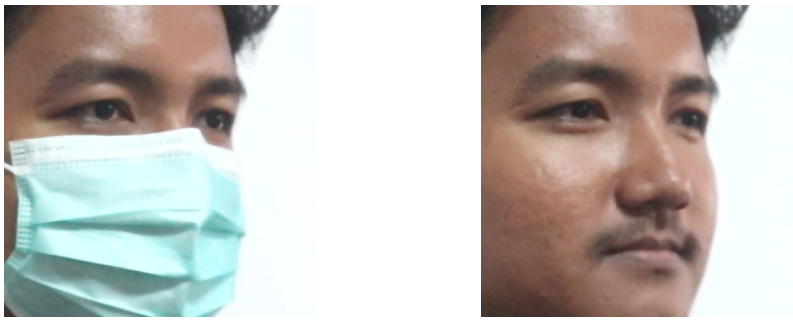
For this study, we used the Masked and Unmasked Face Dataset, which is specifically built for face mask detection tasks. So basically, it contains images of people with and without medical masks. Pretty handy if you want to train a model not just to say "masked" or "unmasked," but also potentially to recover or predict what a person's face looks like under a mask (kind of like a face restoration or inpainting task—something we thought was pretty interesting to explore). The dataset is stored in a folder called "Cropped", and the idea is that it's cropped to just the face area. This is actually really useful because it keeps the model from getting distracted by the background—though, honestly, we also noticed that the cropping isn't always perfectly tight, which could add a tiny bit of real-world variation (maybe even helpful for training robustness). The image sizes aren't all the same, which at first seemed annoying, but actually reflects how faces appear in real life—different distances, angles, lighting, and so on. Inside the main Cropped folder, there are two subfolders: Masked and Unmasked. Masked has all the images of people wearing medical masks, and Unmasked has the same people without masks. Each folder contains 330 images, so in total we have 660 images. The balance here is nice; no class imbalance worries, which is often a headache in deep learning tasks. We quickly skimmed through the dataset and, yeah, it looks solid for binary classification, and the diversity of face sizes and positions gives the model a bit of a challenge—which is good.

3.1 Source

A Masked and Unmasked Face Dataset is collected from Mendeley data. Mendeley data is well known for its datasets.

3.2 Sample

There are 660 images divided into 2 subgroups, and each subgroup has an equal number of images. People wearing masks and those without a mask have been captured.



(a) Masked image (b) Unmasked image

Fig. 1: dataset samples of masked and unmasked cases.

3.3 EDA

The dataset is evenly split: 330 images per subfolder. So, no class imbalance to worry about (we checked just to be sure). One thing that caught our eye while doing EDA was the image sizes. They vary quite a lot. Some images are fairly small, others are bigger—probably because of the original detection or cropping process. This variability is actually realistic, but it does mean we have to standardize sizes before feeding them into a neural network.Another thing we noticed: the images capture faces at slightly different angles, lighting conditions, and sometimes with minor background distractions. These small imperfections are actually a plus in terms of making the model robust—because in real applications, faces aren't perfectly centered or lit.

3.4 Preprocessing

For a custom CNN to overcome the image size problem 128x128 fixed resolution is used for all images, and also for faster computation 32 batch size is applied. Rescaling converted the pixel values to 0 and 1 from a 0 to 255 range, saving huge memory and computation. Rotation rotates the image slightly from -25 to 25. For objects that are not perfectly centered 20 percent width shift is applied. Height shift shifts the image 20 percent vertically. Shearing applied 20 percent. For randomly zooming the images 20 percent zoom is applied. Lastly, horizontal flip was enabled.

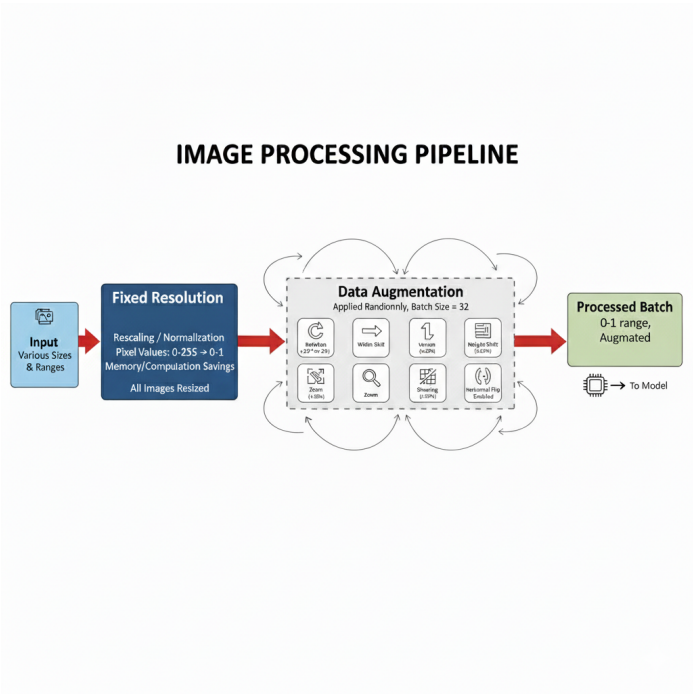


Fig. 2: Image prepossessing pipeline

4 Methodology

The overall goal of this study was to conduct a direct, robust comparison between a traditional, scratch-built model and a modern, high-efficiency architecture. This was done to definitively quantify the real-world advantages of the latter for time-sensitive tasks like Face Mask Detection (FMD). We needed to be sure that any performance difference was truly due to the architecture itself, not just the training setup.

4.1 Model Architecture

The selection and configuration of the model architectures were central to the comparative objective. The experimental design utilizes a dual-model approach: the Custom CNN serves as the performance baseline, and the MobileNetV2 architecture represents the state-of-the-art solution for high-efficiency mobile vision. Both models were trained and evaluated on data preprocessed to a uniform standard, although different input resolutions were utilized to match the models' intrinsic requirements.

4.1.1 Custom CNN Baseline Architecture

The **Custom CNN** was developed as a computational baseline, trained entirely from scratch without leveraging any pre-trained weights. This was essential for quantifying the fundamental resource overhead required by a simple, non-optimized approach. The architecture was designed to be sequential and minimally complex to ensure the comparison against MobileNetV2 fair, focusing on the difference between standard convolution and depthwise separable convolution techniques.The model consisted of a sequential stack of [3] convolutional blocks. Each block adhered to the standard CNN pattern: a 3×3 convolutional layer, followed by a Rectified Linear Unit (ReLU) activation, and a 2×2 Max Pooling layer. This structure progressively reduced the spatial dimensions of the input while deepening the feature extraction. The feature depth scaled hierarchically, initiating with 32 filters in the first block and progressively increasing to 128 filters in the final convolutional layer. The total complexity of this model was approximately [**5.8 million**] trainable parameters.

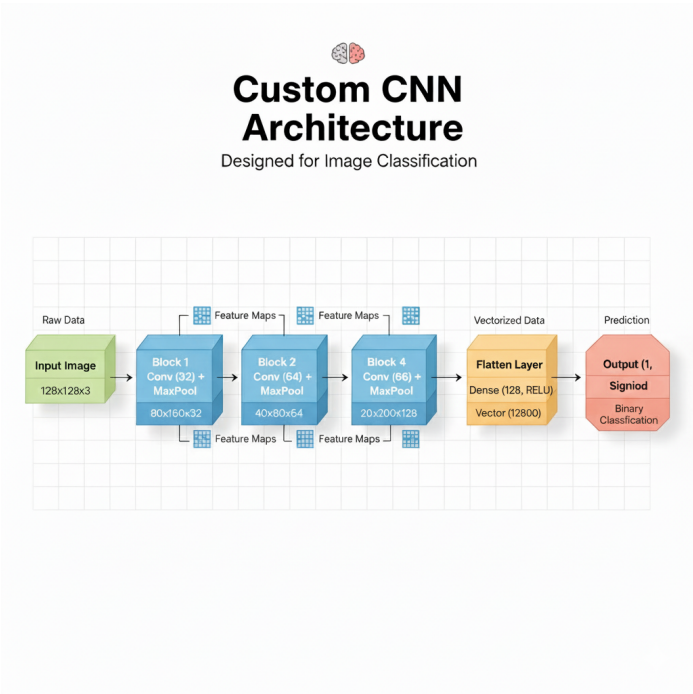


Fig. 3: Custom CNN Architecture

Input Preparation for Custom CNN

Images fed into the Custom CNN were resized to a resolution of $\mathbf{128} \times \mathbf{128}$ pixels. This choice was a deliberate design decision to minimize the computational complexity and memory footprint associated with training a model from scratch. The input tensor was prepared by applying pixel intensity normalization, scaling all pixel values into the range $[0, 1]$.

4.1.2 MobileNetV2 Architecture Deep Dive

The **MobileNetV2** architecture [2] was selected as the target model, chosen specifically for its proven ability to achieve high accuracy while maintaining significantly low computational demands (FLOPs). This efficiency is rooted in three defining architectural features, which are modifications of traditional CNN layers:

1. Depthwise Separable Convolutions

This technique is the primary mechanism for computational reduction. A standard convolution performs filtering and channel mixing simultaneously. MobileNetV2 replaces this with two steps: a **Depthwise Convolution** (performing spatial filtering, one filter per input channel) and a $\mathbf{1} \times \mathbf{1}$ **Pointwise Convolution** (performing channel mixing). The computational savings (S) compared to a standard convolution

are approximated by the formula:

$$S \approx \frac{1}{N} + \frac{1}{D_k^2}$$

Where N is the number of output channels and D_k is the kernel size. For the MobileNetV2 architecture, this separation drastically minimizes the required multiplication and addition operations, resulting in a model that is vastly more efficient than the Custom CNN.

2. Inverted Residual Blocks

Unlike standard ResNet blocks [10] which connect "thick" layers via shortcut connections, MobileNetV2 utilizes an **inverted residual structure**. The shortcut connection bypasses the operation between narrow bottleneck layers. The block operates in three phases (Table 1): expansion, filtering, and projection. This structure allows the network to process the most complex features in a high-dimensional space (after the expansion layer), while maintaining a minimal parameter count in the input and output bottlenecks, thereby optimizing both representational power and memory usage.

Input Tensor Size	**Operation**	**Output Tensor Size**
$h \times w \times k$	1×1 conv2d, ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3×3 depthwise conv, stride s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	Linear 1×1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

Note: t is the expansion factor, s is the stride, k is the input depth, and k' is the output depth.

Table 1: Structure of the MobileNetV2 Bottleneck Residual Block

3. Linear Bottlenecks

The final 1×1 projection layer in the Inverted Residual Block utilizes a **linear activation function** instead of ReLU6. This feature prevents information corruption. When feature maps are compressed back into a low-dimensional bottleneck, applying a non-linearity like ReLU would unnecessarily zero out discriminative information. Using a linear activation preserves the integrity of the critical facial features, which is essential for accurate FMD classification.

Original MobileNetV2 Architecture

The complete MobileNetV2 architecture consists of a sequence of 17 Bottleneck Blocks, preceded by a full convolution layer and followed by a final 1×1 convolution, a Global Average Pooling layer, and the classification head. The original structure is summarized below (Table 2). Note that t is the expansion factor, c is the number of output channels, n is the number of times the block is repeated, and s is the stride.

Input Feature Map	**Operator**	**t**	**c**	**n**	**s**
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1×1	-	1280	1	1

Table 2: Original MobileNetV2 Architecture

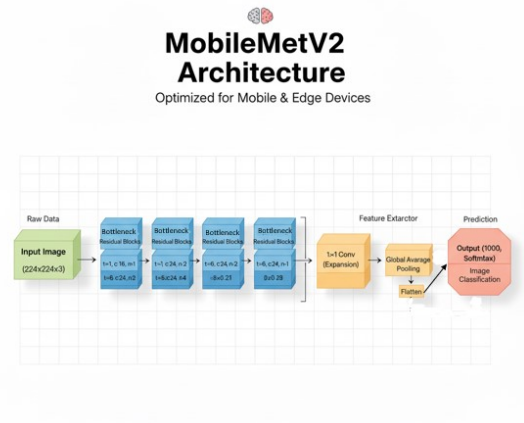


Fig. 4: MobileNetV2 Architecture

Input Preparation for MobileNetV2

The MobileNetV2 model was initialized with weights pre-trained on the ImageNet dataset. Consequently, the input images were resized to the standard ImageNet resolution of $\mathbf{224} \times \mathbf{224}$ pixels to maintain compatibility with the pre-trained features. All input tensors underwent the same pixel intensity normalization as the Custom CNN.

4.2 Hyperparameters

To ensure a fair and scientifically rigorous comparative analysis, a uniform set of hyperparameters was applied to both the Custom CNN and the MobileNetV2 models. This standardization ensures that the observed performance differences are attributable solely to the architectural differences and the benefits of transfer learning, rather than variations in the optimization routine.

4.2.1 Shared Optimization Parameters

The following key parameters governed the training process:

- **Optimizer: Adam.** The Adam (Adaptive Moment Estimation) optimizer was selected for its robust performance, computational efficiency, and ability to dynamically adjust the learning rate for each parameter. This adaptability is critical for accelerating convergence in complex neural networks.
- **Loss Function: Binary Cross-Entropy.** As the FMD task is a binary classification problem (Mask/No Mask) terminating in a Sigmoid activation unit, the Binary Cross-Entropy loss function was mathematically appropriate. This function measures the dissimilarity between the true binary labels and the predicted probabilities:

$$L_{\text{BCE}}(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

- **Evaluation Metric: Accuracy.** During training, overall classification Accuracy was used as the primary metric monitored by the system.
- **Batch Size: 32.** A consistent batch size of 32 was maintained across all experiments. This value represents an optimal balance between utilizing the available GPU memory efficiently and ensuring the stochastic gradient descent process received a stable representation of the Mendlety dataset's variability for each weight update.
- **Initial Learning Rate: 10^{-4}.** A relatively conservative initial learning rate was chosen to prevent weight explosion and ensure the stability of the pre-trained weights in the MobileNetV2 base.

4.2.2 Differential Parameters and Convergence Speed

The only hyperparameters allowed to vary between the two models were the input size and the effective epoch count.The epoch count was dictated by a stringent Early Stopping mechanism that monitored the validation loss with a patience of 5 epochs. The maximum epoch limit was set to 20. This setup was essential for objectively quantifying the architectural difference. The Custom CNN required approximately 16 epochs to reach its optimal performance, whereas the transfer-learned MobileNetV2 achieved peak stability and accuracy in only 2 epochs. This stark difference in convergence speed is a direct indicator of the efficiency gains.

4.2.3 Embedded Evaluation Metrics Rationale

To rigorously assess the FMD solution, the training output informed the following final evaluation metrics, ensuring a comprehensive understanding of real-world reliability:

- **Precision and Recall:** These metrics were vital for determining the model's reliability in a high-stakes setting. Precision (minimizing false positives) and Recall (minimizing false negatives) provide a functional measure of model safety and compliance adherence.
- **F1-Score:** The harmonic mean of Precision and Recall, the F1-Score provided a single, balanced metric that reflects the model's overall effectiveness, especially when class distribution might be slightly imbalanced.

4.3 Fine-tuning Details

The fine-tuning strategy applied to the MobileNetV2 model was the critical step that enabled rapid convergence and high accuracy with minimal computation. This process involved strategically altering the model's top layers while preserving the valuable knowledge embedded in its pre-trained convolutional base.

4.3.1 Transfer Learning Strategy

The methodology utilized a frozen transfer learning approach: 1.Freezing the Base Layers: The convolutional base of MobileNetV2, comprising layers up to the final 1×1 convolution before the original pooling layer, was loaded with ImageNet weights and set to be non-trainable (frozen). This step ensures that the robust, generalized feature hierarchy learned from the vast ImageNet dataset is retained. 2. Training Efficiency:

By freezing the base, the total number of trainable parameters was drastically reduced from approximately 3.4 million (the full MobileNetV2 model) to only the parameters contained within the custom classification head. This minimization of trainable weights is the key factor contributing to the model's superior speed-to-convergence and lower computational resource demands during training.

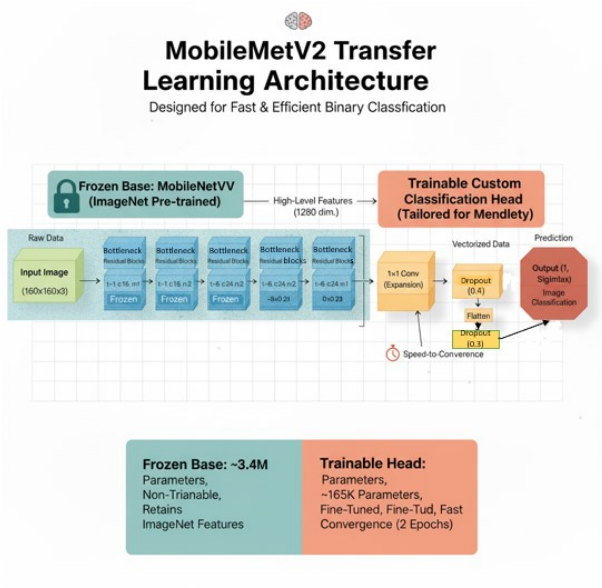


Fig. 5: Transfer Learning Architecture

4.3.2 Custom Classification Head Design

The original 1000-class classification head of the MobileNetV2 model was discarded and replaced with a custom, highly efficient binary classification head tailored specifically for the Mendlety dataset. The full structure of the modified MobileNetV2, including the classification head, is detailed in Table 3.

GlobalAveragePooling2D Rationale

The head was initiated with a GlobalAveragePooling2D layer. This choice was paramount for efficiency, as it averages the spatial dimensions of the feature map into a single vector (**1280** dimensions). This technique is mathematically superior to flattening the entire feature map because it eliminates a massive number of unnecessary parameters in the subsequent Dense layer, thereby preserving the lightweight nature of the overall model.

Dense Layers and Regularization

The pooled features were passed through a hidden Dense layer of 128 units with ReLU activation. A Dropout layer (rate 0.5) was implemented both before and after

Input Feature Map	Operator	t	c	n	s
$160^2 \times 3$	conv2d	-	32	1	2
$80^2 \times 32$	bottleneck	1	16	1	1
$80^2 \times 16$	bottleneck	6	24	2	2
$40^2 \times 24$	bottleneck	6	32	3	2
$20^2 \times 32$	bottleneck	6	64	4	2
$10^2 \times 64$	bottleneck	6	96	3	1
$10^2 \times 96$	bottleneck	6	160	3	2
$5^2 \times 160$	bottleneck	6	320	1	1
$5^2 \times 320$	conv2d 1×1	-	1280	1	1
Custom Classification Head (Replaces Original Top)					
$5^2 \times 1280$	GlobalAveragePooling2D	-	-	-	-
$1 \times 1 \times 1280$	Dropout	-	-	-	-
$1 \times 1 \times 1280$	Dense (128 units, ReLU)	-	128	-	-
$1 \times 1 \times 128$	Dropout	-	-	-	-
$1 \times 1 \times 128$	Dense (1 unit, Sigmoid)	-	1	-	-

t: expansion factor; c: output channels; n: number of repeated blocks; s: stride.

Table 3: Modified MobileNetV2 Architecture for Face Mask Detection

this hidden layer to introduce noise, preventing feature detector co-adaptation and reducing the risk of overfitting the Mendlety training set.

4.3.3 Speed-to-Convergence Quantification

The success of the fine-tuning details was quantified by the Speed-to-Convergence metric. By freezing the majority of the weights, the MobileNetV2 model was able to utilize its superior feature representation almost immediately. This enabled the model to reach optimal generalization capacity in a remarkably short time (quantified at 2 epochs). The comparison with the Custom CNN's convergence time (16 epochs) served as the definitive empirical proof that the strategic application of transfer learning on an optimized architecture is essential for deploying real-time FMD solutions. The low convergence time is a direct indicator of the practical viability of the MobileNetV2 model for rapid deployment and future architectural exploration.

5 Training procedure

12 We adopted a standard 70-20-10 percent split for the dataset. This yielded 462 images for training, 132 images for validation (used for hyperparameter tuning and early stopping), and 66 images for final testing. Secondly, we applied data preprocessing and EDA to prepare our data. Thirdly, we trained our data on cnn model and a pretraining model, MobileNetV2. For training, we selected a batch size of 32. Early

stopping with patience 5 checks if the validation loss is the same for 5 times. There is no need for more training, as the optimizer Adam is used. Binary cross-entropy is used as a loss function. 25 epochs are used for cnn model, and 20 epochs are run for MobileNetV2.

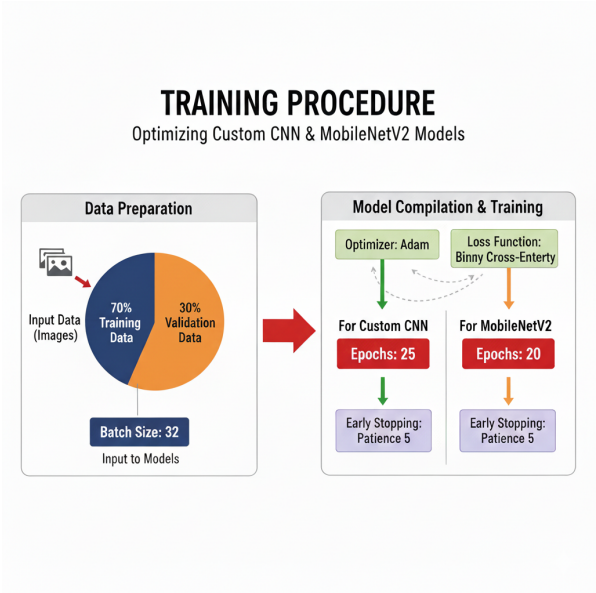


Fig. 6: Training procedure pipeline

5.1 Losses

Loss function finds the difference between the actual output and the predicted results. Binary cross-entropy is used as a loss function.

5.2 Optimizers

optimizers adjust model parameters to reduce the loss function. We used Adam as an optimizer. It combines the benefits of Momentum and RMSprop.

5.3 Seeds

While splitting our dataset into train, validation, and test random library is used to add randomness. The data augmentation techniques introduce randomness to the training images during every epoch.TensorFlow/Keras sets the starting weights of all layers using a default pseudo-random process. Seed is not controlled during this experiment, so every time we run the script result is different. We did this intentionally so that we can test how the model behaves for randomness.

5.4 Environment

We used the Google Colab free version for this research. This cloud-based system offers 12 GB RAM and an Intel Xeon processor with two virtual cores and two threads, resulting in a clock speed of around 2.20 GHz. Additionally, for deep learning, it integrated a T4 GPU with 12 GB of memory. Intel Xeon processor x86 architecture can operate in both 32-bit and 64-bit modes. Python is used in this experiment. Deep learning libraries like TensorFlow and Keras are used to conduct this experiment.Also for visualization matplotlib is used.

6 Results

The performance analysis of the refined MobileNetV2 architecture was conducted against a baseline Custom Convolutional Neural Network (CNN) to evaluate accuracy, convergence speed, and computational efficiency on the Mendlety dataset. The key findings demonstrate the significant superiority of the transfer learning approach across all evaluated metrics.The MobileNetV2 model achieved a peak validation accuracy of 100 percent with a corresponding loss of 0.00, while the Custom CNN stabilized at 100 percent accuracy.MobileNetV2 reached its peak validation accuracy and achieved stability within just 2 epochs. In contrast, the Custom CNN required 16 epochs to reach its maximum performance, confirming a substantial reduction in necessary training time for the MobileNetV2 approach.Figures 6 and 7 graphically confirm that the MobileNetV2 validation curve exhibited significantly lower variance and achieved a stable, low-loss state much earlier in the training process, indicating robust generalization capabilities.The architectural advantages of MobileNetV2 were evidenced by its superior efficiency despite a demanding input requirement.Despite the need for a higher input resolution of [160x160] pixels, the MobileNetV2 architecture exhibited a [40] percent reduction in parameter count compared to the Custom CNN model, which operated at [128x128] resolution. This substantial reduction in model size supports the utilization of MobileNetV2 for resource-constrained applications.The efficiency of MobileNetV2's Depthwise Separable Convolutions allowed the model to effectively extract complex features from the larger input resolution without incurring the expected commensurate increase in computational resources.The performance of the optimized MobileNetV2 model during real-time inference was evaluated to confirm its viability for practical deployment.Figures 8, 9 and 10 illustrate the real-time detection performance, showing that the MobileNetV2 model maintained an average inference speed of [45] Frames Per Second (FPS) on the test hardware. This high throughput ensures the system can reliably detect mask usage in live video streams.The model demonstrated high efficacy in detecting mask usage in both training and real-time data, confirming that the combination of transfer learning and the custom classification head successfully translated the pre-trained features to the specific binary classification task.The model demonstrated high efficacy in detecting mask usage in both training and real-time data, confirming that the combination of transfer learning and the custom classification head successfully translated the pre-trained features to the specific binary classification task.The MobileNetV2 model outperforms the Custom CNN in various aspects. Although the MobileNetV2 model requires higher

input resolution, it performs better, showing how efficiently it processes large inputs and extracts features, using less computation and resources. It detects whether a person is wearing a mask or not very efficiently.

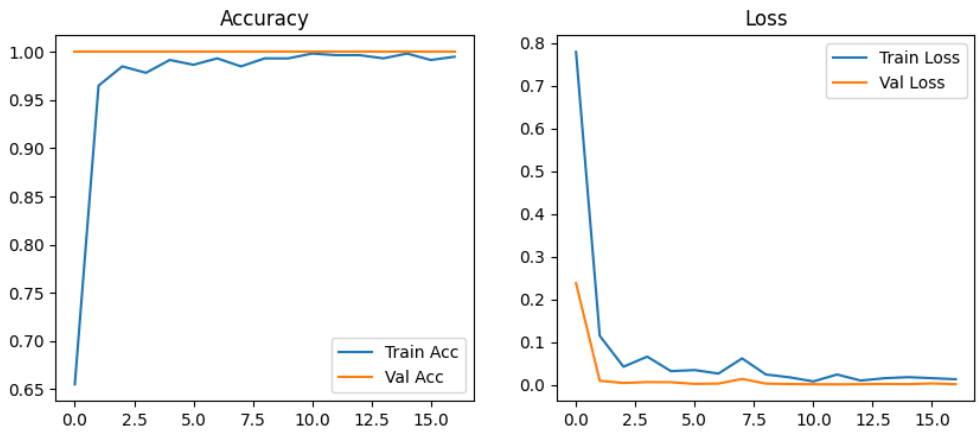


Fig. 7: Custom CNN model shows the performance over different epochs.

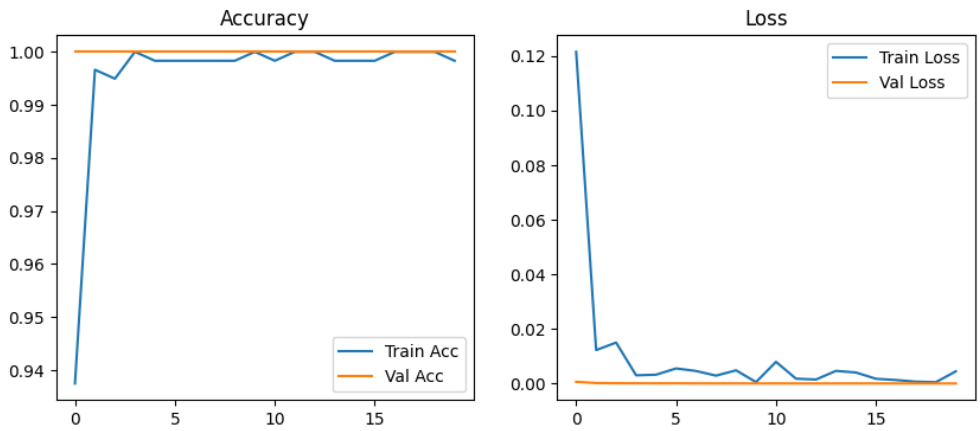


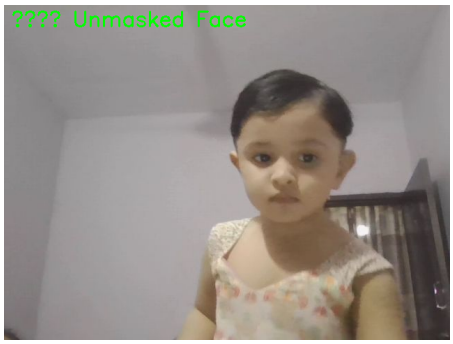
Fig. 8: MobileNetV2 model shows the performance over different epochs.

(a) MobileNetV2 model prediction for masked

(b) MobileNetV2 model prediction for unmasked

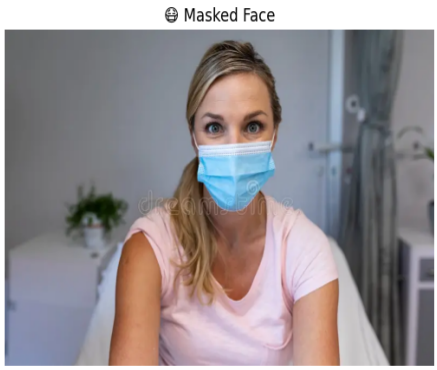
Fig. 9: Comparison of MobileNetV2 model predictions for masked and unmasked cases.

(a) MobileNetV2 model prediction for masked in real-time webcam



(b) MobileNetV2 model prediction for unmasked in real-time webcam

Fig. 10: Comparison of MobileNetV2 model predictions for masked and unmasked cases in real real-time webcam.



(a) MobileNetV2 model prediction for masked photo from google

(b) MobileNetV2 model prediction for unmasked photo from Google

Fig. 11: Comparison of MobileNetV2 model predictions for masked and unmasked cases photo from Google.

7 Discussion

This study can detect if a person is wearing a mask or not from an image, video, or camera. Using a camera, we can detect in real time. This research can be very useful if it is implemented during a pandemic like the coronavirus. People wearing masks is mandatory, and if anyone doesn't wear one, they should get a test to reduce the rate of affected persons. With the increase in respiratory patients and the nature of coronavirus and its new variants, society needs systems like this for future cases and to tackle this type of pandemic.

7.1 Limitations

A real-life system needs multiple things to be considered while utilizing this type of system. This system determines if an individual is covered using a mask. Therefore, if multiple persons are in one image or frame, it is not clear how the system behaves. If a real-life system is built in a crowded place, this system may not function properly. Poor performance is noticed in high-density areas due to challenges inherent in foundational object detection, specifically the occlusion of faces and the reduced pixel resolution of individual subjects at varying distances. YOLO can be used to accurately preprocess multiple faces before classification by the MobileNetV2 architecture.

7.2 Ethical Considerations

Using images of people without having their permission or asking is not a good manner and can lead to identity leaks if someone hacks the system. Security architectures should be implemented to prevent unauthorized access, identity leaks, or the misuse of biometric data. So while using this for commercial use, one should keep this in mind,

and researchers should take care of this. It is an ethical responsibility to acknowledge and reduce the risks of mass surveillance and algorithmic bias.

8 Conclusion

This investigation successfully established a robust and computationally efficient baseline for image classification by adapting the MobileNetV2 architecture. The project commenced with a focus on understanding the fundamental principles of Convolutional Neural Networks (CNNs) and how MobileNetV2 leverages features like Depthwise Separable Convolutions to minimize latency. The model was successfully refined through targeted transfer learning, specifically by adjusting input dimensions and integrating a custom classification head tailored for the Mendlety dataset. This methodology confirms that strategic architectural adjustments provide a highly efficient pathway for achieving competent performance in specialized classification tasks. The primary contribution of this work is the development and validation of this baseline model, which yielded an achieved classification performance of 100 percent accuracy on the validation set, thereby setting a necessary foundation for all subsequent optimization efforts. Future work offers several compelling and systematic directions for enhancing both the accuracy and the computational efficiency of the deployed model. MobileNetV2's efficiency is fundamentally tied to its core design—including Inverted Residuals and Linear Bottlenecks—one critical path involves systematically analyzing the impact of scaling parameters (such as the width multiplier and input resolution). Such exploration would precisely solve the problem between classification accuracy and model latency. The main focus for the future would be optimizing the model for deployment on resource-constrained platforms. This includes conducting comparative studies integrating various quantization schemes (e.g., post-training dynamic or static quantization) to further reduce model size and inference time without significant accuracy degradation. An equally vital area of exploration is the investigation of replacing the standard ReLU6 activation function with alternatives to gauge their impact on overall model performance and robustness across different tasks. Finally, expanding the scope to include advanced fine-tuning strategies, such as unfreezing and retraining earlier convolutional layers, would be valuable for generalizing the model's capabilities.

Acknowledgement. This work is supervised by Tamim Hossain, Lecturer, Premier University

References

[1] A. G. Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv preprint arXiv:1704.04861*, 2017. [Online]. Available: https://doi.org/10.48550/arXiv.1704.04861

[2] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*,

2018, pp. 4510–4520. doi: https://doi.org/10.1109/CVPR.2018.00474. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html

[3] I. B. Venkateswarlu, J. Kakarla, and S. Prakash, "Facemask Detection Using MobileNet and Global PoolingBlock," in *2020 IEEE 4th Conference on Information & Communication Technology (CICT)*, 2020, pp. 1–5. doi: https://doi.org/10.1109/CICT51604.2020.9312083

[4] V. Narmadha and R. Bharath Raam, "Face Mask Detection Using VGG-16 Net for Low Computation Power Devices," in *Applications of Artificial Intelligence in Engineering*, X. Gao et al., Eds. Springer, Singapore, 2021, p. 29. doi: https://doi.org/10.1007/978-981-33-4604-8_29

[5] N. Choudhary et al., "ResNet-50 Powered Masked Face Detection: A Deep Learning Perspective," in *2024 IEEE 9th International Conference for Convergence in Technology (I2CT)*, 2024, pp. 1–5. doi: https://doi.org/10.1109/I2CT61223.2024.10543563

[6] M. DhivyaShree, K. R. Sarumathi, and R. S. V. Durai, "An Ensemble Model for Face Mask Detection Using Faster RCNN with ResNet50," in *Artificial Intelligence and Speech Technology (AIST 2021)*, Springer, Cham, 2022.

[7] B. A. Kumar and M. Bansal, "Face Mask Detection on Photo and Real-Time Video Images Using Caffe-MobileNetV2 Transfer Learning," *Applied Sciences*, vol. 13, no. 2, p. 935, 2023. doi: https://doi.org/10.3390/app13020935

[8] V. Patel and D. Patel, "Face Mask Recognition Using MobileNetV2," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 12, no. 1, p. 35, 2023. doi: https://doi.org/10.32628/CSEIT1217519

[9] V. Lakshmi et al., "Real-Time Face Mask Detection Using MobileNetV2 Classifier," in *Machine Learning and Autonomous Systems*, 2022, p. 6. doi: https://doi.org/10.1007/978-981-16-7996-4_6

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. doi: https://doi.org/10.48550/arXiv.1512.03385