Research Article

Proposed Detection Face Model by MobileNetV2 Using Asian Data Set

Phat Nguyen Huu, 1 Vinh Tran Quang, 1 Chau Nguyen Le Bao, 2 and Quang Tran Minh 3, 4

1 School of Electrical and Electronic Engineering, Hanoi University of Science and Technology, Hanoi, Vietnam
2 Specialized Math Class, Hanoi – Amsterdam Highschool for the Gifted, Hanoi, Vietnam
3 Department of Information Systems, Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet, District 10, Ho Chi Minh City, Vietnam
4 Vietnam National University Ho Chi Minh City (VNU-HCM), Linh Trung Ward, Thu Duc District, Ho Chi Minh City, Vietnam

Correspondence should be addressed to Phat Nguyen Huu; phat.nguyenhuu@hust.edu.vn

Received 21 February 2022; Revised 23 August 2022; Accepted 24 September 2022; Published 11 October 2022

Academic Editor: Gongping Yang

Copyright © 2022 Phat Nguyen Huu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In 2019, the infectious coronavirus disease 2019 (COVID-19) was first reported in Wuhan, China. It has then become a public health problem in the world. This pandemic is having a heavy impact on the lives of people in our country. All countries are trying to control the spread of this disease. To solve the problem, each person needs to wear masks in a public place. Therefore, we propose a model capable of distinguishing between masked and nonmasked faces using a convolutional neural network (CNN) based on deep learning (DL)—MobileNetV2 in this paper. The model can detect people who are not wearing masks. It has an accuracy of up to 99.37%. The model will be applied in places such as schools, offices, and so on to monitor the wearing masks.

1. Introduction

According to the World Health Organization [1–4], coronavirus disease 2019 (COVID-19) is an infectious disease caused by the SARS-CoV-2 virus. The virus can spread from an infected mouth or nose in small liquid particles when they cough, sneeze, speak, sing, or breathe.

Most people infected with the virus will experience mild-to-moderate respiratory illness and recover without requiring special treatment. However, some people will become seriously ill and require medical attention. Older people with underlying medical conditions such as cardiovascular disease, diabetes, chronic respiratory disease, or cancer are more likely to develop serious illnesses. Anyone can get sick with COVID-19 and become seriously ill or die at any age.

The best way to prevent and slow down transmission is to be well-informed about the disease and how the virus spreads. We can protect ourselves and others from infection by staying at least 2 meters apart from others, wearing a properly fitted mask, washing our hands, or using an alcohol-based rub frequently.

Vaccines have been developed. However, they only can relieve symptoms while infecting and cannot prevent the spread of disease. Vietnam has carried out vaccination coverage across the country and is aiming to bring activities back to normal. Therefore, wearing a mask is essential to slow down the spread of COVID-19.

However, everyone does this well all time. Many people do not wear masks or wear them the wrong way in public. This greatly affects the prevention of disease.

To support the control of mask-wearing in public places, we propose a model that can recognize and distinguish between people who wear and do not wearing masks. The expected model is trained based on deep learning (DL)—MobileNetV2 in this paper. The model can detect people who are not wearing masks. It has an accuracy of up to 99.37%. The model will be applied in places such as schools, offices, and so on to monitor the wearing masks.

First, a face detector model, Retina Face, was used to detect faces.
Second, we use the MaskTheFace program to create our data set.
Finally, the MobileNetV2 model is trained on our data set and used to classify whether the input faces are masked or not.

The rest of the paper is presented as follows. In Section 2, we will present related work. In Sections 3 and 4, we present and evaluate the effectiveness of the proposed model, respectively. Finally, we give a conclusion in Section 5.

2. Related Work

Since the outbreak of the COVID-19 pandemic, people have been severely affected. This respiratory disease has a rapid spread. Countries around the world have had to apply a lot of measures to prevent it, even lock down the country, and not allow people moving from other countries. Up to now, many vaccines have been developed. These vaccines work to reduce symptoms and the impact of the disease on people. However, they cannot prevent the infection [5–7].

The disease is no longer too dangerous for healthy people with the current high vaccination coverage rate. However, it is still dangerous for underage children and the elderly who have other diseases. All countries are moving toward the reopening of outdoor activities and tourism services to prevent economic decline. Our country is also moving toward normalcy. Everyone can resume activities like before: traveling, going to the office, or going to school, as shown in Figures 1 and 2.

However, we should still follow pandemic prevention regulations to minimize risks. The most strictly recommended measure is wearing a mask. Therefore, an automatic system to detect people who are not wearing masks is the subject of much attention to be able to control compliance with mask-wearing wells.

There are many researches based on masked face detection [8–23]. In 2020, a face mask detection model using YoLo-V4 is published [8]. The system has already been installed at Politeknik Negeri Batam. The authors show that the system works well with high accuracy and speeds up to 11 fps. This speed is quite impressive that can be applied in real time. However, the specific accuracy and data set are not published. Therefore, we cannot estimate the accuracy of the model. The authors [9] have designed a face mask identification method using the SRCNet classification network and achieved an accuracy of 98.7% in classifying the images into three categories, namely, correct, incorrect, and not wearing face masks. Their frame rate is 10 fps. This model can detect even cases of wearing the wrong mask while being quite accurate. Since the data set used to train the model is aimed at diversity, it does not focus on a single group of objects.

The authors [10] have successfully researched a deep learning model for detecting masks over faces in public places. The proposed model efficiently handles varying kinds of occlusions in dense situations by making use of an ensemble of single- and two-stage detectors. The accuracy of the model is 98.2% for mask detection with an average inference time of 0.05 seconds per image. The paper [11–13, 24] presents the MobileNetV2 deep learning (DL) method to detect mask wearers on real-time images and video. The network is trained to perform two-layer identification of people wearing and without masks. Sample images were obtained from the real-world masked face data set database (RMFRD). The results show that the accuracy of network is 99.22% with 4,591 samples.

The above masked face detection models are all trained by some famous masked face data sets such as Real-World Masked Face Dataset (RMFRD), labeled faces in the wild (LFW), and face mask label data set (FMLD). People on each continent have different appearance characteristics: face, hair, skin color, eye color, and so on. In addition, people prefer and use different mask types in each country. Therefore, the existing data sets cannot contain all characteristics of all people in the country. The paper proposes a model to classify faces wearing masks with the scope of research in Vietnam and trains on the appropriate data set about the appearance and types of masks. This data set provides the closest features to Vietnamese people. Therefore, the trained model will show high accuracy when applied in our country.

In the scope of research in Vietnam, the paper proposes a model to classify faces wearing or without masks and train on the appropriate data set. This data set provides the closest features to Vietnamese people. Thereby, it is expected that the model can achieve high accuracy while applied in Vietnam.

3. Method

3.1. Deep Learning. Artificial intelligence (AI) is a wide-ranging branch of computer science concerned with building smart machines capable of performing tasks that
typically require human intelligence [25]. Several famous AIs are mentioned, such as Siri, Alexa, and other smart assistants; self-driving cars; conversational bots; e-mail spam filters; and tag recommendations of Facebook.

Machine learning (ML) is a subset of AI that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. In ML, there are different algorithms (e.g., neural networks) that help solve problems.

DL is a subfield of ML concerned with algorithms inspired by the structure and function of the brain called artificial neural networks [26]. It uses multiple layers to progressively extract higher-level features from the raw input. For example, lower layers may identify edges, while higher layers may identify concepts relevant to a human such as digits, letters, or faces. DL algorithms perform a task multiple times to improve the results. These systems help a computer model filter the input data through layers to predict and classify information. DL processes information in the same manner as the human brain. The architectures of the DL network are classified into convolutional neural networks (CNNs), recurrent neural networks, and recursive neural networks [27].

3.2. Artificial Neural Networks (ANNs). In information technology (IT), an ANN is a system of hardware and/or software patterned after the operation of neurons in the human brain. ANNs are a variety of DL technology that also falls under the umbrella of AI [28]. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another. It comprises node layers, containing an input layer, one or more hidden layers, and an output layer.

As we can see in Figure 3, each node or artificial neuron connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, it is activated and sends data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network. Each node has its linear regression model, composed of input data, weights, a bias (or threshold), and an output. The formula is

\[
\sum_{i=1}^{m} w_i x_i + \text{bias} = w_1 x_1 + w_2 x_2 + w_3 x_3 + \text{bias},
\]

where \( w \) is weight, \( X \) is the input data, and the output is

\[
Z = f(x) = \begin{cases} 
1 & \text{if } \sum_{i=1}^{m} w_i x_i + b \geq 0 \\
0 & \text{if } \sum_{i=1}^{m} w_i x_i + b < 0
\end{cases}.
\]

Once an input layer is determined, weights are assigned. These weights help determine the importance of any given variable with larger ones contributing more significantly to the output compared to other inputs. All inputs are then multiplied by their respective weights and then summed. The output is then passed through an activation function, which determines the output. If the output exceeds a given threshold, its fires (or activates) are passed data to the next layer in the network. As a result, its output becomes the input of the next node. This process of passing data from one to the next layer defines this neural network as a feedforward network.

Activation functions are usually nonlinear functions. One of the most widely used activation functions today is the sigmoid function, shown as follows:

\[
\sigma(Z) = \frac{1}{1 + e^{-Z}}.
\]

The loss function is used to evaluate the accuracy. It is a function that allows determining the degree of deviation of the prediction result from the actual value. If the model predicts more errors, the value of the loss function will be larger. Besides, the more correct predictions, the lower the value of the function.

Finally, the model uses a backward propagation algorithm to calculate the gradient of the parameters to find the parameter for the desired neural network. This method traverses the neural network in the reverse direction from the output to the input. The backward propagation algorithm stores the intermediate variables that are the partial derivatives during the gradient computation over the parameters.

3.3. Convolutional Neural Network (CNN). A CNN/ConvNet is a class of deep neural networks [10]. CNNs are made up of neurons with learned weights and biases. It uses a special technique called convolution. It reduces the images into a form that is easier to process without losing features that are critical for getting a good prediction.

Each neuron takes several inputs and performs a matrix multiplication and then passes a nonlinear function. The CNN architecture encodes certain properties of the image of the model. This makes forward propagation more efficient to deploy and greatly reduces the number of parameters in the network. CNNs take as input a one-dimensional vector and transform it through a series of hidden layers. Each hidden layer is made up of a set of neurons and fully connected to all neurons in the previous layer. The neurons in the hidden layer operate completely independently and do not share connections. The final fully connected layer is called the output layer, which represents the probability of the class in the classification problem as shown in Figure 4.

![Figure 3: An example of ANN architecture [11, 12].](image-url)
Unlike a regular neural network (RNN), the layers of a CNN have neurons arranged in three dimensions, namely, width, height, and depth, as shown in Figure 5. For example, the input image in the CIFAR-10 data set is $32 \times 32 \times 3$ [31], and the CNN connects only a small area of the previous layer instead of all neurons like an RNN.

CNN included three main layers, namely, convolutional, pooling, and fully connected layers as shown in Figure 6. These layers are stacked to form the architecture of the CNN. The numerical and ordering arrangement between these classes will create different models suitable for different problems.

The convolutional layer is the most important layer of CNN. This layer is responsible for performing all calculations. Convolutional layers are often used as the first layer to extract features of the input image. The result after convolution of the input image and a filter is a matrix called a feature map. Over many convolutional layers, the characteristic tends to decrease in size (width and height) and increase in depth (depth or channels). This is also one reason that helps CNN work well on image recognition problems. Figure 7 illustrates how the convolution layer works with color images.

We will learn different features of the image with each different filter. In each convolution layer, we will use many filters to learn many attributes of the image. The convolution layer applying $K$ filters will have the output of the layer as a three-dimensional tensor whose size is calculated by

$$\left( \frac{H - F + 2P}{S} + 1 \right) \times \left( \frac{W - F + 2P}{S} + 1 \right) \times K. \quad (4)$$

In feature maps, an activation function is often used to act on all points, and the size of feature maps does not change when passing through the activation function. In the CNN, the most commonly used activation function is the rectified linear unit (ReLU), shown as follows:

$$f(x) = \max(0, x). \quad (5)$$

The pooling layer is often used between convolutional layers in CNN architectures. The function of the pooling layer is to reduce the size of the representation space to reduce the number of parameters and computational requirements of the model. It controls the overfitting learning of the model. The pooling layer works independently on each depth. Assume the kernel size of the composite layer is $K \times K$. The input of the pooling layer is $H \times W \times D$, which is decomposed as a matrix of size $H \times W$. On the area $K \times K$ of the matrix, we find the maximum or the average value of the data and then write it to the output. It has two main types, namely, max and average pooling as shown in Figure 8.

In Figure 9, we can be seen that the fully connected layer works like a normal neural network. The neurons in one layer will connect with all neurons in the next layer. The features of the fully connected layer are extracted by the convolution and pooling layers that are generated as the final result. For the classification problem, the final fully connected layer will use the softmax activation function to give the output classification probability of each class.

The basic structure of a CNN usually includes three main parts.

The local receptive field is responsible for separating and filtering data and image information and selecting the image areas with the highest use value.

The shared weights and bias layer helps minimize the number of parameters that have the main effect of this factor in the CNN network.

The pooling layer is the last and has the effect of simplifying the output information.

LeNet-5 network [29, 30] is the first CNN. It is used for image classification, which is specifically numerical classification. This network was used by several banks at that time to recognize handwritten digits on checks. The input image of the network was a grayscale image with an image resolution of $32 \times 32$ pixels. The network consists of seven layers (two layers (conv + max pooling) and two fully connected layers, and the output is the probability of the softmax function). AlexNet has a similar architecture to LeNet with more layers, filters per layer, and stacked convolutional layers [35]. The network consists of three parts, namely, convolution, max-pooling, and dropout. They are combined with data enhancement techniques (ReLU activation function) and SGD optimization algorithm for output nonlinearity. ZFNet CNN is a network with a top-5 error of 11.7%. This result was achieved by adjusting the hyperparameters of the AlexNet [11, 12] while keeping the architecture of elements constituting a CNN similar to AlexNet with the difference being the filter size at each convolutional layer. GoogleNet/Inception [31] is a generated CNN. It is developed by Google with a top-5 error of 6.67%. This architecture is inspired by the LeNet that has been implemented with a new network constituent element. The network training process uses batch normalization, image distortion, and the RMSprop optimization algorithm. The inception module is made up of convolutions of small size to minimize many network parameters. VGGNet [36] is ranked second with a top-5 error of 7.3% that includes 16 convolutional layers. LeNet in AlexNet uses conv-max-pooling...
Figure 6: CNN structure [32–34].

Figure 7: Performing convolution on color images [29, 30].

Figure 8: Illustrating the operation of the pooling layer [29, 30].
architecture and VGG in the middle and end architecture. This leads to a longer computation time. However, features will be retained than using max pooling after each convolution. Residual neural network (ResNet) [37] is developed by Microsoft. The network model has an error rate of 3.57%. It has a structure similar to VGG with many layers making the model deeper. This network is made up of residual blocks that help solve the problem of vanishing gradients allowing it to be easily trained in hundreds of classes.

In addition to the typical network architectures mentioned above, many other CNN architectures have been researched, developed, and applied to other problems. Convolutional neural architectures are increasingly improving both in terms of the number of parameters as well as the accuracy of the network suitable for specific problems.

3.4. Proposing Face Mask Detection. In the problem of masked face detection, we need to solve two problems. Firstly, we have detected the face in the image/video. Secondly, we detect a mask of a face.

During the analysis and design, the following cases occur:

1. Use a single-stage method for object detection system.
2. Use a two-stage method, one model for detection and one for classification. In single-stage method systems, face detection and classification are performed simultaneously on a model.

If the model cannot detect faces, classification will not be performed. This leads to a lack of face detection. For the two-stage method, we use two separate models for detection and classification. We will have more options with this method. Changing the model at each stage will give solutions to problems that solve different problems.

Much research used other two-stage methods. In our proposed method, the model used for classification is trained on our data set. The data sets are created to meet the conditions of applying in Vietnam.

The initial requirement of the problem is to determine who is not wearing a mask in the input image/video. We use two separate models that perform two independent functions, respectively, Retina Face, and MobileNetV2. Previous publicly available research has had varying accuracy on different data sets. This is easy to understand since each data set contains images with different features. Therefore, it is necessary to choose a data set to train the system’s model to apply in different places.

While designing the system, we realized that when using two separate models for detection and classification, the accuracy of the system will be increased. However, the processing speed of the whole system will decrease compared to the use of one model at the same time. Considering the practicality of the system, this is not a hindrance. The face detection system of people wearing masks operating in real-time toward applying in large public places such as airports, train stations, shopping malls, and so on is not useful. While identifying these subjects, it is difficult to immediately determine where they are in reality and warn or punish them. Therefore, our proposed system is toward a more practical application, which is applying a masked face detection system in offices and classrooms. These are places where the people there have known identities. When the system runs, the supervisor can completely know who is not wearing a mask and remind them. We can let the system run itself and save the face images of the violators and then aggregate the violators and apply penalties to them. Therefore, low processing speed is no longer an obstacle affecting the application of the system in practice.

We choose Retina Face to detect faces and MobileNetV2 to classify masked and nonmasked faces. The proposed system is shown in Figure 10, where Retina Face will take care of face detection, and the area containing the face (ROI) will be cut out. MobileNetV2 will receive the face ROI from the previous step, extract the feature through many layers, and give the final classification result as a mask or nonmask face.

The reason we choose Retina Face to detect faces and MobileNetV2 to classify face-wearing or not wearing a mask will be shown in the next sections. Details of the system will also be presented.

3.5. Face Detector with the Retina Face Model. The authors [6] propose one of the first face detection models based on Cascade-CNN. This method performs simultaneously on multiscale images and removes background areas from low-resolution images. MTCNN [7] is a method including three stages for face detection. In the first stage, the P-Net CNN is used to detect the regions that are likely to contain faces and then combined with the NMS algorithm. In the second stage, all image regions obtained in the first stage will be put into an R-Net CNN to refine, remove the areas with a low probability of containing faces, and merge the areas with a low probability of containing faces. Finally, the O-Net is used to locate the face and its important points. The model based on the region proposal network (RPN) achieved many successes in object detection that is applied to the face detection problem. In [38], the authors propose a supervised transformer network (STN) model for face detection. In the first stage, RPN simultaneously predicts face regions along with facial landmarks. The predicted faces are then normalized by mapping face landmarks to standard positions to better normalize the face samples. In the second stage, R-CNN helps verify valid faces.
The authors [39] apply the faster RCNN model to the face detection problem. They achieved the highest accuracy on two large data sets by training the faster RCNN model on the WIDER FACE data set [40]. This is a testament to the important data in building deep learning models. Unlike two-stage detection models such as RCNN, the SSD model detects faces in one stage from the first layer. In [41], the authors proposed a single-stage headless (SSH) model achieving SOTA with WIDER FACE, FDDB, and Pascal Faces. Instead of relying on image cascades to detect faces with different scales, SSH simultaneously detects faces of size from different layers during forwarding propagation. The authors propose the model single shot scale-invariant face detector (S3FD) [42] to better detect faces of different sizes. Small face detection is a common challenge with anchor-based models. The field of face detection has been studied for many years; one of the biggest challenges is detecting small, tilted, blurred, and partially obscured faces in the real environment. Retina Face is a face detection model launched by Insight Face in May 2019 to address the above challenges. By manually assigning five landmark points on the WIDER FACE data set and using the multitasking loss function at launch, Retina Face was the model with the highest accuracy on the WIDER FACE data set.

As shown in Figure 11, the images detected by the Retina Face model are put through five processing steps: (1) first using MobileNet or ResNet50 to exploit the backbone feature network, (2) then using the FPN (feature pyramid network) and SSH (single-stage headless) to exploit the advanced feature, (3) next using Class Head, Box Head, and Landmark Head networks to obtain prediction results from the feature, (4) finally decode the prediction results, and (5) remove the duplicate detected values through NMS [41]. During the actual training, the model provides two types of backbone networks: MobileNet and ResNet. The Retina Face model uses the ResNet backbone to detect faces with high accuracy and uses the MobileNet backbone to detect faces with faster speed.

FPN: Detecting small-sized faces is a challenge worth solving to improve accuracy. An FPN is a network model designed based on the pyramid concept to solve this challenge. FPN model (Figure 12) combines the information of the model in the bottom-up direction (bottom-up) combined with the top-down direction (top-down) to determine the face position (while other algorithms only often use bottom-up). When the face feature is transitioned from the bottom, the resolution will decrease, but the semantic value will increase.

During the reconstruction from the upper layer to the lower layer, we will consider the loss of information about the faces. For example, a small face when transitioning to the upper layer will disappear, so the model cannot reconstruct that small face when the feature is forwarded from the upper layer to the opposite. To solve this problem, the model creates skip connections between the
reconstruction layers and the feature maps that help the prediction process of face locations perform better than information loss.

The features extracted from the FPN are fed over the single-stage headless (SSH) network to further extract the important features of the face as shown in Figure 13. The class head determines whether the anchor contains a face or not. The box head locates the face. The landmark head locates five key points on the face. For each anchor I, the loss function is calculated according to the following formula:

\[ L = L_{cls}(p_I, p^*_I) + \lambda_1 p^*_I L_{box}(t_I, t^*_I). \]  

(6)

Here, \( L_{cls} \) is a categorical cross-entropy loss function, \( p_I \) is the probability that anchors I is a face predicted by the model, \( p^*_I \) is the actual label of anchor I (\( p^*_I = 1 \) when anchor I is a face and \( p^*_I = 0 \) when it is not facing), \( L_{box} \) is the L1-smooth loss function of the face position and \( t_I = (t_x, t_y, t_w, t_h) \) and \( t^*_I = (t'_x, t'_y, t'_w, t'_h) \) correspond to the coordinates of the face predicted by the model and the actual coordinates assigned by the user.
Since the system proposed in Figure 10 prioritizes high accuracy in the face detection step, the paper uses the Retina Face model with the ResNet backbone. This step is essential. If the face is not detected, the classification of the masked face will not occur. Therefore, we select the Retina Face (ResNet50).

3.6. Face Mask Classification with the MobileNetV2 Model.

In [47], the authors conduct a comprehensive experimental evaluation of several recent face detectors for their performance on masked-face images. Fifteen models were trained and tested on the face mask label data set (FMLD). The data set is the biggest annotated face mask data set with 63,072 face images. The results are shown in Table 2. The average classification accuracy of each model on the FMLD data set together with the prediction speed and the size of the model on disk is arranged in descending order. The processing time is computed over all 12,688 face images and on a per-image basis on an NVIDIA Titan Xp GPU. The prediction accuracy is reported for a bootstrapping protocol with 100 sampled test sets of 5,000 images.

In Table 2, it is easy to see that SqueezeNet v1.1 has the lightest size, fastest speed, and accuracy only 0.8% less than the top model. We need a lightweight model with high speed and acceptable accuracy to compensate for the slow processing speed of the proposed system because of using two models. However, there were several problems with SqueezeNet in our framework. Therefore, we choose MobileNetV2—a model with weight and speed ranked third. This model works perfectly on our device.

The authors [48] describe MobileNetV2 to improve the performance of models on multiple tasks and benchmarks as well as across a spectrum of different model sizes. The great idea behind the MobileNet model is to replace expensive convolutional layers with depthwise separable convolutional blocks. Each block consists of a $3 \times 3$ depthwise convolutional layer that filters the input, followed by a $1 \times 1$ pointwise convolutional layer that combines these filtered values to create new features. It is much faster than the regular convolution with approximately the same result.

MobileNetV1 architecture started with a regular $3 \times 3$ convolution and was followed by 13 depthwise separable convolutional blocks. In MobileNetV2, each block contains...
a 1×1 expansion layer in addition to depthwise and pointwise convolutional layers. The pointwise convolutional layer of V2 is known as a high number of channels in a tensor. The bottleneck residual block is a bottleneck. A 1×1 expansion convolutional layer will expand the number of channels depending on the expansion factor in the data before going into the depthwise convolution. The block of MobileNetV2 is the residual connection. The residual connection exists to help the flow of gradients through the network. Each layer of MobileNetV2 has batch normalization and the ReLU6 as the activation function. However, the output of the paper layer does not have an activation function. The full MobileNetV2 architecture consists of 17 bottleneck residual blocks in a row followed by a regular 1×1 convolution [49], a global average pooling layer, and a classification layer as shown in Table 3.

According to [48, 50], the standard convolution takes an $h_i \times w_i \times d_i$ input tensor $L_i$ that applies the convolutional kernel $K \in R^{k \times k \times d_i \times d_i}$ to produce an $h_i \times w_i \times d_i$. Output tensor $L_j$ will have the computational cost of $w_i \ast d_i \ast d_i \ast k \ast k$. Depthwise separable convolutions are a drop-in replacement for standard convolutional layers. They work almost as well as regular convolutions, but the cost is only $h_i \ast w_i \ast d_i (k^2 + d_i)$, which is the sum of the depthwise and 1×1 pointwise convolutions. Effectively depthwise separable convolutions reduce the computation, compared to standard convolutional layers by almost a factor of $k^2$. MobileNetV2 uses $k = 3 \times 3$ depthwise separable convolution since the computational cost is 8 to 9 times smaller than that of standard convolution with only a small reduction in inaccuracy.

We built a customized fully connected layer that contains four sequential layers on top of the MobileNetV2 model. The layers are as follows:

1. Average pooling layer with 7×7 weights
2. Linear layer with ReLu activation function
3. Dropout layer
4. Linear layer with softmax activation function with the result of two values

<table>
<thead>
<tr>
<th>Model</th>
<th>Model size (MB)</th>
<th>Overall (s)</th>
<th>Per image (ms)</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet152</td>
<td>223</td>
<td>79</td>
<td>6.23</td>
<td>98.93</td>
</tr>
<tr>
<td>DenseNet161</td>
<td>103</td>
<td>76</td>
<td>5.99</td>
<td>98.75</td>
</tr>
<tr>
<td>DenseNet201</td>
<td>71</td>
<td>60 4</td>
<td>.73</td>
<td>98.62</td>
</tr>
<tr>
<td>ResNet101</td>
<td>163</td>
<td>64</td>
<td>5.04</td>
<td>98.55</td>
</tr>
<tr>
<td>DenseNet121</td>
<td>28</td>
<td>51</td>
<td>4.02</td>
<td>98.50</td>
</tr>
<tr>
<td>VGG19</td>
<td>513</td>
<td>62</td>
<td>4.89</td>
<td>98.49</td>
</tr>
<tr>
<td>DenseNet169</td>
<td>49</td>
<td>54</td>
<td>4.26</td>
<td>98.48</td>
</tr>
<tr>
<td>ResNet50</td>
<td>90</td>
<td>51</td>
<td>4.02</td>
<td>98.42</td>
</tr>
<tr>
<td>ResNet34</td>
<td>82</td>
<td>41</td>
<td>3.23</td>
<td>98.28</td>
</tr>
<tr>
<td>ResNet18</td>
<td>43</td>
<td>38</td>
<td>2.99</td>
<td>98.24</td>
</tr>
<tr>
<td>MobileNetV2</td>
<td>8.8</td>
<td>38</td>
<td>2.99</td>
<td>98.19</td>
</tr>
<tr>
<td>SqueezeNet v1.1</td>
<td>2.8</td>
<td>35</td>
<td>2.76</td>
<td>98.09</td>
</tr>
<tr>
<td>SqueezeNet v1.0</td>
<td>2.9</td>
<td>39</td>
<td>3.07</td>
<td>98.06</td>
</tr>
<tr>
<td>GoogleNet</td>
<td>22</td>
<td>38</td>
<td>2.99</td>
<td>98.01</td>
</tr>
<tr>
<td>AlexNet</td>
<td>218</td>
<td>33</td>
<td>2.60</td>
<td>97.81</td>
</tr>
</tbody>
</table>

### Table 3: MobileNetV2 architecture.

<table>
<thead>
<tr>
<th>Input</th>
<th>Operator</th>
<th>$t$</th>
<th>$c$</th>
<th>$n$</th>
<th>$s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>224×224×3</td>
<td>Convolution 2D</td>
<td>—</td>
<td>32</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>112×112×32</td>
<td>Bottleneck</td>
<td>1</td>
<td>16</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>112×112×16</td>
<td>Bottleneck</td>
<td>6</td>
<td>24</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>56×56×24</td>
<td>Bottleneck</td>
<td>6</td>
<td>32</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>28×28×32</td>
<td>Bottleneck</td>
<td>6</td>
<td>64</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>14×14×64</td>
<td>Bottleneck</td>
<td>6</td>
<td>96</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>14×14×96</td>
<td>Bottleneck</td>
<td>6</td>
<td>160</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>7×7×160</td>
<td>Bottleneck</td>
<td>6</td>
<td>320</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7×7×320</td>
<td>Convolution 2D 1×1</td>
<td>—</td>
<td>1,280</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7×7×1280</td>
<td>Average pooling 7×7</td>
<td>—</td>
<td>—</td>
<td>1</td>
<td>—</td>
</tr>
<tr>
<td>1×1×1280</td>
<td>Convolution 2D 1×1</td>
<td>—</td>
<td>k</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>
The final layer softmax function gives the result of two probabilities, each one representing the classification of “mask” or “no mask.” The final classifier architecture was shown in Figure 14.

3.7. Preprocessing Data. In the proposed system, we proceed to create a new data set to train the MobileNetV2 model as shown in Figure 15. Therefore, the Retina Face detector is trained on the WIDER TRAIN data set. It is a face detection benchmark data set, in which images are selected from the publicly available WIDER data set. It has 32,203 images and labels 393,703 faces with a high degree of variability in scale, poses, and occlusion as depicted in the sample images. This data set is large and diverse enough for many different face cases, so we do not intend to improve it further. Our main aim is to create a masked face data set suitable for the classifier.

Since we built the data set ourselves, we faced difficulties when it came to training data, such as different image resolutions and sizes, null values in the data set, unprocessed labels, and so on. We proceed to preprocess the images in our data set to realize their importance.

Preprocessing step is applied to all raw input images to convert them into clean versions that could be fed to a neural network deep learning model.

The preprocessing steps are performed as follows. The images in the data set are divided in a ratio of 6:2:2, corresponding to the training set, validation set, and test set. We resize images to $224 \times 224$ pixel and convert them to array format. They are then converted from BGR to RGB color channels, and the pixel intensities are scaled to the range $[-1, 1]$. Then use scikit-learn One-Hot-Encoding to generate a layered label for each image. In this strategy, each output label value vector is converted to a new form, where only 1 output equals “1” corresponding to the classification code of the corresponding input vector, and the other outputs are all equal to “0.” Finally, we convert the images into NumPy arrays. This step is used not only to preprocess the input data to train the model but also for the proposed system’s input images/frames.
4. Simulation and Results

4.1. Data Set. There are many face mask face data sets. Several famous data sets can be mentioned as face mask label data sets (FMLD). The data set is the biggest annotated face mask data set with 63,072 face images. Labeled faces in the wild (LFW) data sets are 13,233 images that are collected from 5,749 people. The real-world masked face data set (RMFD) is large for masking, including 5,000 images wearing masks and 90,000 ones without masks of 525 different people as shown in Figures 16 and 17.

However, none of these data sets are suitable for Vietnamese people. Although the models after training on these data sets can still be applied for our application, they are not completely suitable. Therefore, we built the data set ourselves.

The authors [51] use a dlib-based face landmark detector to identify the face tilt and six key features of the face necessary for applying the mask. Based on the face tilt, the corresponding mask template is selected from the library of a mask. The template mask is then transformed based on the six key features to fit perfectly to the face. The complete block diagram can be seen below. The system provides several masks to select from. It is difficult to collect mask data sets under various conditions. It can be used to convert any existing face data set to a masked-face data set. It identifies all faces within an image and applies the user-selected masks to them considering various limitations such as face angle, mask fit, lighting conditions, and so on. A single image or an entire directory of images can be used as input to code.

Inspired by [51], we applied it to create our masked data set. We collected the Asian face age data set (AFAD) [52]
including 164,432 well-labeled images of Asians. The faces in AFAD were passed through the MaskTheFace program to obtain different masked faces. In the paper, we use four types of masks that are popular in our country surgical, cloth, N95, and K95. Since our model is trained on our laptop, the amount of data that can be used without being out of memory is 8,000 images, 5,000 masked face images, and 3,000 nonmasked face images. In the data set, there are 1,500 images taken from the RMFRD data set to increase the diversity of the data set as shown in Figure 18.

Our model is trained on our laptop, and the amount of data that can be used without being out of memory is 8,000 images, 5,000 masked face images, and 3,000 nonmasked face images. In 5,000 masked face images, 3,500 images use a medical mask with the colors white, blue, gray, and black evenly divided. N95, K95, and cloth each are used for 500 images with a basic color. If using 100% of the images generated from the MaskTheFace simulator, the data set will be monotonous, lacking realism and diversity. Realizing this problem, the paper took more than 1,500 images from the RMFRD and other data sets to increase the diversity of the data set as shown in Figure 19.

Besides, our data set (in Figure 20) is not large enough to change all weights of the model; we use the transfer learning method to train our model—fine-tuning. It keeps useful weights and changes the weight of several layers of a pre-trained model to conform to the target of the paper.

4.2. Setup. In this paper, we use our laptop to build the proposed system. The information on our hardware and software consists of processor Intel® Core™ i7-8750H, Ubuntu 18.04.6 LTS, Nvidia GeForce GTX 1050Ti, Cuda 11.2 and Cudnn 8.1, Tensorflow 2.5.0, and Keras 2.4.3.

Since our data set is not large enough to change all weights of the model, we use the transfer learning method to train our model—fine-tuning. It keeps useful weights and changes the weight of several layers of a pretrained model to conform to the target of the paper.

4.3. Evaluation Method. When building a masked face detection system it needs an evaluation method to be effective for the system, to provide a conclusion and way to improve the system. There are many ways to evaluate, depending on different problems to choose the suitable method. The following method is used in our paper: ACC, TP, TN, FP, and FN. Mask is considered a positive class (P), and no mask is considered a negative class (N). Each video is considered a data point as follows:

TP (true positive): an outcome where the model correctly predicts the positive class. Specifically, the correct prediction number is that the masked face image is predicted as masked.

TN (true negative): an outcome where the model correctly predicts the negative class. Specifically, the
correct prediction number is that the no-mask image is predicted as not masked.

FP (false positive): an outcome where the model incorrectly predicts the positive class. Specifically, the correct prediction number is that no mask image is predicted as a mask.

FN (false negative): an outcome where the model incorrectly predicts the negative class. Specifically, the correct prediction number is that the mask image is predicted as no mask.

ACC (accuracy): the ratio between the number of correct predictions and the total number of data points as follows:

\[
ACC = \frac{TP + TN}{TP + TN + FP + FN}. \tag{7}
\]

4.4. Results. The whole process of training the MobileNetV2 model occurred in 1 hour and 6 minutes with the computation time per epoch being 75 seconds. This speed is fast since the amount of training data is not too large—8,000 face images. However, one of the important things that determine the fast training time with high accuracy is fine-tuning. This training method “inherits” a trained network on a very large data set of generic images to create a specialized model for a more specific task, specifically in this problem distinguishing between wearing and not wearing masks. Thanks to that, it did not take us too long to do the training.

The model was trained on Nvidia GeForce GTX 1050Ti. To perform model training, the selection of parameters is also essential. We set up epoch = 40, batch size = 32, and learning rate = 0.00001. These are the parameters that after many tests, the model after training achieves the highest accuracy.
Figure 22: Result of (a) accuracy and (b) loss of a model.

Table 4: Comparing with the system [9].

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy (%)</th>
<th>Frame rate (fps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>YOLO version 4 [53]</td>
<td>95.00</td>
<td>—</td>
</tr>
<tr>
<td>YOLO version 4 [54]</td>
<td>98.00</td>
<td>49.5</td>
</tr>
<tr>
<td>MTCNN + SRNet [9]</td>
<td>98.7</td>
<td>10</td>
</tr>
<tr>
<td>ResNet50 [55]</td>
<td>98.2</td>
<td>20</td>
</tr>
<tr>
<td>Proposal system, Retina Face + MobileNetV2</td>
<td>99.37</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 5: Final classifier model architecture.

<table>
<thead>
<tr>
<th>Type</th>
<th>Output shape</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average pooling 2D</td>
<td>(1, 1, 1,280)</td>
<td>0</td>
</tr>
<tr>
<td>Flatten</td>
<td>(None, 1,280)</td>
<td>0</td>
</tr>
<tr>
<td>Dense</td>
<td>(None, 128)</td>
<td>163,968</td>
</tr>
<tr>
<td>Dropout</td>
<td>(None, 128)</td>
<td>0</td>
</tr>
<tr>
<td>Dense</td>
<td>(None, 2)</td>
<td>258</td>
</tr>
</tbody>
</table>

Figure 23: Result on webcam video: (a) with mask and (b) without mask.
To evaluate the accuracy of the masked face detection method that the paper is using, different methods are compared. The accuracy of our model is 99.37%. In Figure 21, easy to see that the model works very well with different mask types. When we try it in webcam video on our laptop, it is working perfectly with a rate is 3 fps. The results after the models are trained are shown in Figure 22.

We compared the proposed accuracy and processing speed of the proposal with other systems [9]. The accuracy comparison table is not intuitive since the models are tested on different test data sets. Table 4 shows that the proposed system has impressive accuracy although the speed is not high. This is understandable since the proposed method uses two independent CNN models: The Retina Face model for face detection and MobileNetV2 for classification. However, this does not affect the practical applicability of our system. Instead of having the system run in real time, we can let it run automatically. When masks do not appear, the system crops and saves based on these images to find the violator and sanction. This is entirely possible since the applicable locations are schools, offices, and so on, where people can be controlled.

Table 5 shows the details of the layers that we added in terms of output shape and parameters. According to the table, the number of additional parameters of the model is not large (164,226 parameters), combined with the pre-trained model MobileNetV2 on ImageNet has a total of 3.4 million parameters [48]. Compared with current CNN models, this can be considered one of the models with very few parameters. Therefore, the model has a fast processing speed in the classification stage. Therefore, we can see that the processing speed of the whole system is not high because of the face detection stage by Retina Face. However, we get high accuracy in the face detection step.

Besides, the method can be applied to other problems, for example, thermal imaging [56–58]. The ventilation hole of the angle grinder will be considered as a mask, and the algorithm needs to build a data set about this system. When we design the full data set, the algorithm can completely determine the exact ventilation hole.

Several experimental results of the masked face detection system are shown in Figures 22–24. As we can be seen in Figures 24 and 22, the masked face detection proposed by the paper works well for different types of masks. In Figure 24, the system works excellently when there are many faces in the image. The Retina Face detection model works fine. The undetectable faces are all hidden faces larger than 80%. Figure 23 shows the result when we run our masked face detection system on a webcam video.

In this section, we present the face mask data sets and built them for the proposed system. The evaluation results are then presented and combined with other methods. Finally, we show the output images.

5. Conclusion

The paper focuses on studying the use of CNN to detect masked faces. The paper presented network models and deep learning algorithms for face detection and classification based on face Asian. The system can detect and classify face that is wearing or not wearing the mask with up to 99.37% accuracy with Retina Face detection with ResNet50 backbone and face classification with MobileNetV2, and the frame rate is 3 fps. The rate is at an acceptable level. It does not affect the practical application of the proposed system. In the context of preparing to normalize activities, the model is proposed to be applied in schools and office places with fixed
identities to control people to follow the rule of wearing masks. Recording faces labeled as not wearing a mask can replace real-time applications. The system achieves high accuracy. However, it is not optimal for blurred faces. Since the data set is not large enough to train the model, we use more samples of blurred and complex face angles.

Although the proposed system received quite positive results, there were still a few test cases that gave bad results. For example, the face that is included in the classification is blurred due to the camera lighting conditions or face tilt angles being too large. Therefore, we plan to extend the data set to train the classifier model of the system on more powerful devices in the future. Furthermore, we will improve the system to achieve a higher FPS processing speed while maintaining the same accuracy. We will focus the research to optimize the execution time of the system that is suitable for embedded systems. Therefore, it will increase the practicality to be able to apply in practice.

Data Availability
There are many face mask face data sets. Several famous data sets can be mentioned as face mask label data sets (FMLD). The data sets are in [31] and [32].

Conflicts of Interest
The authors declare that they have no conflicts of interest.

Acknowledgments
This research was funded by Vietnam National University Ho Chi Minh City (VNU-HCM) under grant number NCM2021-20-02.

References


