# FACE MASK RECOGNITION WITH CONVOLUTION NEURAL NETWORK

CHETSADA CHONGTHANACHOTE

AN INDEPENDENT STUDY SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN
DATA SCIENCE AND ANALYTICS
KMITL-DIGITAL ANALYTICS AND INTELLIGENCE CENTER SCHOOL OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2022
KMITL- 2022-SC-M-017-107

| | |
|---|---|
| Independent Study Title | Face mask recognition with convolution neural network methods |
| Student Name | Chetsada Chongthanachote |
| Student ID | 63605070 |
| Degree | Master of Science (Data Science and Analytics) |
| | KMITL-Digital Analytics and Intelligence Center |
| Year | 2022 |
| Independent Study Advisor | Dr.Yuwadee Klomwises |

## ABSTRACT

Nowadays, the practice of verifying the identity of a person or knowing your customer is widespread and legally applicable. This makes entrepreneurs have to deal with the application that supports their customers. Using AI technology to check a customer's face is one of the most common ways to identify them. As a result, this will reduce the work time of employees. By 2021, facial recognition will be in use at the top 20 U.S. airports for 100% of international passengers, including American citizens. This is according to an executive order from Runaway Suitcase. In this research, we put emphasis on procedure for distinguishing between people who wore masks did not wear masks. There is preparing process for overlaying mask to face image before bringing to the classification mode. We discovered that self-prepared datasets for images of persons wearing masks can be utilized as an input to the classification model and achieve the desired result, However the VGG-16 model can identify people both wearing masks and not wearing masks effective than ResNet50 model.

# ACKNOWLEDGMENT

Foremost, I would like to express my sincere gratitude to my advisor Dr. Yuwadee Klomwise for the continuous support of my master's degree study and research, for her patience, motivation and enthusiasm knowledge. Her guidance helped me in all the time of research and writing successfully.

Besides my advisor, I would like to thank the rest of my thesis committee: Asst. Prof. Dr Kulsawasd Jitkajomwanich, Dr. Jiraphat Yokrattanasak or their encouragement, comments, and direction.

I am grateful for my family whose constant love and support keep me motivated and confident. My accomplishments and success are because they believed in me. Finally, I am forever thankful for the unconditional love and support throughout the entire thesis process and every day.

Chetsada Chongthanachote

# TABLE OF CONTENTS

Chapter 1

# INTRODUCTION

## 1.1 Background and Significance of Research

The registration procedure has a problem that takes time in order to collect people's information. There are many factors that cause the delay problem such as manual registration, inefficient equipment or technology, number of people registering. Today's artificial intelligence (AI) technology is able to work with the system as well. In literature, there are many researchers have used artificial intelligence technology to solve problems such as face detection system, suspect identification system, face registration system, etc. The Thai government has set guidelines for the use of face recognition for financial services, which complies with international law and standards Fig 1.1 (https://www.bot.or.th/Thai/PressandSpeeches/Press/2020/Pages/n4463.aspx)


**Figure 1.1** Guidelines for using Face recognition for financial services by Bank of Thailand

In addition, the banking sector such as Kasikorn Bank Co., Ltd. has introduced artificial intelligence (AI) technology applied to e-KYC services that allows KBank's retail customers to access financial services across 1,300 Boonterm kiosks, based on the highest standards of security and accuracy. It can detect 512 points on the human face. As each point on the human face represents one password, it would be extremely difficult to forge all 512 points (asianbankingandfinance, 2021). Fig1.2 (https://thesmartlocal.com/thailand/contactless-banking)

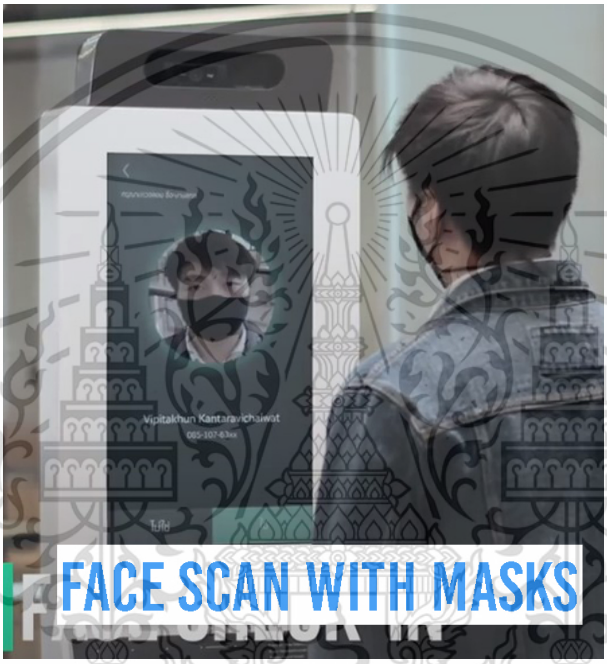



**Figure 1.2** Face identity application in Kasikorn Bank Co., Ltd.(Kasikornbank)

With the help of these methodologies, several deep learning algorithms for diverse tasks, including face recognition in particular, made significant advancements. The hierarchical architecture of deep learning techniques helps face recognition systems learn discriminative face representation. As a result, deep learning approaches greatly enhance current performance on face recognition and promote a variety of effective real-world applications. (Tahmid Hasan Fuad and Awal Ahmed Fime 2021). In terms of algorithms, we use the convolution neural network to predict the Labeled Faces in the Wild Home dataset.

Therefore, this work focuses on the implementation of convolution neural network techniques as part of deep learning. Based on multi function which can specify individual characteristics, we aim to develop methods that can analyze face and wearing a mask from pictures. Moreover, VGG16 and ResNet techniques are going to be incorporated with CNN in this study. For performance evaluation, precision and recall are applied with real dataset.

## 1.2 Objectives

1) To study and operate a facial recognition system to analyze face and mask wearing by VGG16 and ResNet50 methods.

## 1.3 Research Scope

1) Convolution neural network technique is going to be discussed.
2) We use secondary data, named Labeled Faces in the Wild Home (LFW) dataset (http://vis-www.cs.umass.edu/lfw/).
3) The tools called the Python programming language based on Opencv library are applied.

## 1.4 Expect Outcome

1) Face recognition based on a multi-function convolution neural network is obtained.
2) The effectiveness of the proposed methods is derived by considering precision and recall values.

Chapter 2

# LITERATURE REVIEW

## 2.1 Face detection technology

Commonly used biometric methods have many disadvantages. Iris recognition is extremely accurate, but expensive in breadth and not widely accepted. Fingerprints are reliable and non-invasive biometrics, but not for the uncooperative. Currently, facial recognition appears to be a good compromise between reliability and social acceptance, a good balance of security and privacy. Facial recognition technologies can work in unrestricted recording conditions and have the great advantage that they can work in places with large numbers of unintentional visitors. Due to these advantages, facial recognition has become one of the most popular biometric technologies. Faces are complex objects; therefore, recognizing and recognizing them is a challenging task, although humans are able to do it with relative ease. Given any image, the goal of face detection is to determine whether there are faces in the image, and if so, the location and extent of each face. Then the face must be recognized. Although this may seem like a trivial task to humans, it is a very challenging task for any hardware system and thus has been the most important research topic in image processing technology for the past few decades one. The efforts of machine detection and recognition of faces can be now combined into a general terminology referred to as automated face recognition (AFR). however, is to mimic the activities of the human brain performing the tasks of face detection and recognition. The main problem is understanding how to create facial representations that enable the kind of powerful facial recognition that people exhibit in their everyday interactions. The techniques of AFR can be broadly divided into two interlinked operations: (a) face detection and (b) face recognition as shown in Figure 2.1 Face detection is a necessary first step in locating and extracting face regions from the background. Problem-solving involves segmenting and extracting faces and possibly facial features from

uncontrolled backgrounds. Face recognition operation involves performing verification and identification [6]. This step takes the probe images extracted from the scene in the face detection stage and compares them with a previously registered database of known faces. A search of the best matching image is then performed to identify the most likely matching face. The final stage of face recognition is identification and verification. Identification is the process of comparing a face with a set of two or more faces in order to determine the most likely match. Face evaluation is the most common way of contrasting the test face and one more known face in the data set, coming about in one or the other acknowledgment as client or dismissal of the face as imposture. The picture which is to be either confirmed or perceived is supposed to be a question picture. Exhibition pictures are those pictures with which the inquiry picture is analyzed. [Asit Kumar Datta, Madhura Datta, Prdipta Kumar Banerjee, Face detection and recognition]

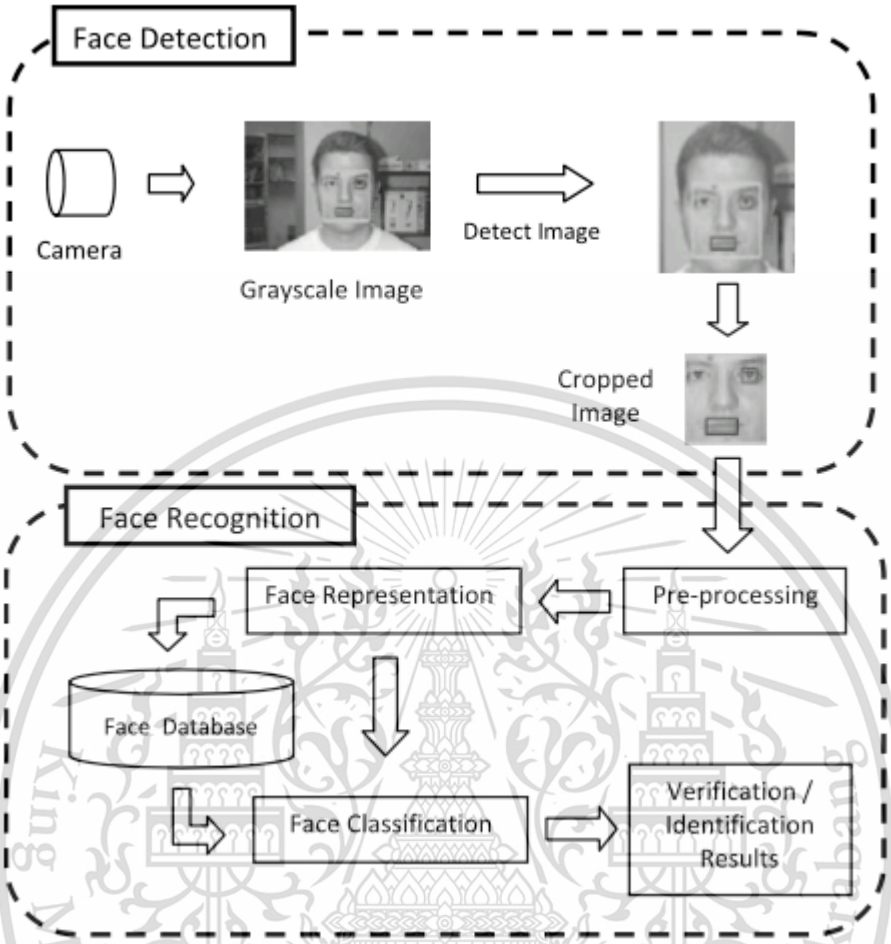



**Figure. 2.1** System diagram of a typical face detection/face recognition system

## 2.2 Neural network-based face recognition

Neural networks provide another nonlinear solution to the face recognition problem. The advantage of neural networks over linear networks for classification is that they can reduce misclassification between neighborhood classes. The basic idea is to consider a grid with one neuron for each pixel in the image. However, because of the pattern dimension, the neural network is not trained directly on the input images, but before applying this dimensionality reduction technique. One solution to this problem is to use a second neural network running in auto-association mode. First, the face image is approximated by a first grid (auto-association) as a vector of a smaller dimension, then this vector is finally used as input to the classification.

In general, however, neural network-based methods encounter problems as the number of classes increases. Furthermore, they are not suitable for single model image recognition tasks as each individual requires multiple model images to train the system. [Asit Kumar Datta, Madhura Datta, Prdipta Kumar Banerjee, Face detection and recognition]

## 2.3 Convolutional neural network

A convolutional neural network, or CNN, is a deep learning neural network designed for processing structured arrays of data such as images. The state-of-the-art for many visual applications, such as image classification, convolutional neural networks are widely employed in computer vision. They have also found success in NLP (natural language processing) for text classification. A feed-forward neural network with up to 20 or 30 layers is known as a convolutional neural network. The convolutional layer is a unique kind of layer that gives convolutional neural networks its power [Thomas Wood, DeepAI]. A convolutional neural network's architecture is a multi-layered feed-forward neural network created by sequentially stacking numerous hidden layers on top of one another. Convolutional neural networks can learn hierarchical features because of their sequential construction.

A simple convolutional neural network that aids understanding of the core design principles is the early convolutional neural network LeNet-5 in Figure 2.2

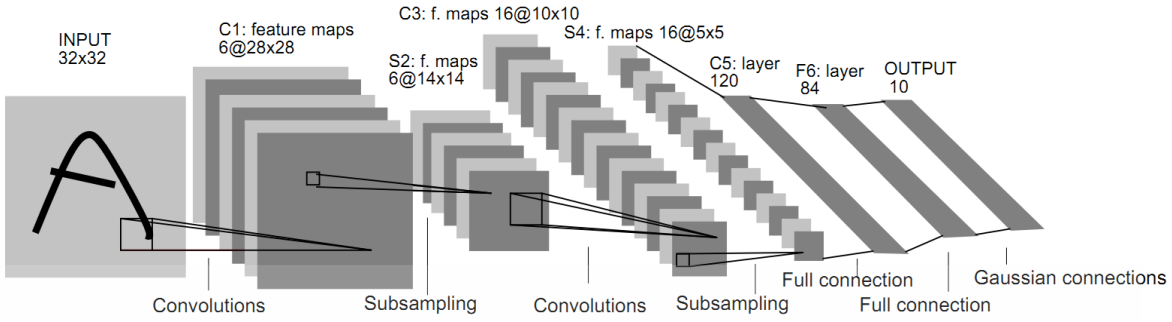



Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

**Figure 2.2** Convolution neural network architecture

**VGG16** is regarded as one of the best vision model architectures available today. The most distinctive feature of VGG16 is that it focuses on having convolution layers of 3x3 filter with stride 1 rather than having a lot of hyper-parameters and always uses the same padding and maxpool layer of 2x2 filter with stride 2. Throughout the entire architecture, convolution and max pool layers are arranged in the same way. It concludes with two fully connected layers (FC) and a softmax for output. The 16 in VGG16 represents the fact there are 16 layers with weights. This network is quite huge, with over 138 million parameters.. In Figure 2.3

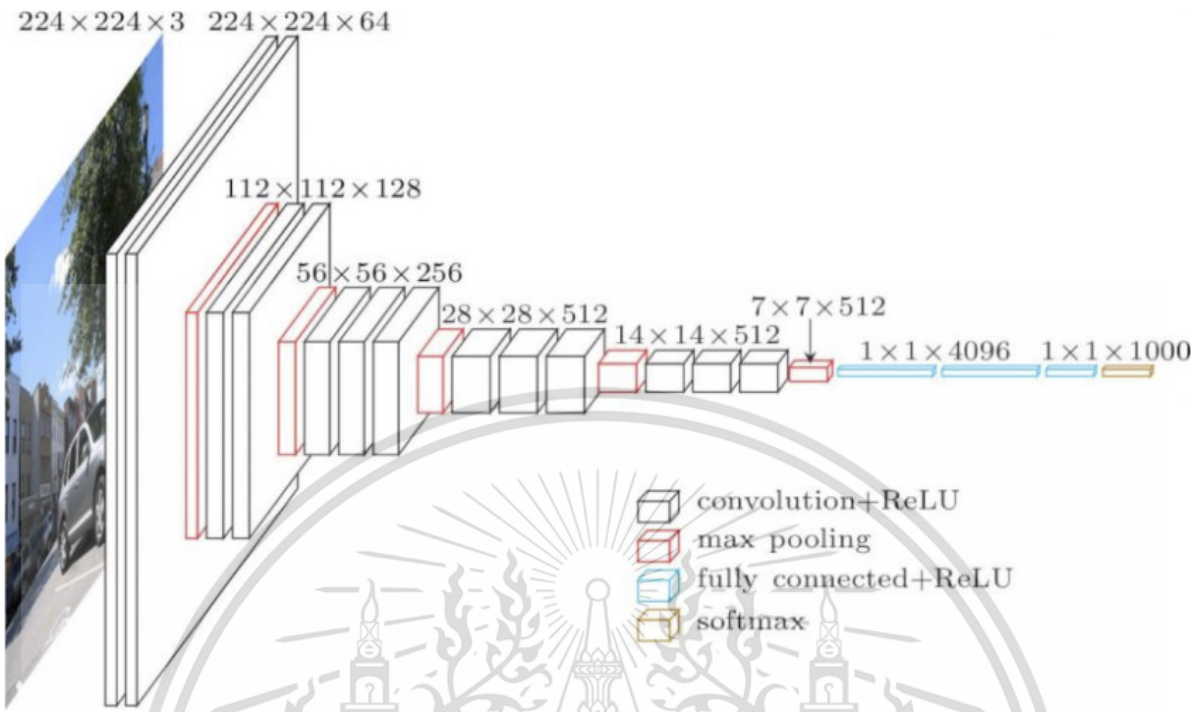



Figure 1. VGG16 Architecture

**Figure 2.3** VGG-16 architecture

**ResNet** stands for Residual Network. It is an innovative neural network for computer vision. This model was immensely successful, as can be ascertained from the fact that its ensemble won the top position at the ILSVRC 2015 classification competition with an error of only 3.57%. There are many variants of ResNet architecture i.e. same concept but with a different number of layers. We have ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-110, ResNet-152, ResNet-164, ResNet-1202 etc. ResNet models are almost similar to the networks which have convolution, pooling, activation and fully-connected layers stacked one over the other. The only construction of the simple network to make it a residual network is the identity connection between the layers. The screenshot below shows the residual block used in the network. You can see the identity connection as the curved arrow originating from the input and sinking to the end of the residual block. In Figure 2.4

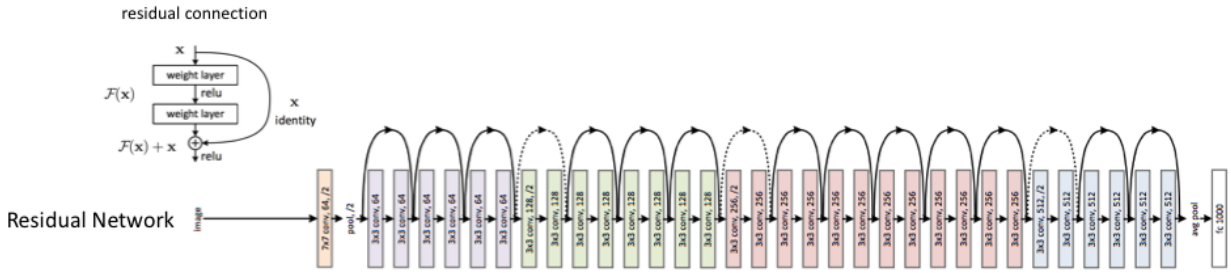



**Figure 2.4** ResNet architecture

## 2.4 Related work

Huang et al. (2019) Face recognition is the problem of identifying a specific individual, rather than merely detecting the presence of a human face. The database is to provide a large set of relatively unconstrained face images. By unconstrained, we mean faces that show a large range of the variation seen in everyday life. This includes variation in pose, lighting, expression, background, race, ethnicity, age, gender, clothing, hairstyles, camera quality, color saturation, focus, and other parameters.

Savchenko (2021) Study multi-task learning of lightweight convolutional neural networks for face recognition and classification of face attributes (age, gender, ethnicity) on borderless cropped faces to train. The need to fine-tune these networks to predict facial expressions is highlighted. Several models are proposed based on MobileNet, EfficientNet and ResNet architectures. Experiments show that they produce results close to the state-of-the-art on the UTKFace dataset for age, gender, and ethnicity detection, and the AffectNet dataset for sentiment classification. Furthermore, the results show that using the trained model as a feature extractor for facial regions in video images achieves 4.5% higher accuracy than previously known state-of-the-art single models for the AFEW and VGAF datasets EmotiW Challenge.

Manikandan et al. (2020) Facial recognition represents a system event that may determine a person whose face is backed up using Computer Vision (Open CV). Facial recognition is used in the field of identity recognition. police investigation and enforcement It is a feature system that supports facial expressions. This system operates in 2 steps, they are the training phase and the test phase. This study consists of three fundamentals. Face the discovery from the picture, The origin point, and storage of many commemorative photographs and recognition. Face Changing Rules are used to describe faces from a given image. The most useful and distinctive option of the face image is withdrawn from the origin. Finding faces can be grueling as the filming area and video frame have advanced backgrounds. Completely different head performances and maskings, such as holding glasses or scarves.

Gwyn et al. (2021) In the realm of computer security The username/password standard is becoming increasingly obsolete. Using the same username and password across different accounts May allow users to open potential vulnerabilities. Future authentication methods need to maintain the ability to provide secure access without sacrificing speed. Facial recognition technology is fast becoming an important part of user safety. This makes it possible to authenticate users at a secondary level.

Increasing traditional username and password security with facial biometrics has already yielded impressive results. However, studies of these techniques are necessary to determine how effective these methods are within various parameters. A Convolutional Neural Network (CNN) is a powerful classification approach that is often used for image identification and verification. Quite recently, CNN has shown great promise in the field of facial recognition. The comparative study presented in this paper offers an in-depth analysis of several cutting-edge deep learning facial recognition technologies. to consider through accuracy and other indicators which is the most efficient. In studies, VGG-16 and VGG-19 demonstrated the highest level of image recognition accuracy, including the F1-Score. CNN optimization should be documented as an effective way

to increase username/code standards. pass current By increasing the security of current methods with additional facial biometrics.

Dara and Palanivel (2021) Being able to automatically identify and certify human faces using real-time images is essential in surveillance. security and other domains related There is a separate application that helps to identify a person in a specific location which helps in the detection of intruders. Real-time recognition is essential for surveillance. A number of machine learning methods, along with classifiers, are used for facial recognition. This work His work introduces a new real-time facial recognition system. Then use a powerful real-time facial recognition algorithm to recognize faces with a known database. For real-time facial recognition, VGG-16 with Transfer Learning and Convolutional Neural Network (CNN) is used.

Tammina (2021) Transfer learning is a method of reusing a pre-trained model of knowledge for another task. Transfer learning can be used for classification, regression, and clustering problems. The pre-trained models with VGG - 16 with Deep Convolutional Neural Network to classify images.

Microsoft Research (2015) Deep convolutional neural network has led to a major breakthrough in image classification. Deep Networks naturally integrate low/medium/high-level properties and classifiers into end-to-end multi-layers, and the "level" of properties can be supplemented by the number of nested layers (depth). The latest shows that network depth is of paramount importance, And the leading results of all challenging ImageNet datasets take advantage of "very deep" models. Many other simple image recognition tasks benefit greatly from very deep models as well.

Coe and Atay (2021) The objective of this research was to assess the impact of facial recognition competitions on two types of algorithms for facial recognition. An in-depth evaluation

plan for each algorithm with results and experimental results. and provides analysis of machine learning algorithms and deep learning algorithms. Summary of the investigation Compare the results of two types of algorithms and compare their accuracy, metrics, miss rates, and performances to observe which algorithms mitigate racial bias the most. Racial bias was assessed in five different machine learning algorithms and three deep learning algorithms using racially unbalanced datasets and compared the accuracy and error rates between the two. All tested algorithms and reported that SVC is the superior machine learning algorithm and VGG16 is the best deep learning algorithm. experimental study.

Chapter 3

# RESEARCH METHODOLOGY

The objective of this study was 1) to investigate the use of deep learning to detect masked persons by bringing knowledge of Face detection, Image Classification to apply in interesting applications. The problem that arises is providing information to learn and sorting out masks from masks with a variety of colors. This will cause a detection error to occur.

## 3.1 Datasource

In the study, the study of the color diversity of the masks was selected. The researcher has downloaded the Labeled Faces in the Wild (LFW) dataset which is an image dataset containing face photographs, collected especially for studying the problem of unconstrained face recognition. It includes over 13,000 images of faces collected from across the web. For this reason, we used a picture of a mask from the Internet, black and white.

## 3.2 Research tools

This research was done using image data for face detection and face mask classification. To sort out masked and non-masked person, The researcher has chosen to use Python programming language which is an open source development technology. For preparing mask wear data, the OpenCV library was implemented to cope with detecting landmark faces in the face. Detecting landmark faces is a subset of the shape prediction problem. Given an input image, a shape predictor makes an attempt to localize key points of the shape. The pre-trained facial landmark detector inside the dlib library is used to estimate the location of 68 (x, y)-coordinates that map to facial structures on the face in Figure 3.1

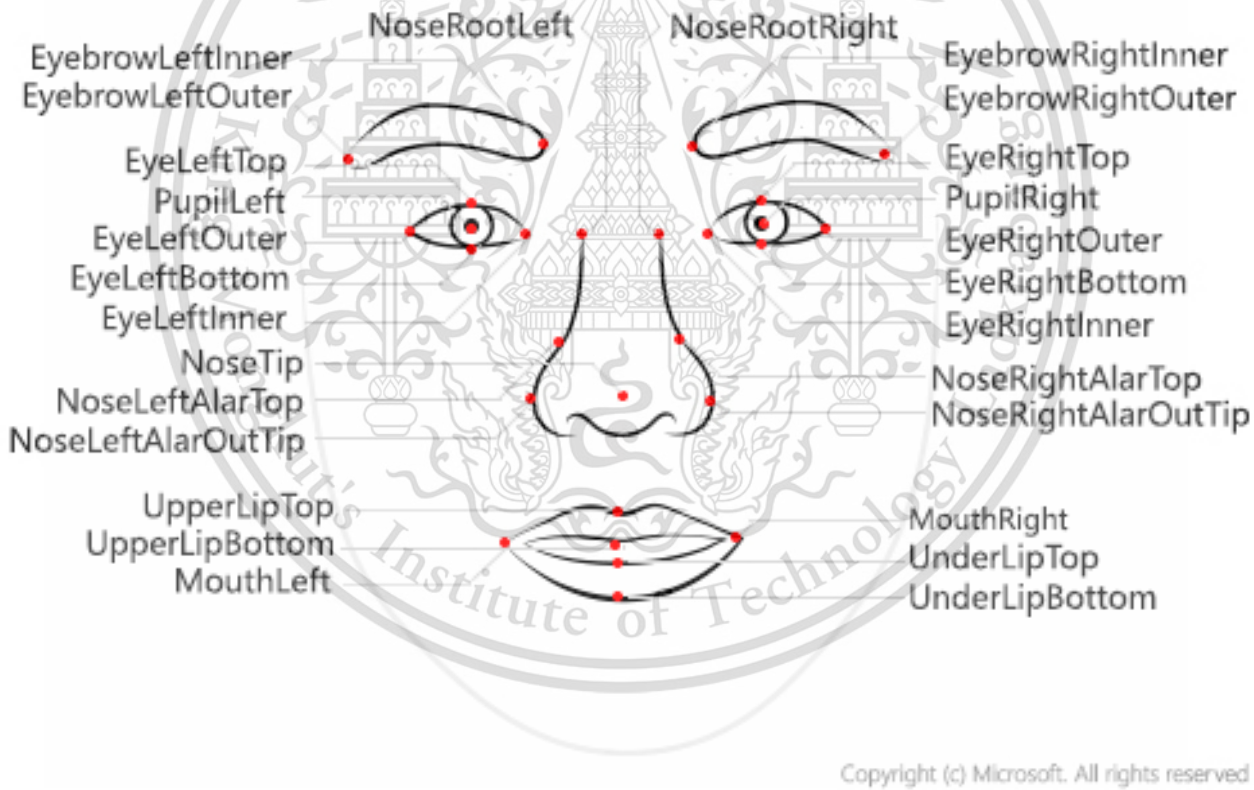



**Figure 3.1** Face landmark points

For making face mask classification to sort out masked and non-masked persons, we chose to utilize TensorFlow which is Google's technology for deep learning. We employ Vgg-16 and Resnet50 for screening masked and non-masked individuals.

## 3.3 Research methodology

### 3.3.1 Data preparation

We prepared Wild Home (LFW) data for generating masked person data. Usually, the information to wear a mask must be worn to match the shape of the face above the mouth area. Therefore, we has taken Face detection and used the data to detect the location of highlights (eyes, mouth, nose) inside a face on a landmark face using shape_predictor_68_face_landmarks to be able to find faces on the dataset as shown in following Figure 3.2


**Figure 3.2** Facial landmarks recognition with shape_predictor_68_face_landmarks

Then, we bring the image that has been detected, face detection, to crop the mouth area of the face of the data. When the mask's image is obtained, its size is later adjusted to match the picture. It is then combined with the cropped data. The matrix of the two data must be matched

by using the bitwise method provided in OpenCV package to make the use of Bitwise operation. This technique includes the bitwise AND, OR, NOT, and XOR operations. They will be highly useful while extracting any part of the image, defining and working with non-rectangular ROI's, etc. Figure 3.3 shown an example of how to change a particular region of an image.

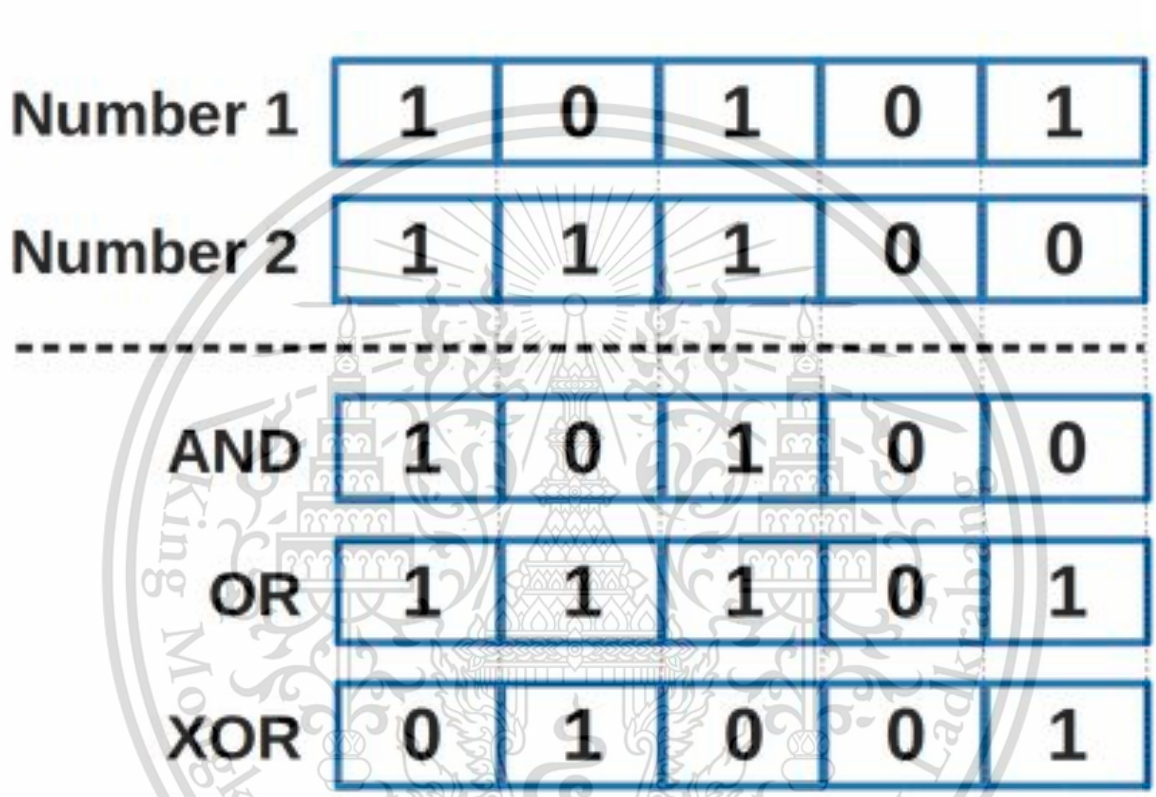



**Figure 3.3** Explain Bitwise Operations

Bitwise joins for each element of two arrays are computed using the formula dst = src1 & src2.. The function bitwise_and calculates the per-element bitwise logical conjunction for two arrays when src1 and src2 have the same size

$$dst(I)=src1(I) \wedge src2(I) \text{ if } mask(I)\neq 0 \quad (3.1)$$

src1(I) = input array1

scr2(I) = input array2

According to Bitwise, the researchers were able to crop an image by specifying x, y according to the size of each image in order to obtain the desired position as shown in Figure 3.4

```
roi = img1[y+50:y+h+350, x-250:x+w+250]
```

**Figure 3.4** Code and Crop face detection image

After the cropping method is finished, we obtained the mask image using a png file and resized the image based on the size of the face cropped above to fit the individual's face shape. Consequently, we cut the edges of the mask using black-and-white images, capturing only the black ones, as illustrated in Figure 3.5

```
img2 = cv2.imread('{path}/dataset/black_mask.png')
small_img = cv2.resize(img2,(roi.shape[1],roi.shape[0]))
cv2_imshow(small_img)
rows,columns,chanels = small_img.shape
small_img.shape
small_img_gray = cv2.cvtColor(small_img, cv2.COLOR_RGB2GRAY)
ret, mask = cv2.threshold(small_img_gray, 120, 255,
cv2.THRESH_BINARY)
```

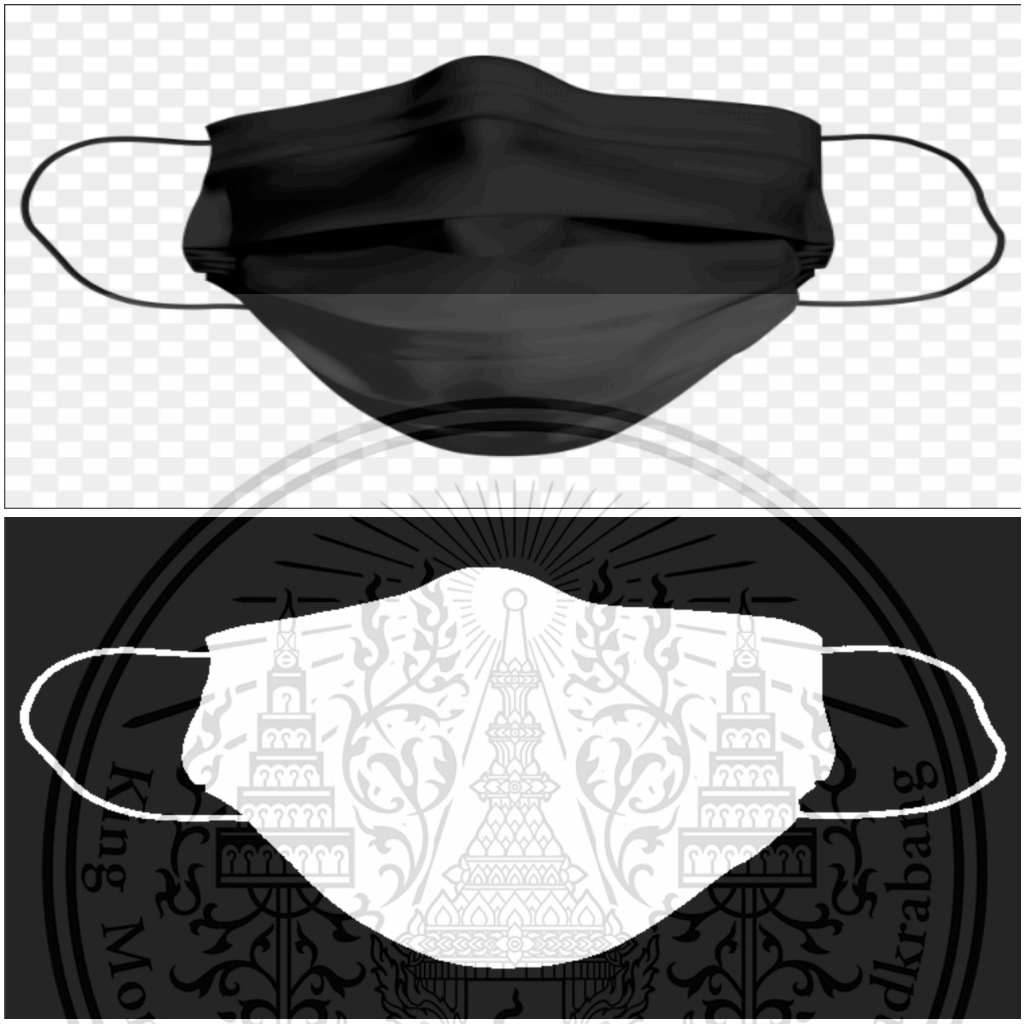


**Figure 3.5** Code and Mask adjustment image

Then the desired image information is obtained from the aforementioned process. Figure 3.6 is the result of combining the mask and cropped image using the bitwise_and method.

```
# Now create a mask of logo and create its inverse mask also
img2gray = cv2.cvtColor(small_img,cv2.COLOR_BGR2GRAY)
ret, mask = cv2.threshold(img2gray, 200, 255,
cv2.THRESH_BINARY)
mask_inv = cv2.bitwise_not(mask)
# Now black-out the area of logo in ROI
img1_bg = cv2.bitwise_and(roi,roi,mask = mask)
img2_fg = cv2.bitwise_and(small_img,small_img,mask =
mask_inv)
cv2_imshow(img1_bg)
cv2_imshow(img1_bg)
final_roi = cv2.add(img1_bg,img2_fg)
cv2_imshow(final_roi)
cv2_imshow(mask_inv)
img1[y+50:y+h+350, x-250:x+w+250] = final_roi
cv2_imshow(img1)
```

**Figure 3.6** Code and Mask adjustment image

All pictures integrated between the face and the mask have been created. In Figure 3.7, a png image of has been placed over a black-and-white image of a mask. Figure 3.8 depicts a complete image that was created by merging the obtained image with the original.

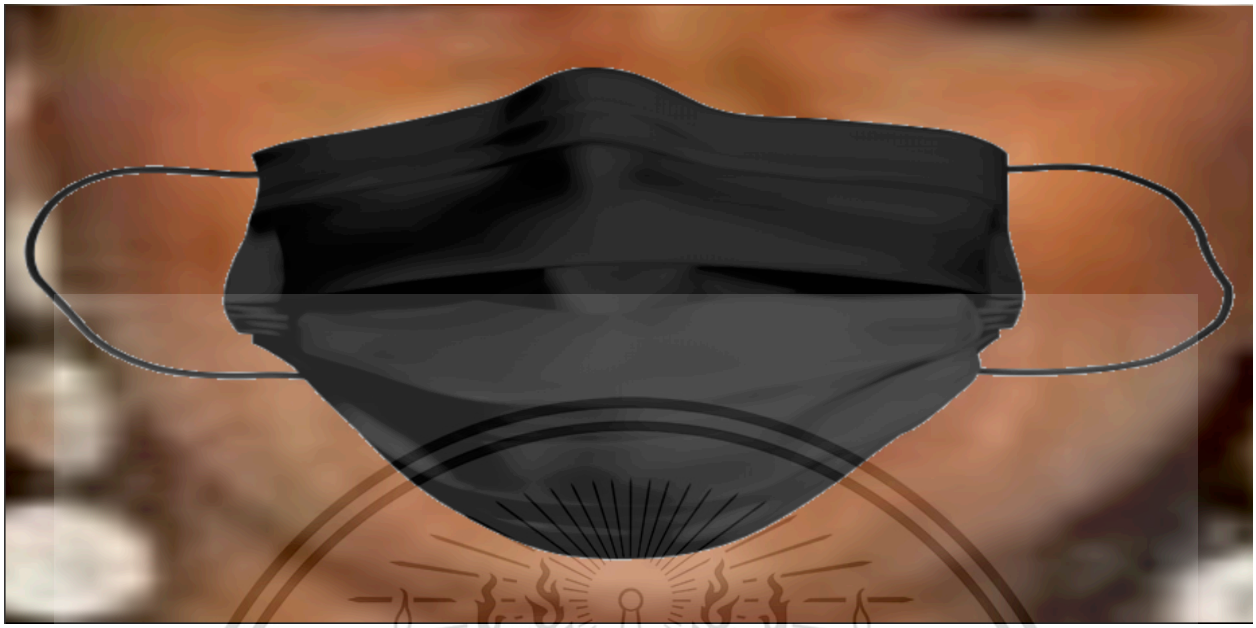


**Figure 3.7** Merge mask and face cropped

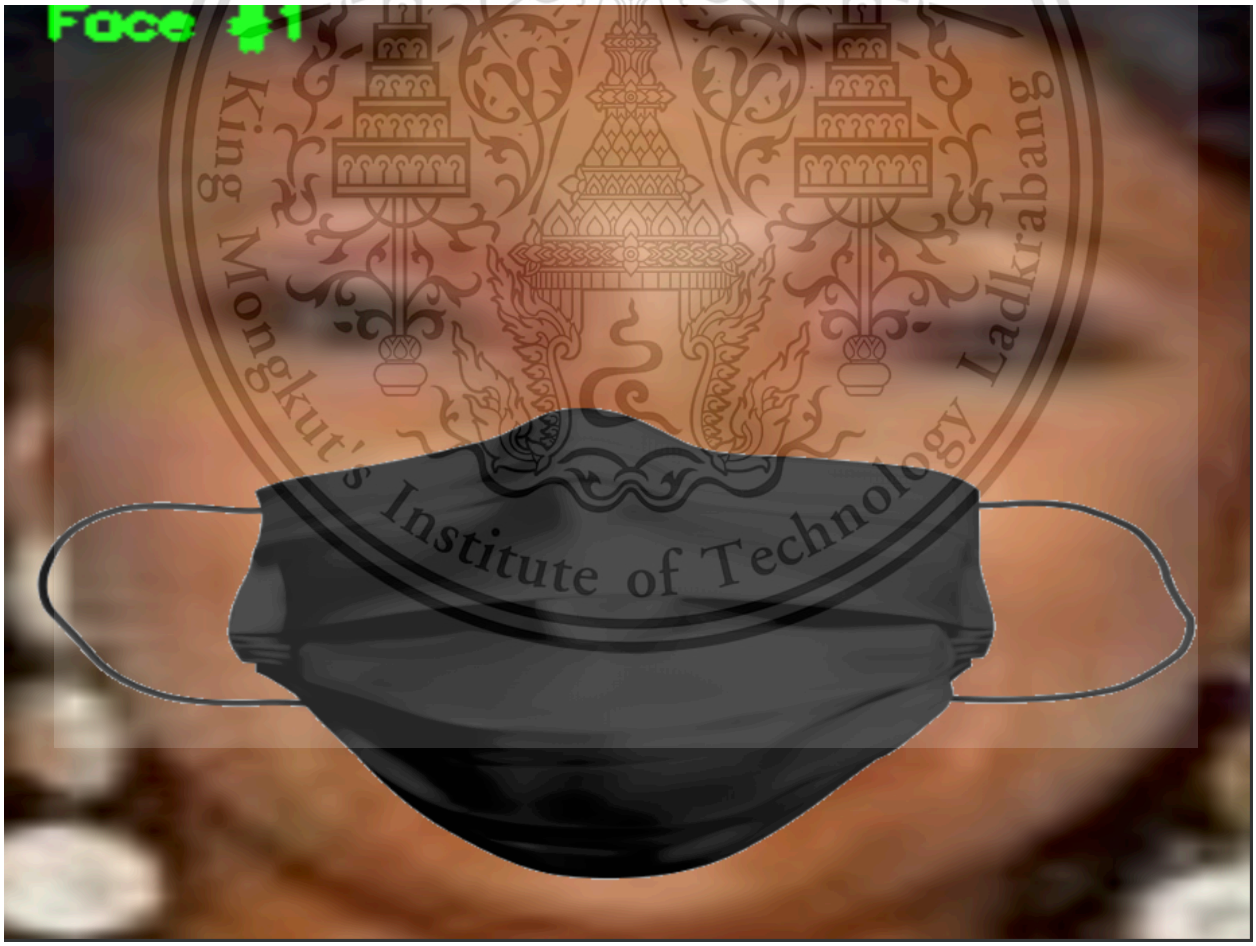


**Figure 3.8** Result of face and mask integration

3.3.2 Implementation

All required information was prepared as mentioned above. Data for prediction was labeled as 1 = Mask, 0 = NonMask. Consequently, a total of 1226 images were prepared by the researchers, which can be divided into 627 images of people wearing masks and 599 photos of people without masks as shown in Figure 3.9 and 3.10.

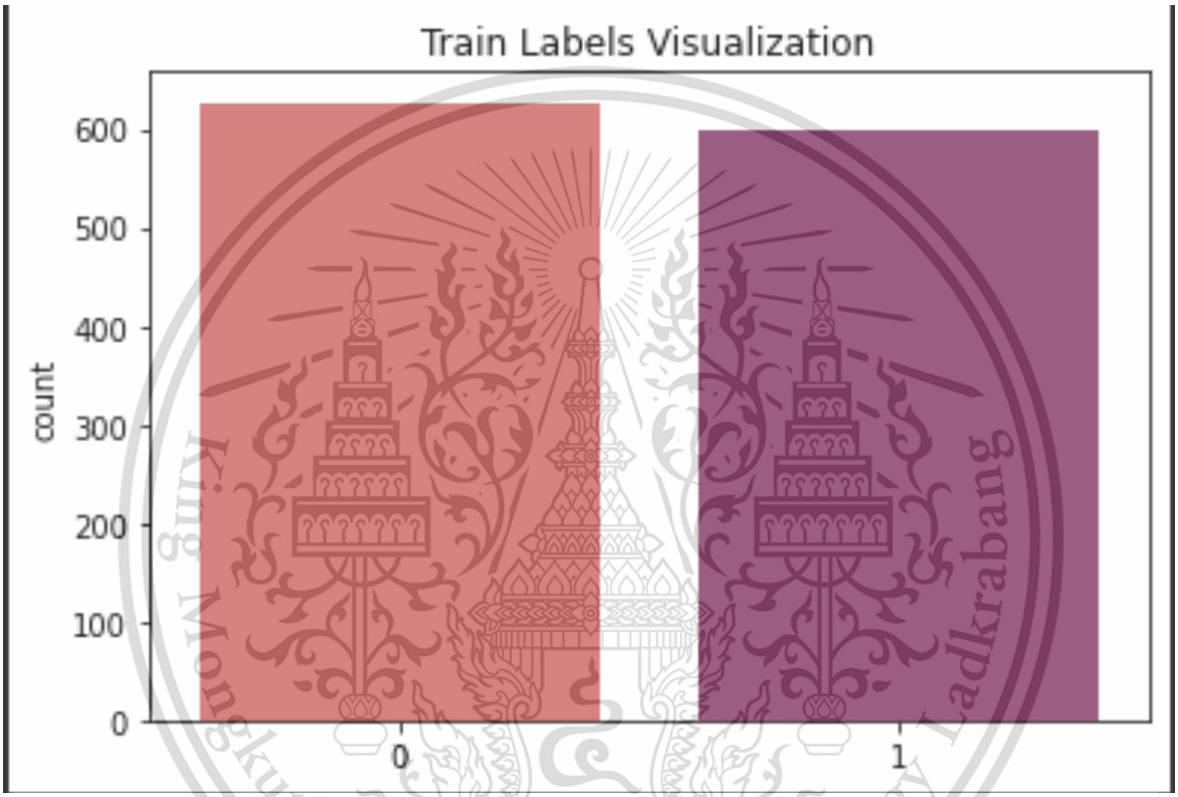
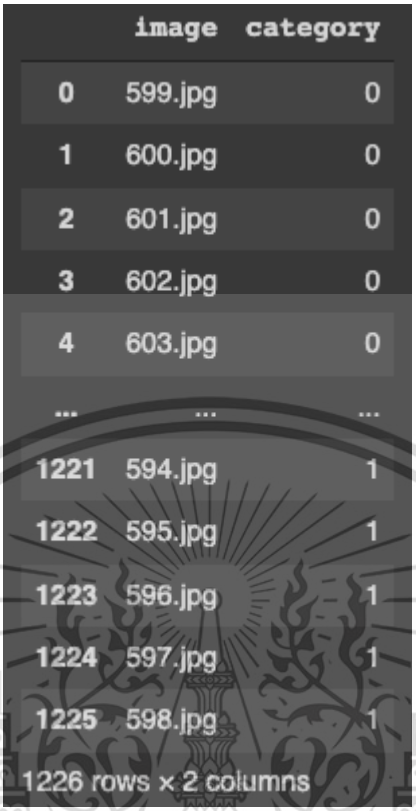



**Figure 3.9** Train dataset ratio



| | image | category |
|---|---|---|
| 0 | 599.jpg | 0 |
| 1 | 600.jpg | 0 |
| 2 | 601.jpg | 0 |
| 3 | 602.jpg | 0 |
| 4 | 603.jpg | 0 |
| ... | ... | ... |
| 1221 | 594.jpg | 1 |
| 1222 | 595.jpg | 1 |
| 1223 | 596.jpg | 1 |
| 1224 | 597.jpg | 1 |
| 1225 | 598.jpg | 1 |

1226 rows × 2 columns

**Figure 3.10** Summary data prepared

The dataset was split randomly into 80:20 ratios. The training data contain 70 percent of a dataset, which include 980 images. Then, the validation set covers 70% of a dataset, consisting of 246 images. There are 224 × 224 thumbnails out of the large-sized images. This setting can pave the way for a well-trained model and high classification accuracy. The first step is to preprocess with ImageDataGenerator function. We set rescale = 1/255, shear_range = 0.2, zoom_range = 0.5, height_shift_range=0.2, width_shift_range=0.2 in Figure 3.11. Then, we will learn the Vgg16 and ReNet50 mods. and we will customize the RestNet50 model by adding a Dense layer. The model was developed in Python with the Tensorflow backend library. The process of training is illustrated in Figure 3.12

```python
train = ImageDataGenerator(
  rescale = 1./255,
  shear_range = 0.2,
  zoom_range = 0.5,
  height_shift_range=0.2,
  width_shift_range=0.2,
  fill_mode='nearest',
  horizontal_flip=True,
  rotation_range = 20
).flow_from_dataframe(
  x_train,desitination+'/'+'train/MaskNonMask/',
  x_col='image',y_col='category',target_size=(128,128),
  class_mode='categorical',
  shuffle=True,
  batch_size=batch_size)
```

Figure 3.11 ImageDataGenerator function

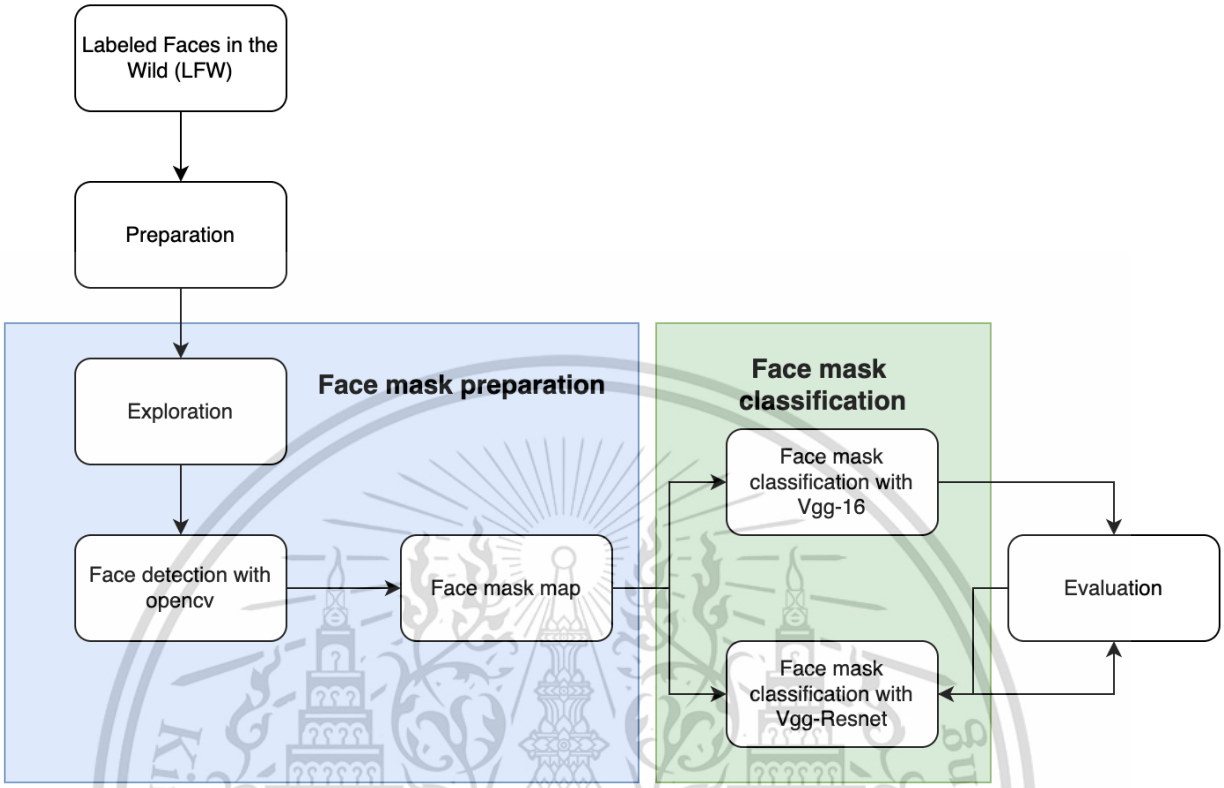



**Figure 3.12** Process model training

3.3.2.1 VGG-16

For the Vgg-16 model, we have created a layer for learning based on the standard structure of Vgg-16. We also add relu (Rectified Linear Unit) activation to each layer so that all the negative values are not passed to the next layer. After we set the default convolution architecture, we add the dense layer and flatten which comes out of the convolutions. We can see the final VGG-16 architecture in Figure 3.13

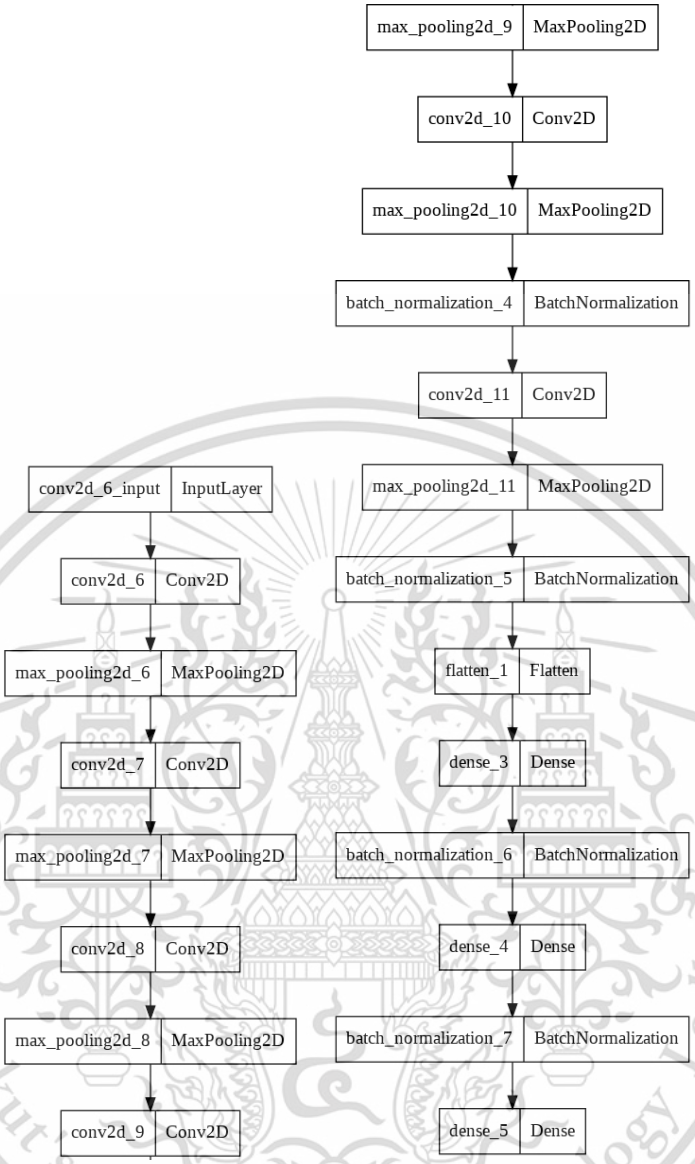



**Figure 3.13** VGG-16

We set the hyper parameters using Adam's method. Adaptive moment estimation is an algorithm for optimization technique for gradient descent. The method is really efficient when working with large problem involving a lot of data or parameters. It requires less memory and is efficient in equation 3.2

$$m_t = \beta_1 m_{t-1} + (1 + \beta_1)[\frac{\delta L}{\delta w_t}] \; v_t = \beta_2 m_{t-1} + (1 + \beta_2)[\frac{\delta L}{\delta w_t}]^2 \quad (3.2)$$

Since mt and vt have both initialized as 0 (based on the above methods), it is observed that they gain a tendency to be 'biased towards 0' as both $\beta 1$ & $\beta 2 \approx 1$. This optimizer fixes this problem by computing 'bias-corrected' mt and vt. This is also done to control the weights while reaching the global minimum to prevent high oscillations in equation 3.3.

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t} \; \hat{v}_t = \frac{m_t}{1-\beta_2^t} \tag{3.3}$$

Intuitively, we are adapting to the gradient descent after every iteration so that it remains controlled and unbiased throughout the process, corresponding to Adam. Instead of using normal weight parameters mt and vt, we substitute by the bias-corrected weight parameters $\widehat{m}_t$ and $\hat{v}_t$. As a result, we can obtained following equation 3.4

$$w_{t+} = w_t - \widehat{m}\left(\frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\right) \tag{3.4}$$

The loss function called the binary cross-entropy is applied in binary classification tasks. Loss function is applied to many independent classification problems, each problem having only two possible classes with target probabilities $y_i$ and $(1 - y_i)$. This function calculates the loss of an example by computing the average in equation 3.5

$$Loss = \frac{1}{outputsize} \sum_{i=1}^{outputsize} \; y_i * log\hat{y}_i + (1 + y_i) * log(1 + \hat{y}_i) \tag{3.5}$$

where $\hat{y}_i$ is the i-th scalar value in the model output, $y_i$ is the corresponding target value, and output size is the number of scalar values in the model output. The term "epoch" is applied to

the number of passes the machine learning algorithm has made across the full training dataset. For the epoch we use all 60 cycles to learn the model. Finally, as a result of training the model, it is noticeable that the loss value starts to decrease as the number of cycles (epoch) of learning increases, On the other hand, the value of accuracy increases.

3.3.2.2 ResNet50

For the Resnet50 model, we have created a layer for learning based on the standard structure of ResNet50. We also add relu (Rectified Linear Unit) activation in the same VGG-16. dense layer is added at Resnet50 architecture as present in Figure 3.14

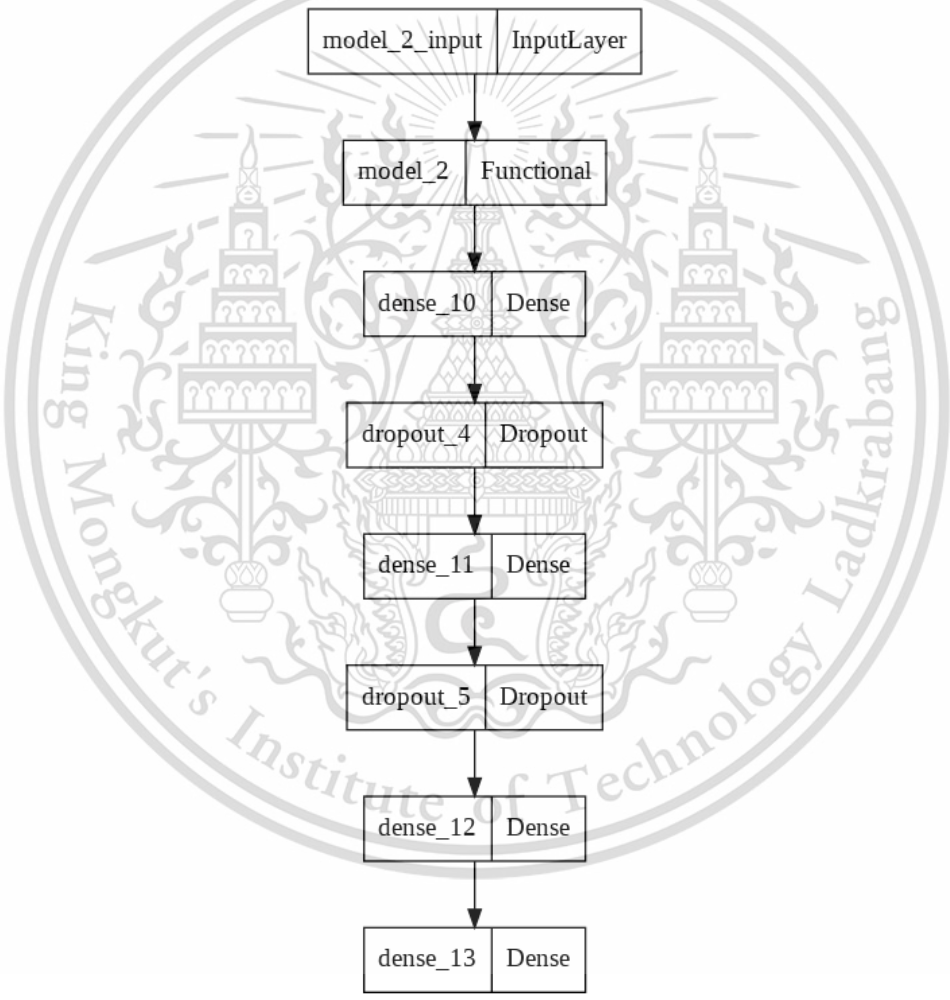



**Figure 3.14** ResNet50 with dense layer

Resnet50 model is very large compared to VGG-16. So the researcher will set callback function. Callbacks can be used to observe the training process. This is because training in a large corpus

can take a long time. This way, convergence issues or other potential problems can be identified early in the process, saving valuable time and resources.

Chapter 4

# RESULT

## 4.1 VGG-16 result

For the VGG-16 approach, the results are summarized in Figure 4.1 and Table 4.1. The output represents the performance of the model after the method has been applied. It demonstrates the accuracy improvements that contribute to 25% and 89% of model precision for non-mask detection and mask detection, respectively. The recall analysis shows similar results, with 80% for non mask detection and 40% for mask detection

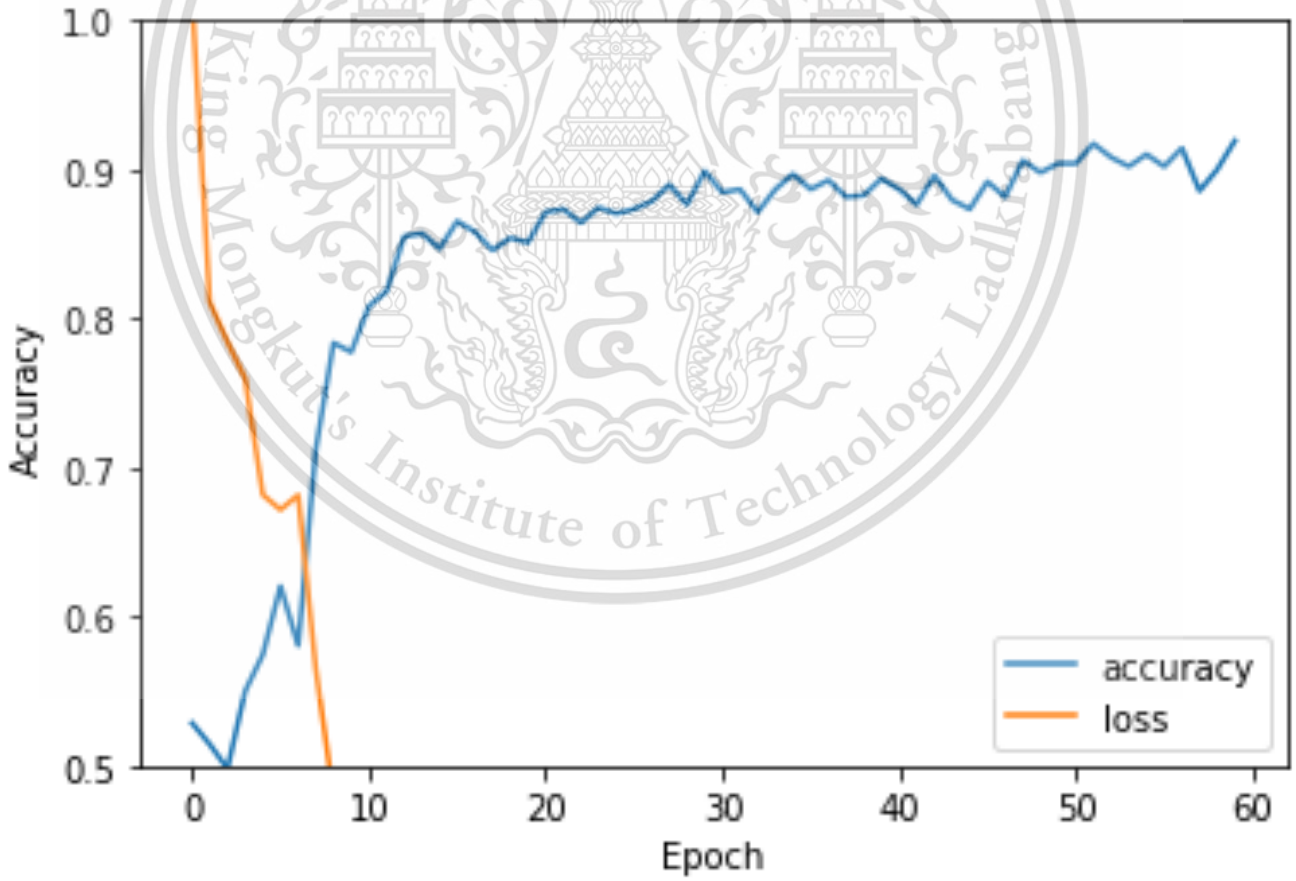



**Figure 4.1** Summary model prediction

Based on the results of the training phase, the face mask detection has a high accuracy value. However, when applying the method to the test dataset, it was found that the data had detected unsatisfactory masks in table 4.1. We have attempted to investigate classification results by conducting experiments based on different mask colors including white, pink, black, and green. There are 60% of accuracy for white mask, 99% of accuracy for pink mask, 70% of accuracy for black color, and 50% of accuracy for green color.

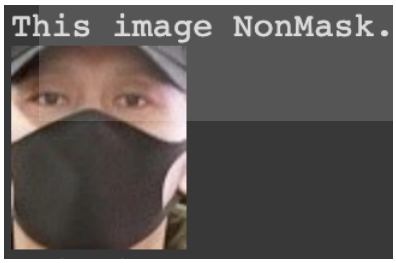
**Table 4.1** Summary model prediction

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| NonMask | 0.25 | 0.80 | 0.38 | 5 |
| Mask | 0.89 | 0.40 | 0.55 | 20 |

**Table 4.2** The VGG-16 result when we use color to evaluate performance

| Image testing | Actual | Predict |
|---|---|---|
|   | Mask | Mask |
|   | Mask | NonMask |
|   | Mask | Mask |

**Table 4.2** (Continued)

| Image testing | Actual | Predict |
|---|---|---|
|   | Mask | NonMask |
|   | NonMask | NonMask |
|   | Mask | Mask |
|   | Mask | NonMask |

## 4.2 Resnet50 result

The summary of analysis result for the Resnet50 is presented in figure 4.2 and table 4.3, which are related to the performance of the model after applying the method., The pre trained model's results in predicting the wearing of masks are not accurate enough.

Based on the results of the training phase, the face mask detection has a high accuracy value. However, when applying the method to the test dataset, it was found that the data had detected unsatisfactory non masks in table 4.3. We have attempted to investigate classification results by conducting experiments based on different mask colors including white, pink, black, and green. There are 99% of accuracy for white mask, 99% of accuracy for pink mask, 80% of accuracy for black color, and 80% of accuracy for green color.

**Table 4.3** Summary model prediction

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| NonMask | 0.00 | 0.00 | 0.00 | 5 |
| Mask | 0.80 | 1.00 | 0.89 | 20 |

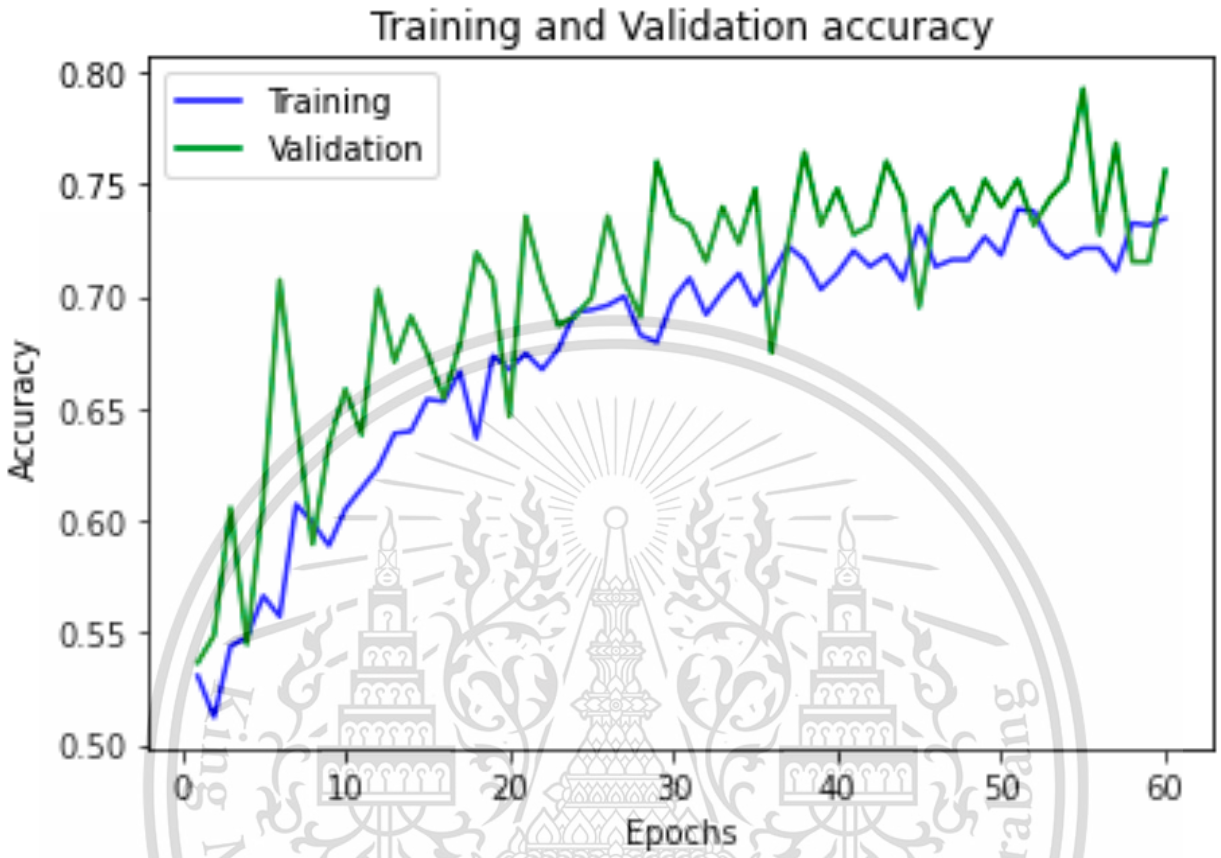



**Figure 4.2** Summary model prediction

**Table 4.4** The Resnet50 result when we use color to evaluate performance

| Image testing | Actual | Predict |
|---|---|---|
|   | Mask | Mask |

**Table 4.4** (Continued)

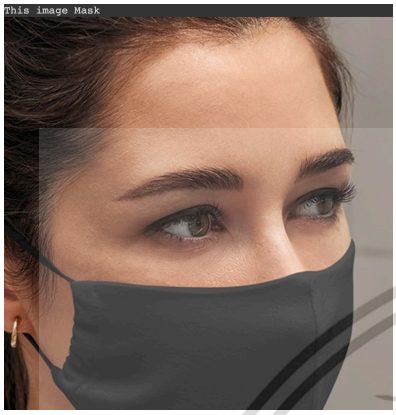
| Image testing | Actual | Predict |
|---|---|---|
|   | Mask | Mask |
|   | Mask | Mask |
|   | Mask | Mask |

**Table 4.4** (Continued)

| Image testing | Actual | Predict |
|---|---|---|
|   | NonMask | Mask |
|   | Mask | Mask |
|   | Mask | Mask |

Chapter 5

# CONCLUSION

In the study of the detection of masks on the faces of different people. The purpose of this research is to study and operate a facial recognition system to analyze face and mask wearing by VGG16 and ResNet50 methods. The findings can be summarized into the four topics as follows.

## 5.1 Face mask preparation

In terms of face detection of people in the images obtained from the LFW dataset, the face landmark model is capable of detecting face effectively. This allows us to use the data to prepare information on wearing masks. This method can also be used to guide the data preprocessing process that is required to create images of people wearing other objects such as glasses and jewelry.

## 5.2 Face mask classification

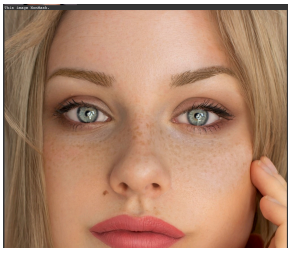
In the output of the VGG-16, the image can be detected, which can distinguish between people who wear masks and those who do not wear masks adequately. On the other hand, ResNet50 outcome exhibits extreme overfitting. In other word, this method cannot detect people who can't wear masks.

The classification result of the two models can be compared with each other as shown in Table 5.1 - 5.2

**Table 5.1** Confusion matrix to compare VGG-16 and ResNet50 methods

| | | precision | recall | f1-score | support |
|---|---|---|---|---|---|
| **VGG-16** | NonMask | 0.00 | 0.00 | 0.00 | 5 |
| | Mask | 0.80 | 1.00 | 0.89 | 20 |
| | | precision | recall | f1-score | support |
| **ResNet50** | NonMask | 0.25 | 0.80 | 0.38 | 5 |
| | Mask | 0.89 | 0.40 | 0.55 | 20 |

**Table 5.2** Some classification results of VGG-16 and ResNet50 methods

| Image testing | Actual | VGG-16 Predict | ResNet50 Predict |
|---|---|---|---|
|   | Mask | Mask | Mask |
|   | Mask | Mask | Mask |
|   | NonMask | NonMask | Mask |
|   | Mask | NonMask | Mask |
|  | NonMask | NonMask | Mask |

## 5.3 Limitation

In this study, we are limited by the computational capabilities which make it hard for us to prepare or predict a huge amount of image data. Therefore, in the data acquisition, we have to reduce the number of images from 13000 to 1226 images. For learning on the Vgg-16 and ResNet50 models, we specifically set the Epoch 60 cycles, which resulted in an adequate level of effectiveness for prediction.

## 5.4 Conclusion

In this study, the results showed that the VGG-16 model had better results than the ResNet50 model because VGG-16 was able to detect people who were unable to wear masks. We rationalized modeling with VGG-16 and ResNet50 because based on research by Gwyn and Roy (2021) which show the highest value of accuracy. VGG-16, ResNet50 for measuring the results of face. We have taken this as a reference to get the best model results. It is recommended that other researchers can make an attempt to increase the model accuracy by modifying these settings. In addition, the framework of data preprocessing can be applied to other works such as lip detection.

## Reference

Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. 2019. “Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments” [Online] Available : http://vis-www.cs.umass.edu/lfw

Andrey V. Savchenko. 2021. “Facial expression and attributes recognition based on multi-task learning of lightweight neural networks” [Online] Available : https://arxiv.org/abs/2103.17107

J. Manikandan, S. Lakshmi Prathyusha, P. Sai Kumar, Y. Jaya Chandra, M. Umaditya Hanuman. 2020. “Face Detection and Recognition using Open CV Based on Fisher Faces Algorithm” [Online] Available : https://www.ijrte.org/wp-content/uploads/papers/v8i5/E5753018520.pdf

Tony Gwyn, Kaushik Roy, Mustafa Atay. 2021. “Face Recognition Using Popular Deep Net Architectures: A Brief Comparative Study” [Online] Available : https://www.mdpi.com/1999-5903/13/7/164

Showkat A. Dara and S. Palanivel. 2021 “Neural Networks (CNNs) and Vgg on Real Time Face Recognition System” [Online] Available : https://www.researchgate.net/publication/351246098_Neural_Networks_CNNs_and_Vgg_on_Real_Time_Face_Recognition_System

Microsoft Research. 2015 “Deep Residual Learning for Image Recognition” [Online] Available : https://arxiv.org/abs/1512.03385

James Coe and Mustafa Atay. 2021. “Evaluating Impact of Race in Facial Recognition across Machine Learning and Deep Learning Algorithms” [Online] Available :https://www.researchgate.net/publication/260299859_Face_Recognition_Performance_Role_of_Demographic_Information

Himanshu Singh. 2019 “Practical Machine Learning and Image Processing” [e-Book] Available: https://link.springer.com/book/10.1007/978-1-4842-4149-3

Haowei Liu. 2015 “Face Detection and Recognition on Mobile Devices” [e-Book] Available : www.elsevier.com/permissions.

Asit Kumar Datta, Madhura Datta, Pradipta Kumar Banerjee. 2016 “Face Detection and Recognition Theory and Practice” Available : http://www.taylorandfrancis.com

# APPENDIX

In this implementation We have developed a total of 3 files:

- prepare_face_mask.ipynb
- Vgg16-Mask_classification.ipynb
- ResNet-Mask_classification.ipynb

prepare_face_mask.ipynb

```
!nvidia-smi
```

```
!pip install mtcnn
!pip install tensorflow
!pip install opencv-python
!pip install cmake
!pip install dlib
!pip install opencv-contrib-python
```

```
import mtcnn
import cv2
from imutils import face_utils
import imutils
import shutil
import dlib
import os
import numpy as np
from PIL import Image , ImageDraw, ImageFilter
```

```
from google.colab.patches import cv2_imshow
from google.colab import drive

print("mtcnn version : "+mtcnn.__version__)
print("cv2 version : "+cv2.__version__)
print("dlib version : "+dlib.__version__)

drive.mount('/content/gdrive')
desitination = "{path}/dataset/dataset"

detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('{path}/Code/Model/shape_predictor_68_face_landmarks.dat')

def shape_to_np(shape, dtype="int"):
        # initialize the list of (x, y)-coordinates
        coords = np.zeros((68, 2), dtype=dtype)
        # loop over the 68 facial landmarks and convert them
        # to a 2-tuple of (x, y)-coordinates
        for i in range(0, 68):
                coords[i] = (shape.part(i).x, shape.part(i).y)
        # return the list of (x, y)-coordinates
        return coords

testimg = cv2.imread(desitination+'/Yolanda_King_0001.jpg')
dsize = (800, 600)
# image = imutils.resize(image, width=600, height=600)
image = cv2.resize(testimg,dsize)
```

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2_imshow(testimg)

cv2.imread(desitination+'/Yolanda_King_0001.jpg')
# Convert the image color to grayscale
dsize = (800, 600)
image = cv2.resize(image,dsize)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
rects = detector(gray, 1)
# Detect the face
for (i, rect) in enumerate(rects):
  shape = predictor(gray, rect)
  shape = face_utils.shape_to_np(shape)
    # convert dlib's rectangle to a OpenCV-style bounding box
    # [i.e., (x, y, w, h)], then draw the face bounding box
  (x, y, w, h) = face_utils.rect_to_bb(rect)
  getCrop = image[y-20:y+h+20,x-20:x+w+20]
    # show the face number
  cv2.putText(image, "Face #{}".format(i + 1), (x - 10, y - 10),
  cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
    # loop over the (x, y)-coordinates for the facial landmarks
    # and draw them on the image
  print("x =",x)
  print("w =",w)
  print("y =",y)
  print("h =",h)
# show the output image with the face detections + facial landmarks
```

```
cv2_imshow(getCrop)


# cv2.circle(image, (234, 325), 2, (0, 0, 255), -1)
image = cv2.imread(desitination+'/Yolanda_King_0001.jpg')
# image = cv2.imread(os.path.join(desitination,des_list[1]),cv2.IMREAD_COLOR)
img1 = cv2.resize(getCrop,dsize)
cv2_imshow(img1)

print(x,y)

# roi = img1[y+100:y+h, x:x+w]
roi = img1[y+50:y+h+350, x-250:x+w+250]
cv2_imshow(roi)
roi.shape

img2 = cv2.imread('{path}/dataset/black_mask.png')
small_img = cv2.resize(img2,(roi.shape[1],roi.shape[0]))
cv2_imshow(small_img)

rows,columns,chanels = small_img.shape
small_img.shape

small_img_gray = cv2.cvtColor(small_img, cv2.COLOR_RGB2GRAY)
# cv2_imshow(small_img_gray)
ret, mask = cv2.threshold(small_img_gray, 120, 255, cv2.THRESH_BINARY)
print(mask.shape)
```

```
# Now create a mask of logo and create its inverse mask also
img2gray = cv2.cvtColor(small_img,cv2.COLOR_BGR2GRAY)
ret, mask = cv2.threshold(img2gray, 200, 255, cv2.THRESH_BINARY)
mask_inv = cv2.bitwise_not(mask)

# Now black-out the area of logo in ROI
img1_bg = cv2.bitwise_and(roi,roi,mask = mask)

# Take only region of logo from logo image.
img2_fg = cv2.bitwise_and(small_img,small_img,mask = mask_inv)
cv2_imshow(img1_bg)
cv2_imshow(img1_bg)
final_roi = cv2.add(img1_bg,img2_fg)
cv2_imshow(final_roi)
cv2_imshow(mask_inv)


img1[y+50:y+h+350, x-250:x+w+250] = final_roi
cv2_imshow(img1)
img1.shape
```

Vgg16-Mask_classification.ipynb

```
!pip install keras
```

```
import cv2
from imutils import face_utils
import imutils
import shutil
import dlib
import os
import seaborn as sns
import pandas as pd
import numpy as np
import tensorflow as tf
from google.colab.patches import cv2_imshow
from google.colab import drive
from sklearn.model_selection import train_test_split
import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPool2D , Flatten
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam, SGD, RMSprop
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import datasets, layers, models
from tensorflow.keras.layers import  Dense,BatchNormalization,Dropout,Input
import matplotlib.pyplot as plt
```

```
drive.mount('/content/gdrive')
desitination = "{path}/dataset/"

train_path = desitination+'train'
val_path = desitination+'val'
all_categories = []
all_filename = []

images = os.listdir(train_path)
for index,r in enumerate(images):
  if r != 'MaskNonMask':
    read = images[index]
    read = train_path+'/'+read
    print(read)
    print(r)
    for i,file in enumerate(os.listdir(read)):
      if r == 'Mask':
        all_filename.append(file)
        all_categories.append(1)
      else:
        all_filename.append(file)
        all_categories.append(0)
  else :
    None
  # print(all_filename,all_categories)
# #creat data fram to save each image with its label
df =pd.DataFrame({
```

```
    'image': all_filename,
    'category':all_categories
})

df['category'].value_counts()
plt.title('Train Labels Visualization')
sns.countplot(x=all_categories,palette='flare')
plt.show()

df['category'] = df['category'].replace({1:'Mask',0:'NonMask'})

x_train,x_val  = train_test_split(df ,random_state=42,shuffle=True,test_size=0.2)

x_train = x_train.reset_index(drop=True)
x_val = x_val.reset_index(drop=True)
print('Train Images shape is       : ',x_train.shape)
print('Validation  Images  shape is : ',x_val.shape)

batch_size = 8
#create image generator for images
train = ImageDataGenerator(
                           rescale = 1./255,
                            shear_range = 0.2,
                            zoom_range = 0.5,
                            height_shift_range=0.2,
                            width_shift_range=0.2,
                            fill_mode='nearest',
```

```
                                horizontal_flip=True,
                                rotation_range = 20,

                            ).flow_from_dataframe(
    x_train,
    desitination+'/'+'train/MaskNonMask/',
    x_col='image',
    y_col='category',
    target_size=(128,128),
    class_mode='categorical',
    shuffle=True,
    batch_size=batch_size
    )

# Create Image Data Generator for Test/Validation Set
validate = ImageDataGenerator(
                                rescale = 1./255,
                                shear_range = 0.2,
                                zoom_range = 0.5,
                                height_shift_range=0.2,
                                width_shift_range=0.2,
                                fill_mode='nearest',
                                horizontal_flip=True,
                                rotation_range = 20,).flow_from_dataframe(
    x_val,
    desitination+'/'+'train/MaskNonMask/',
    x_col='image',
```

```
    y_col='category',
    target_size=(128,128),
    class_mode='categorical',
    shuffle=True,
    batch_size=batch_size
    )

for train_img , train_label in train :
    print('image shape ',train_img.shape)
    print('label  shape ',train_label.shape)
    break

for v_img , v_label in validate :
    print('image shape ',v_img.shape)
    print('label  shape ',v_label.shape)
    break

# from tensorflow.keras.applications import ResNet50V2
from tensorflow.keras.utils import plot_model
#start bulding CNN Model
cnn_model = Sequential()
cnn_model = models.Sequential()
cnn_model.add(layers.Conv2D(64,(3,3),padding ='Same',activation =
'relu',input_shape=(128,128,3)))
cnn_model.add(layers.MaxPooling2D(2,2))
cnn_model.add(layers.Conv2D(64,(3,3) ,padding ='same',activation='relu'))
cnn_model.add(layers.MaxPooling2D(2,2))
```

```
cnn_model.add(layers.Conv2D(128,(3,3),padding ='same',activation='relu'))
cnn_model.add(layers.MaxPooling2D(2,2))
cnn_model.add(layers.Conv2D(128,(3,3) ,padding ='same',activation='relu'))
cnn_model.add(layers.MaxPooling2D(2,2))
cnn_model.add(layers.Conv2D(256,(3,3) ,padding ='same',activation='relu'))
cnn_model.add(layers.MaxPooling2D(2,2))
cnn_model.add(BatchNormalization())
cnn_model.add(layers.Conv2D(256,(3,3) ,padding ='same',activation='relu'))
cnn_model.add(layers.MaxPooling2D(2,2))
cnn_model.add(BatchNormalization())
cnn_model.summary()

cnn_model.add(layers.Flatten())
cnn_model.add(layers.Dense(1024, activation='relu'))
cnn_model.add(BatchNormalization())
# cnn_model.add(Dropout(0.7))
cnn_model.add(layers.Dense(512, activation='relu'))
cnn_model.add(BatchNormalization())
# cnn_model.add(Dropout(0.3))
cnn_model.add(layers.Dense(2, activation ='softmax'))
cnn_model.summary()
plot_model(cnn_model, to_file='model.png')

opt = tf.keras.optimizers.Adam(learning_rate=0.01)
cnn_model.compile(optimizer=opt, loss="binary_crossentropy", metrics=['accuracy'])

n_training_samples = len(train)
```

```
n_validation_samples = len(validate)
history = cnn_model.fit(
    train,
    epochs=60,
    validation_data=validate,
    validation_steps=n_validation_samples//batch_size,
    steps_per_epoch =n_training_samples,
    shuffle = True,
    # callbacks=[checkpoint]
    # callbacks = callbacks_list
    )

score, acc = cnn_model.evaluate(validate,batch_size=batch_size)
print('Test score:', score)
print('Test accuracy:', acc)

cnn_prediction = cnn_model.predict(validate)
cnn_prediction

plt.plot(history.history['accuracy'], label='accuracy')
# plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.plot(history.history['loss'], label = 'loss')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')
plt.show()
```

```
#get image path
test_images = os.listdir(desitination+'/'+'testing/total')

#creat data frame
df_test =pd.DataFrame({
    'image': test_images,
})

df_test.head()

#prepare generator
test_data_gen  = ImageDataGenerator( rescale = 1./255, ).flow_from_dataframe(
      df_test,
      desitination+'/'+'testing/total',
      x_col='image',
      y_col= None,
      target_size=(128,128),
      class_mode=None,
      shuffle=True,
      batch_size=batch_size
      )

predection = cnn_model.predict(test_data_gen)
predection
tf.nn.softmax(predection[0])
```

```
test_list1 = os.listdir(test_path_total)
y_true = [1,1,1,1,1,1,1,1,1,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1]

test_list =pd.DataFrame({
    '1': test_list1,
    '2':y_true
 })

test_list

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
test_path_black = desitination+'/testing/black'
test_path_white = desitination+'/testing/white'
test_path_pink = desitination+'/testing/pink'
test_path_green = desitination+'/testing/green'
test_path_non = desitination+'/testing/non'
test_path_total = desitination+'/testing/total'
def get_Label(number):
   labels = {0:'NonMask', 1:'Mask'}
   return labels[number]

y_pred = []

for i in os.listdir(test_path_black):
  pic_predict = os.path.join(test_path_total,i)
  img = tf.keras.utils.load_img(
```

```
    pic_predict, target_size=(128, 128)
)
# print(i)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)

predictions = cnn_model.predict(img_array)
score = tf.nn.softmax(predictions[0])
# print(score)
cv2.imread(pic_predict)
print("This image {} with a {:.2f} percent confidence.".format(get_Label(int(np.argmax(score))),
100 * np.max(score)))
y_pred.append(int(np.argmax(score)))

img_show = cv2.imread(pic_predict)
cv2_imshow(img_show)
y_pred

confusion_matrix(y_true, y_pred)

target_names = ['NonMask','Mask']
print(classification_report(y_true, y_pred, target_names=target_names))

# get label Name
def get_Label(number):
   labels = {0:'NonMask', 1:'Mask'}
   return labels[number]
```

```
#plot predction function
def plot_prediction(model_name):
    plt.figure(figsize=(20,15))
    plt.suptitle("Predection  Images", fontsize=20)
    images = []
    path =desitination+'/'+'testing/'
    count = 0  #val_images,val_labels
    for i,files in enumerate(os.listdir(path)) :
        img = plt.imread(path+files)
        img = cv2.resize(img,(128,128))
        plt.imshow(img,cmap=plt.cm.binary)
        img = np.expand_dims(img, axis=0)
        feature = model_name.predict(img)
        predection  = np.argmax(feature, axis=1)
        plt.subplot(5,7,i+1)
        plt.xticks([])
        plt.yticks([])
        plt.grid(False)
        plt.xlabel("Predicted : "+get_Label(int(predection)))
        count += 1
        if count == 34 :
            break

plot_prediction(cnn_model)
```

ResNet-Mask_classification.ipynb

```
!pip install keras
```

```
import cv2
from imutils import face_utils
import imutils
import shutil
import dlib
import os
import seaborn as sns
import pandas as pd
import numpy as np
import tensorflow as tf
from google.colab.patches import cv2_imshow
from google.colab import drive
from sklearn.model_selection import train_test_split
import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPool2D , Flatten
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam, SGD, RMSprop
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import datasets, layers, models
from tensorflow.keras.layers import Flatten, Dense,BatchNormalization,Dropout,Input
import matplotlib.pyplot as plt
```

```
drive.mount('/content/gdrive')
desitination = "{path}/dataset/"

train_path = desitination+'train'
val_path = desitination+'val'
all_categories = []
all_filename = []


images = os.listdir(train_path)
for index,r in enumerate(images):
  if r != 'MaskNonMask':
    read = images[index]
    read = train_path+'/'+read
    print(read)
    print(r)
    for i,file in enumerate(os.listdir(read)):
      if r == 'Mask':
        all_filename.append(file)
        all_categories.append(1)
      else:
        all_filename.append(file)
        all_categories.append(0)
  else :
    None
    # print(all_filename,all_categories)
```

```
# #creat data fram to save each image with its label
df =pd.DataFrame({
    'image': all_filename,
    'category':all_categories
})

plt.title('Train Labels Visualization')
sns.countplot(x=all_categories,palette='flare')
plt.show()

df['category'] = df['category'].replace({1:'Mask',0:'NonMask'})

x_train,x_val  = train_test_split(df ,random_state=42,shuffle=True,test_size=0.2)

x_train = x_train.reset_index(drop=True)
x_val = x_val.reset_index(drop=True)
print('Train Images shape is       : ',x_train.shape)
print('Validation  Images  shape is : ',x_val.shape)

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Conv2D, Input, ZeroPadding2D, BatchNormalization,
Activation, MaxPooling2D, Flatten, Dense
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.callbacks import TensorBoard, ModelCheckpoint
from tensorflow.keras import optimizers
```

```
batch_size = 8
#create image generator for images
train = ImageDataGenerator(
                           rescale = 1.0/255,
                            shear_range = 0.2,
                            zoom_range = 0.2,
                            height_shift_range=0.2,
                           width_shift_range=0.2,
                            fill_mode='nearest',
                             horizontal_flip=True,
                            rotation_range = 40,

                        ).flow_from_dataframe(
    x_train,
    desitination+'/'+'train/MaskNonMask/',
    x_col='image',
    y_col='category',
    target_size=(224,224),
    class_mode='categorical',
    shuffle=True,
    batch_size=batch_size
    )
```

```
# Create Image Data Generator for Test/Validation Set
```

```
validate = ImageDataGenerator(
                                rescale = 1.0/255,
                               shear_range = 0.2,
                               zoom_range = 0.2,
                               height_shift_range=0.2,
                               width_shift_range=0.2,
                               fill_mode='nearest',
                                horizontal_flip=True,
                                rotation_range = 40,).flow_from_dataframe(
    x_val,
    desitination+'/'+'train/MaskNonMask/',
    x_col='image',
    y_col='category',
    target_size=(224,224),
    class_mode='categorical',
    shuffle=True,
    batch_size=batch_size
    )


checkpoint = ModelCheckpoint('model-
{epoch:03d}.model',monitor='val_loss',verbose=0,save_best_only=True,mode='auto')


from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.models import Model
from tensorflow.keras.utils import plot_model
IMG_WIDTH=224
```

```
IMG_HEIGHT=224
IMG_DIM = (IMG_WIDTH, IMG_HEIGHT)
restnet = ResNet50(include_top=False, weights='imagenet',
input_shape=(IMG_HEIGHT,IMG_WIDTH,3))
output = restnet.layers[-1].output
output = keras.layers.Flatten()(output)
restnet = Model(restnet.input, output)
for layer in restnet.layers:
    layer.trainable = False
restnet.summary()
plot_model(restnet, to_file='model.png')

model = Sequential()
model.add(restnet)
model.add(Dense(512, activation='relu', input_dim=(IMG_HEIGHT,IMG_WIDTH,3)))
model.add(Dropout(0.3))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(lr=2e-5),
              metrics=['accuracy'])
model.summary()

model.add(tf.keras.layers.Dense(2, activation='softmax'))
```

```
# from tensorflow.keras.applications import ResNet50V2
from tensorflow.keras.utils import plot_model
model.summary()
plot_model(model, to_file='model.png')


history = model.fit(train,
                    epochs=60,
                    validation_data=validate
                    ,callbacks=[checkpoint])


loss = history.history['loss']
val_loss = history.history['val_loss']
acc = history.history['accuracy']


val_acc = history.history['val_accuracy']
epochs = range(1, len(loss) + 1)


# plotting accuracy

plt.plot(epochs, acc, color='blue', label='Training')
plt.plot(epochs, val_acc, color='green', label='Validation')
plt.title('Training and Validation accuracy')
```

```
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

# plotting loss


plt.plot(epochs, loss, color='orange', label='Training')
plt.plot(epochs, val_loss, color='red', label='Validation')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()


#get image path
test_images = os.listdir(desitination+'/'+'testing/total')

#creat data frame
df_test =pd.DataFrame({
    'image': test_images,
})


df_test.head()
```

```
#prepare generator
test_data_gen  = ImageDataGenerator( rescale = 1./255, ).flow_from_dataframe(
    df_test,
    desitination+'/'+'testing/total',
    x_col='image',
    y_col= None,
    target_size=(224,224),
    class_mode=None,
    shuffle=True,
    batch_size=batch_size
    )

predection = model.predict(test_data_gen)
predection
tf.nn.softmax(predection[0])

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
test_path_black = desitination+'/testing/black'
test_path_white = desitination+'/testing/white'
test_path_pink = desitination+'/testing/pink'
test_path_green = desitination+'/testing/green'
test_path_non = desitination+'/testing/non'
test_path_total = desitination+'/testing/total'
def get_Label(number):
```

```
    labels = {0:'NonMask', 1:'Mask'}
    return labels[number]

y_pred = []

test_list1 = os.listdir(test_path_total)
y_true = [1,1,1,1,1,1,1,1,1,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1]

test_list =pd.DataFrame({
    '1': test_list1,
    '2':y_true
 })

test_list

for i in os.listdir(test_path_total):
 pic_predict = os.path.join(test_path_total,i)
 img = tf.keras.utils.load_img(
     pic_predict, target_size=(224, 224)
 )
 # print(i)
 img_array = tf.keras.utils.img_to_array(img)
 img_array = tf.expand_dims(img_array, 0)

 predictions = model.predict(img_array)
 score = tf.nn.softmax(predictions[0])
```

```
    # print(score)
    cv2.imread(pic_predict)
    print("This image {}".format(get_Label(int(np.argmax(score)))))
    y_pred.append(int(np.argmax(score)))

    img_show = cv2.imread(pic_predict)
    cv2_imshow(img_show)
y_pred

confusion_matrix(y_true, y_pred)

target_names = ['NonMask','Mask']
print(classification_report(y_true, y_pred, target_names=target_names))

def get_Label(number):
    labels = {0:'NonMask', 1:'Mask'}
    return labels[number]

def plot_prediction_pretrain(model_name):
    plt.figure(figsize=(20,15))
    plt.suptitle("Predection  Images", fontsize=20)
    images = []
    path =desitination+'testing/'
    count = 0  #val_images,val_labels
    for i,files in enumerate(os.listdir(path)) :
        img = plt.imread(path+files)
        img = cv2.resize(img,(224,224))
```

```
        plt.imshow(img,cmap=plt.cm.binary)
        img = np.expand_dims(img, axis=0)
        feature = model_name.predict(img)
        predection  = np.argmax(feature, axis=1)
        plt.subplot(5,7,i+1)
        plt.xticks([])
        plt.yticks([])
        plt.grid(False)
        plt.xlabel("Predicted : "+get_Label(int(predection)))

plot_prediction_pretrain(model)
```

# AUTHOR BIO

| | |
|---|---|
| **Name** | Mr. Chetsada Chongthanachote |
| **Birthdate** | 6 April 1995 |
| **Education** | 2018 Bachelor of Science in Statistics GPA : 2.85 |
| | King Mongkut's Institute of Technology Ladkrabang |