DESIGNING A FACE RECOGNITION SYSTEM WEARING A MASK

Hai Ninh Nguyen Thi[1], Anh Dang Thi Ngoc[1]

[1]Thai Nguyen University of Technology, Thai Nguyen city, Viet Nam

ABSTRACT

This article presents a solution to control and identify people entering or leaving offices, commercial or entertainment areas wearing masks or not. It uses Visual Studio Code software and is embedded in Raspberry Pi 3.. Through the testing process, the system has met the basic requirements of the problem and monitored and warned whether people entering or leaving are wearing masks. The system has been installed and tested in the laboratory of the Department of Electronic Engineering - Faculty of Electronics - University of Industrial Engineering to demonstrate the applicability of the product..

Keyword: *Face recognition, Rasberry Pi 3, computer vision*

1. INTRODUCTION

Recently, a study exploring measures to tackle the COVID-19 pandemic revealed that wearing a mask or other covering the nose and mouth reduces the risk of spreading the Coronavirus. The findings suggest that a near-maximum adoption of masks (80%) could prevent 17–45% of deaths and reduce the peak daily mortality rate to 34–58%. This result specifically recommends the use of masks in general to limit the spread of Coronavirus. Furthermore, with the reopening of countries, Governments and Public Health Authorities are recommending masks as essential measures to stay safe when venturing out in public.

In this article, the authors focus on researching and designing a face recognition system to recognize wearing medical masks using Raspberry Pi 3. The article is divided into 6 parts: (1) Introduction; (2) design the problem of face mask recognition from the face recognition system; (3) CNN algorithm, Keras library and openCV; (4) Hardware and software design; (5) Results; (6) Conclusion.

2. DESIGN THE PROBLEM OF FACE MASK RECOGNITION FROM THE FACE RECOGNITION SYSTEM

In this problem, we design a system to detect people without masks based on the CNN model. Specifically, the program will give a direct warning to remind people not wearing a mask in public places by voice combined with sending information of violators to the monitoring agency.

To design the problem, we have 3 steps:

Step 1: Collecting program data here we use Python language (high-level programming language), use OpenCV open source software library to detect human faces. The data after being collected in the form of an image file (JPG) will be stored in two separate files, including: one file contains 400-500 pictures depicting the face wearing a mask, the other file contains 400-500 pictures describing the mask. Describe a face without a mask.

Step 2: Use the data source collected in step 1 to analyze based on the CNN model. At this stage, pre-data processing aims to bring all images to the same size, then these images will be converted to serve the image processing in the next step. Based on the CNN model, the convolutional neurons are specially designed to process the most important elements in the image to produce accurate data results.

Step 3: Detect the person wearing a mask or not. This step will conduct comparative analysis of the data extracted from the camera (after input data has been processed) with the analyzed data results for voice warnings. Based on the results obtained from step 2, the data will be displayed on the screen showing whether people are wearing masks or not. If the person is not wearing a mask, it will be immediately reminded through direct speech.

3. CNN Algorithms, KERAS AND OPENCV LIBRARY

3.1 CNN Algorithms

In neural networks, convolutional neural networks (ConvNets or CNNs, Convolutional Neural Networks) is one of the main methods to perform image recognition and image classification. CNN is widely used in a number of fields such as object detection, face recognition.
A CNN network is a collection of Convolution layers that overlap and use nonlinear activation functions like ReLU and tanh to activate the weights in the nodes. Each class after passing activation functions will generate more abstract information for the next classes.
Each class after passing activation functions will generate more abstract information for the next classes. In the feedforward neural network model, each input neuron (input node) for each output neuron in subsequent layers.
This model is called fully connected layer or fully connected layer. In the CNNs model, the opposite is true. The layers are linked together through the convolution mechanism.
The next layer is the convolution result from the previous layer, so we have local connections. Thus, each neuron in the next layer is generated from the result of a filter applied to a local image region of the previous neuron.
Each class that uses different filters usually takes hundreds of thousands of such filters and combines their results. In addition, there are some other layers such as pooling/subsampling layer used to filter out more useful information (remove noise information).
During the training of the network, the CNN automatically learns the values through the filter classes based on how you do it. For example, in the image classification task, CNNs will try to find the optimal parameters for the corresponding filters in the order raw pixels > edges > shapes > facial > high-level features. The last layer is used to classify the image.

3.2 Keras Library

The Keras library is an open source mainnet written in the Python language. It is a library developed in 2005 by Francois Chollet, a Deep Learning research engineer. Keras can be used with famous libraries such as Tensorflow, CNTK, Theano.
Keras runs on open source machine libraries such as TensorFlow, Theano, or the Cognitive Toolkit (CNTK). Theano is a python library used for fast numerical computation tasks. TensorFlow is the most famous symbolic math library used to create neural networks and deep learning models. TensorFlow is very flexible and the main benefit is distributed computing. CNTK is a deep learning framework developed by Microsoft. It uses libraries like Python, C#, C++ or standalone machine learning toolkits. Theano and TensorFlow are very powerful but confusing libraries for creating neural networks.

3.3 OpenCV Library

OpenCV is an abbreviation for open source computer vision library - can be understood as an open source library for computers. More specifically, OpenCV is an open source repository for image processing and real-time development of graphical applications [3].
OpenCV allows to improve CPU speed when performing real-time operations. It also provides a large amount of processing code for the process of computer vision or other machine learning.
The OpenCV library is released with a BDS license. Therefore the services it provides are completely free of charge and are limited to the usual barriers. In particular, you are authorized to use this software for both commercial and non-commercial activities. OpenCV owns a friendly interface with all kinds of programming languages, such as C++, C, Python or Java… In addition, it is also easily compatible with different operating systems, including from Windows, Linux, Mac OS, iOS to even Android.

4. DESIGN HARDWARE AND SOFTWARE

4.1 Design hardware

The control system includes the following main parts: Raspberry Pi 3 microcontroller block, Raspberry Pi V1 OV5647 5MP Camera, display block with hardware structure diagram as shown in Figure 1.

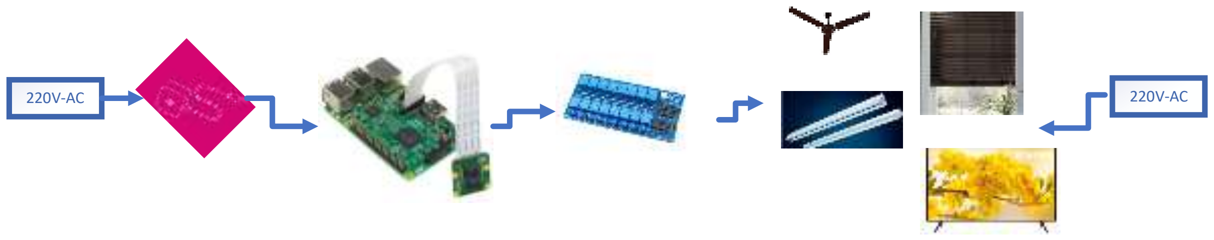


Fig -1: Hardware diagram of the system

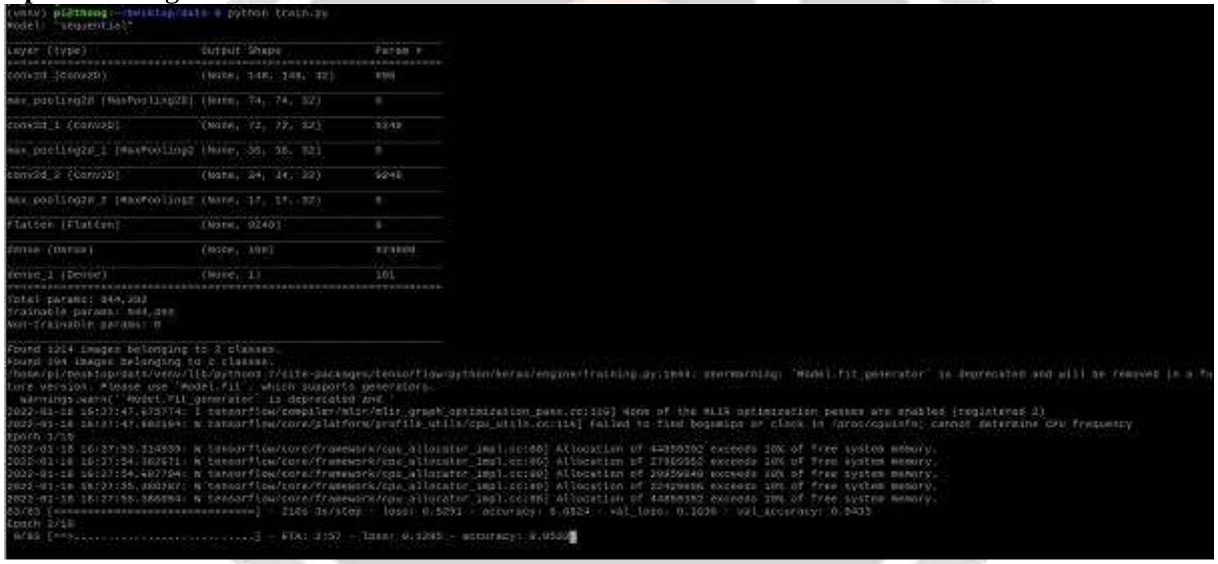
4.2 Design software

To connect Raspberry Pi 3 to the computer, the author team used PuTTy software to enable functions such as: SSH, camera, VNC...; and VNC to control raspberry pi on linux operating system.

Then using Visual Studio Code is because of many outstanding advantages such as: Multi-platform support: Windows, Linux, Mac; Multi-language support: C/C++, C#, F#, Visual Basic, HTML, CSS, JavaScript, JSON; Less capacity; Powerful features; Professional Intellisense; Friendly interface. And above all, Visual Studio Code is free software, used by many programmers around the world.

Implementation steps:

Step 1: Training data



```
(venv) pi@thong:~/Desktop/data $ python train.py
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 32)        9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 9248)              0
_________________________________________________________________
dense (Dense)                (None, 100)               924900
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101
=================================================================
Total params: 944,393
Trainable params: 944,393
Non-trainable params: 0
_________________________________________________________________
Found 1214 images belonging to 2 classes.
Found 194 images belonging to 2 classes.
/home/pi/Desktop/data/venv/lib/python3.7/site-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: `Model.fit_generator` is deprecated and will be removed in a fu
ture version. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
2022-01-18 16:27:47.675774: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of the MLIR optimization passes are enabled (registered 2)
2022-01-18 16:27:47.682164: W tensorflow/core/platform/profile_utils/cpu_utils.cc:116] Failed to find bogomips or clock in /proc/cpuinfo; cannot determine CPU frequency
Epoch 1/10
2022-01-18 16:27:53.314939: W tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of 44859392 exceeds 10% of free system memory.
2022-01-18 16:27:54.382671: W tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of 27959552 exceeds 10% of free system memory.
2022-01-18 16:27:54.407794: W tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of 29959840 exceeds 10% of free system memory.
2022-01-18 16:27:55.380267: W tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of 22429696 exceeds 10% of free system memory.
2022-01-18 16:27:56.386894: W tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of 44859392 exceeds 10% of free system memory.
63/63 [==============================] - 216s 3s/step - loss: 0.5291 - accuracy: 0.6824 - val_loss: 0.1038 - val_accuracy: 0.9433
Epoch 2/10
 4/63 [=>............................] - ETA: 2:57 - loss: 0.1285 - accuracy: 0.9500
```

Fig -2: Training Data

After the training is complete, a file will appear as file_train.h5.

Step 2: Reload trained data file_train.h5

Step 3: Detect face with x,y coordinates as well as width and height of face containing it with opencv and save as a temporary image temp.jpg

Step 4: Convert the saved image to tensorflow array.

Step 5: Make predictions displayed on the screen interface, faces without masks will be saved to the save folder.

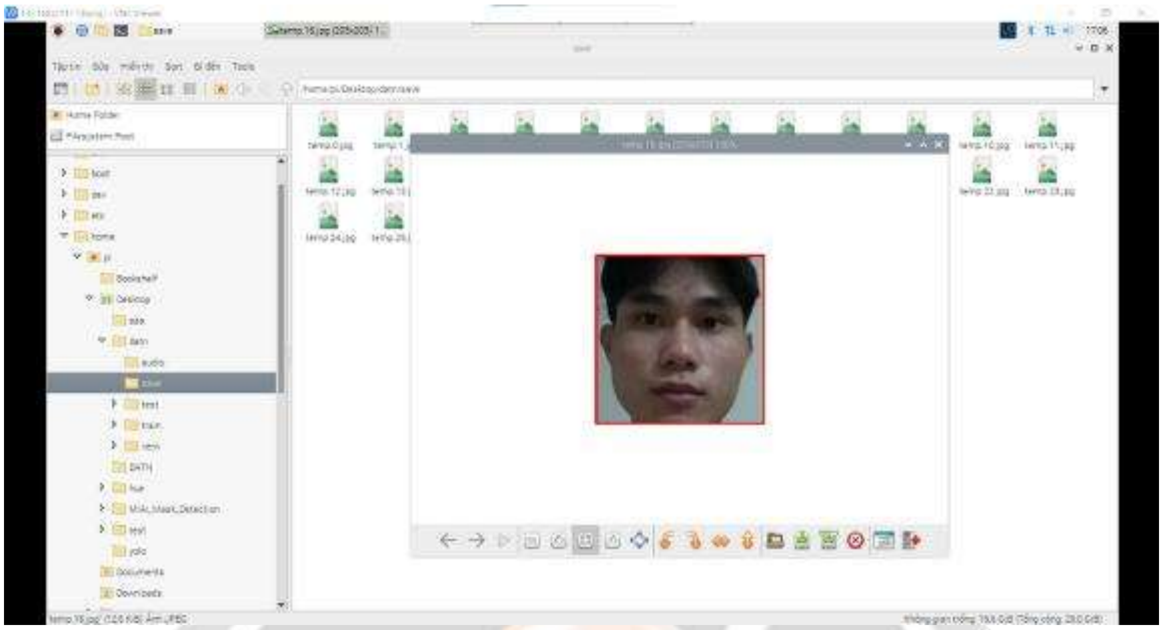


Fig -3: Face recognition image without a mask and saved in the save folder

5. RESULT

After performing many tests, the system showed that the correct identification results were achieved. And the system has also issued warnings to people who are not wearing masks. In addition to the main components designed in part 4 of the article, the system also has a source circuit, a Role and an actuator block.

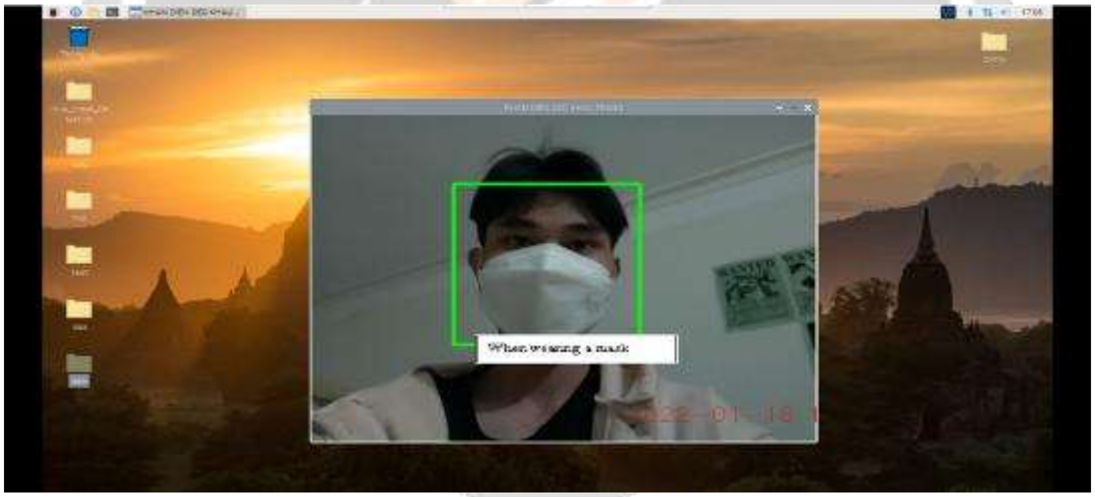


Fig -4: When wearing a mask

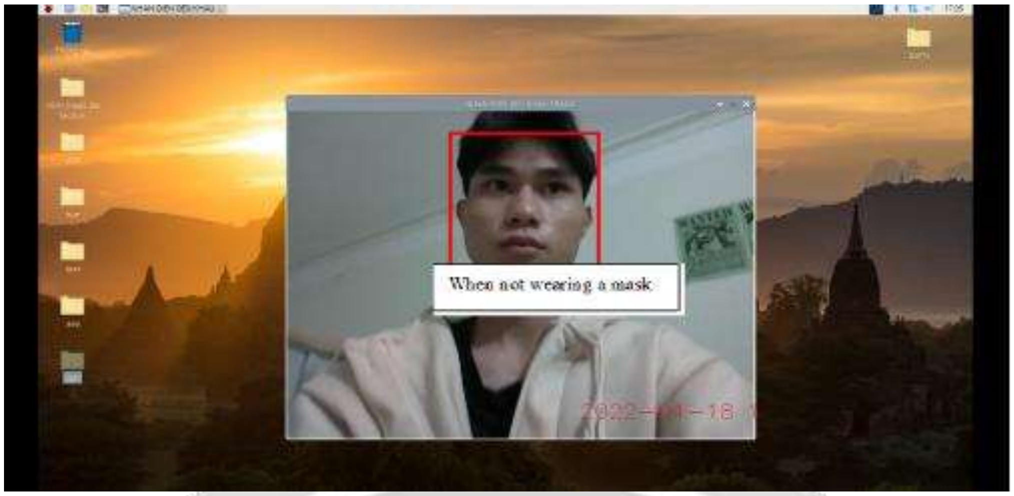


Fig -5: When not wearing a mask

5. CONCLUSIONS

Given the current situation and danger of the SAR-COVID-2 epidemic in particular and respiratory infectious diseases in general, we have seen that wearing a mask is essential. And the face recognition system that recognizes wearing a mask has met the requirements of monitoring and warning mask wearers. With advantages such as: simplicity, fast processing speed, no high processing requirements, accurate identification of mask wearers with many different angles, clear audible warning, recognition of many molds. face at the same time. The system has been tested in the laboratory of Electronic Engineering Department - Faculty of Electronics - Thai Nguyen University of Industrial Technology for students to study and practice. In the coming time, the authors will conduct further improve the ability to recognize non-masked faces not only at the alert level, but also recognize the identity of the person not wearing a mask..

6. ACKNOWLEDGEMENT

The authors thanks for the financial support of Thai Nguyen University of Technology for the research team to design, implement and complete the product.

7. REFERENCES

[1]. Sikender Mohsienuddin Mohammad, "Facial recognition technology" *SSRN Electronic Journal.*
[2]. Hagyeong Leea, Jongwoo Song, "Introduction to convolutional neural network using Keras; an understanding from a statistician" *Journal of Communications for Statistical Applications and Methods*
[3]. Joseph Howse, " OpenCV Computer Vision with Python" Packt Publishing Ltd, 2013
[4]. Jianxin Wu, " Introduction to Convolutional Neural Networks," *2017.*
[5] "Raspberry Pi 3 Model B+", https://static.raspberrypi.org/