Face Mask Detection

COMP4211 Project Report

NG Chun Fung
cfngai@connect.ust.hk
20696453

1. Introduction

During the Covid-19 pandemic, wearing masks has become the must for going outdoor in Hong Kong. According to the Prevention and Control of Disease Regulation [1], citizens are required to wear mask in majority public area, or else need to pay a fixed amount of penalty. Hence, resources are invested in monitoring people whether they are wearing masks or not.

To facilitate the monitoring, this project aims to implement a model which can classify whether peoples are wearing masks or not. Face recognition will first be done to extract the faces part in the image to eliminate all unnecessary components in the image. Then, a Convolutional Neural Networks will be constructed and trained with the dataset provided. Classifier at the end will predict the label of each test sample. The project will evaluate the model with different hyperparameter and methods to achieve the best accuracy.

2. Dataset and data pre-processing

In the dataset, it provided 686 non-masked face images, and 690 masked face images. Before processing into the model training stage, the raw data needed to be pre-processed to prevent any other factors, including background or objects, to interfere with the feature selections. Face recognition is done for extracting the faces from the raw images.

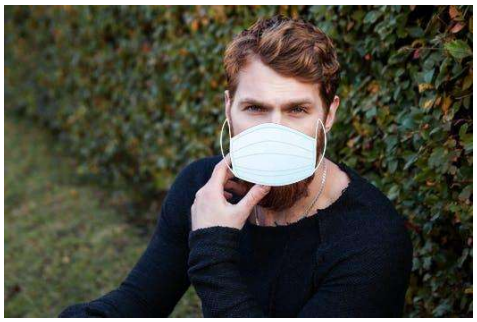



Fig.1a and 1b, Images have different background

The first attempt is using the OpenCV with Haarcascade model [2] to extract the

face from the sample. However, after tuning the hyperparameters and using different models, the results are not accurate and most of the time unrelated pixels are extracted. Considering the low accuracy, this attempt has been considered as failure.

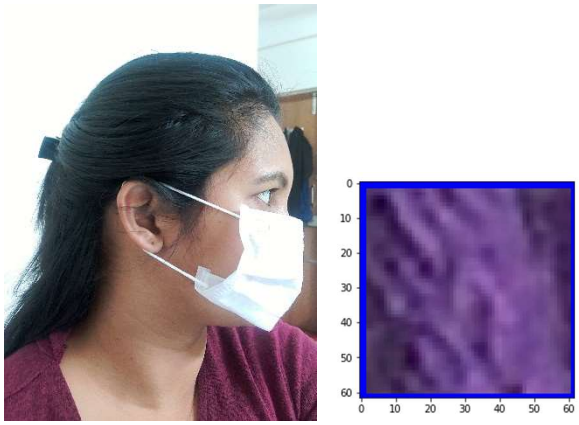


Fig.2a and 2b, Left shown \dataset\1\pra3.jpg, right shown face extracted

Second attempt is using the MediaPipe model [3] to extract the faces from sample. Different from the previous attempt, the model accurately detects the face regions using 3D facial keypoint estimation, facial features or expression classification, and face region segmentation. There are still images that cannot be extracted by the models. However, these cases will be minority, and hence the original photo will be processed as cropped face.

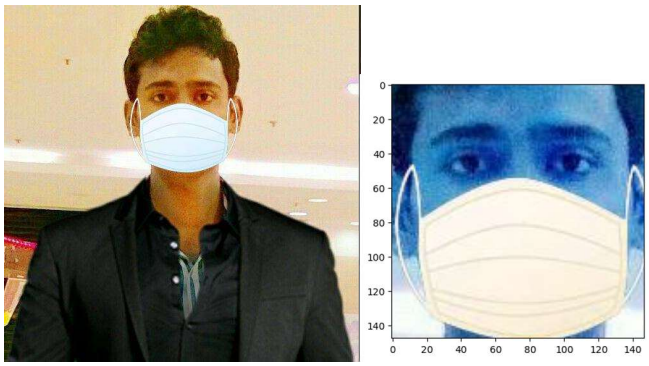


Fig.3a and 3b, Left shown \dataset\1\ 21-with-mask.jpg, right shown face extracted

After cropping out the faces from the raw dataset, they are stored into three subfolders (train, valid, and test folder) in '/data' folder. 80% of the data will be used in training and 10% will be used in validation. After training the model, the remaining 10% of the data will be used for testing the model and evaluate the accuracy of the model trained.

3. Machine learning tasks

During the data preprocessing stage, MediaPipe is used aforementioned. The machine learning pipelines in MediaPipe Face Mesh consists of two real-time deep

neural network models. One is computing the face location and the other is predicting the approximate 3D surface landmark via regression. It can infer the 3D facial surface with a 2D image by estimating 468 3D face landmarks.

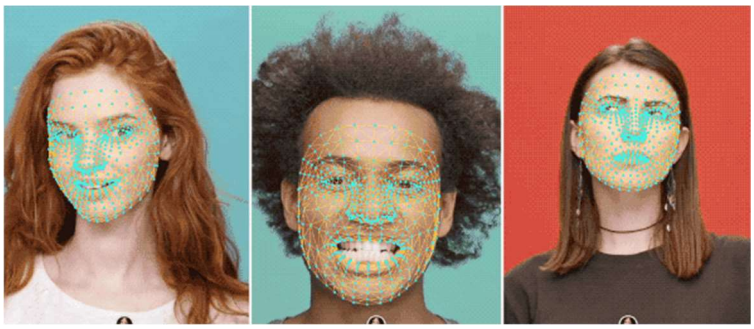
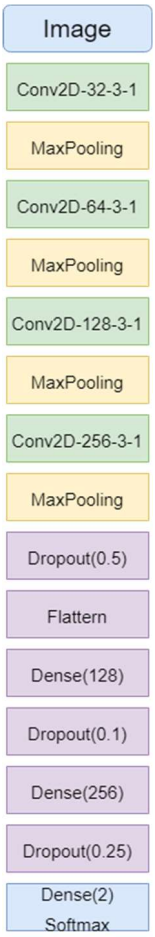


Fig.4, Face landmarks indicated

For the model used in training, a sequential CNN model is constructed, with Conv2D layer, MaxPooling Layer, Flattern Layer, Dropout Layer, and Dense Layer. The architecture is shown as the following:



The last layer will be a softmax function which output the two probabilities corresponding to the two classes.

For the dataset part, in light of the small dataset, ImageDataGenerator has been used for generating batches of tensor image data with real-time data augmentation.

[4] Three generator has been created. Training generator and validation generator has been used in training process, and the test generator is used to predict and evaluate the model after so. Images are also resized to 150x150 before training the model.

```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale=1.0/255,
                                   rotation_range=40,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   fill_mode='nearest')
train_generator = train_datagen.flow_from_directory('./data/train',
                                                    batch_size=10,
                                                    target_size=(150, 150))

valid_datagen = ImageDataGenerator(rescale=1.0/255)
valid_generator = valid_datagen.flow_from_directory('./data/valid',
                                                    batch_size=10,
                                                    target_size=(150, 150))

test_datagen = ImageDataGenerator(rescale=1.0/255)
test_generator = test_datagen.flow_from_directory('./data/test',
                                                  batch_size=10,
                                                  target_size=(150, 150))
```

For the training section, Adam optimizer and the sparse categorical crossentropy loss function has been used. Epoch has been set to 10 to prevent overfitting while attaining a good accuracy.

4. Machine learning methods used for solving the task

Tensorflow and keras are used in building and training the recognition model. Google collaboration is used during the training process with the GPU provided.

Throughout the whole project, sequential CNN from keras is used for training the model. Each convolutional layer contains a set of filters, and the output represents a feature. After passing the images into the model during the training process, parameters will be updated in each filter. As a result, features will be extracted from the image after each layer of convolutional map. Each pooling layer does not contain any learnable parameters, it is used to reduce the number of trainable parameters for the next layers. In the project, max pooling layer is used for finding the maximum value in each patch. Dropout layer is used for avoiding overfitting in the network. The last softmax function in the model of this project is used to calculate a vector of probability. By comparing the higher probability, the prediction from the model can be computed.

About the setting, the epoch is chosen to be 10 in fitting the training data to avoid overfitting. For the data, ImageDataGenerator is chosen to be used because of the small data set. With data augmentation, it can stimulate the real-world problem where the images might not always be well taken, and hence, increase the accuracy

when it is applied in daily uses. All images are resized to 150x150 for computation.

5. Experiments and results

For data pre-processing, as aforementioned, using OpenCV will crop out error region which are not faces. After choosing the MediaPipe model, the accuracy of face detection with mask is high. The following is the comparison between the cropped face images with masked and the raw data.

Fig.5a, cropped result of masked face images

Fig.5b, raw data of masked face images

It is obvious that most of the pictures are well cropped, and there is no false cropping of other objects. Hence, it will be good enough for proceeding to the model training part.

For the model building section, the architecture of the model is constructed in the following way.

```
model = Sequential([
        Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)),
        MaxPooling2D(2,2),
        Conv2D(64, (3,3), activation='relu'),
        MaxPooling2D(2,2),
        Conv2D(128, (3,3), activation='relu'),
        MaxPooling2D(2,2),
        Conv2D(256, (3,3), activation='relu'),
        MaxPooling2D(2,2),

        Dropout(0.5),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.1),
        Dense(256, activation='relu'),
        Dropout(0.25),
        Dense(2, activation='softmax')
])
```

With such model architecture, the trainable parameters are 2,027,714. The accuracy of the model is high during the training process. The result of training is as shown as the following. The accuracy of validation set is higher than 0.9, and the loss value is around 0.1.

```
Epoch 1/10
111/111 [==============================] - ETA: 0s - loss: 0.3064 - acc: 0.8757WARNING:absl:Found untraced functions such as
111/111 [==============================] - 10s 87ms/step - loss: 0.3064 - acc: 0.8757 - val_loss: 0.0369 - val_acc: 0.9927
Epoch 2/10
111/111 [==============================] - ETA: 0s - loss: 0.1576 - acc: 0.9555WARNING:absl:Found untraced functions such as
111/111 [==============================] - 10s 90ms/step - loss: 0.1576 - acc: 0.9555 - val_loss: 0.0129 - val_acc: 1.0000
Epoch 3/10
111/111 [==============================] - 9s 77ms/step - loss: 0.0976 - acc: 0.9691 - val_loss: 0.0210 - val_acc: 0.9927
Epoch 4/10
111/111 [==============================] - 8s 74ms/step - loss: 0.1572 - acc: 0.9474 - val_loss: 0.0395 - val_acc: 0.9854
Epoch 5/10
111/111 [==============================] - 8s 74ms/step - loss: 0.1461 - acc: 0.9519 - val_loss: 0.0339 - val_acc: 0.9854
Epoch 6/10
111/111 [==============================] - 8s 74ms/step - loss: 0.0976 - acc: 0.9701 - val_loss: 0.0355 - val_acc: 0.9927
Epoch 7/10
111/111 [==============================] - ETA: 0s - loss: 0.0972 - acc: 0.9637WARNING:absl:Found untraced functions such as
111/111 [==============================] - 10s 87ms/step - loss: 0.0972 - acc: 0.9637 - val_loss: 0.0024 - val_acc: 1.0000
Epoch 8/10
111/111 [==============================] - 8s 74ms/step - loss: 0.0920 - acc: 0.9737 - val_loss: 0.0062 - val_acc: 1.0000
Epoch 9/10
111/111 [==============================] - 8s 75ms/step - loss: 0.0544 - acc: 0.9873 - val_loss: 1.0797 - val_acc: 0.8175
Epoch 10/10
111/111 [==============================] - 8s 76ms/step - loss: 0.1514 - acc: 0.9583 - val_loss: 0.0025 - val_acc: 1.0000
```

Moreover, using the test set to evaluate our final trained model, it can attain 97% of accuracy. The confusion matrix is shown as following. There are only few false positive and false negative, and majority are true positive and true negative.

6. Conclusion

With all the training and evaluation, the model is well-trained with high accuracy. Its usage can be further extended to daily life, including alerting customers if they are not wearing masks.

Reference

[1] https://www.elegislation.gov.hk/hk/cap599I!en
[2] https://github.com/opencv/opencv/tree/master/data/haarcascades
[3] https://google.github.io/mediapipe/solutions/face_detection.html
[4] https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator