Image Classification of Facemask using TensorFlow

Raihan Astrada Fathurrahman 13519113
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: raihan.astrada@gmail.com

***Abstract*—Image classification is the task of identifying what an image represents. An image classification model will extract features from the image and observe some pattern from it. One of the Model that is used in Image Classification was Convolutional Neural Network (CNN). By utilizing CNN as the main part of image classification, image classification can now be widely used for a range of applications. One of the uses of image classification is to recognize whether someone is wearing a mask or not. The use of facemask is very useful for prevention of COVID-19 spread. This classification could help check the use of facemask automatically & help reduce the infection rate of COVID. This paper will discuss further about designing an image classification model using CNN on TensorFlow, an end-to-end open-source platform for machine learning.**

Keywords—Image Classification; Convolutional Neural Network; COVID-19; Facemask; TensorFlow;

I. Introduction

The Coronavirus disease (COVID-19) has various ways to spread, such as through droplets and contaminated surfaces. To prevent the spread from one person to another, each country has taken several mitigation steps as optimally as possible so that this virus does not spread further. Various policy was taken to minimize the rate of spread, such as performing a lockdown, limiting distance (social distancing), and applying self-isolation rule after traveling from/to abroad.

One of the simplest and easiest way for people to prevent getting COVID-19 is by using a facemask. Wearing face masks is recommended as part of personal protective equipment and as a public health measure to prevent the spread of COVID-19. It can prevent COVID-19 spread since coronavirus can spread through droplets and particles released into the air by speaking, singing, coughing, or sneezing.

Due to the importance of facemasks, many countries have required the use of face masks in various public places, including Indonesia. As facemasks have become very important during pandemics, many public places need to monitor someone for wearing them. However, Indonesia's public places still apply manual checking, thus requiring more resources to assign staff at each entrance of public places. Therefore, a machine is needed to classify whether someone is wearing a mask or not.

In this paper, a simple classification problem solution will be given to classify whether someone uses a mask. This paper will focus on creating a classification model using a Convolutional Neural Network on TensorFlow technology.

II. Theory

A. Classification Problems

Classification is the process of predicting the class of given data. Classification task will approximate a mapping function from input variables to discrete output variables. The target class can be either a binary classification or a multi-class classification.

In the classification, there are two type of learners, lazy learners & eager learners. In lazy learner, classifier firstly stores the training dataset and wait until it receives the test dataset. This learner takes less time in the training process but more time on the prediction. The example of this learner is K-NN Algorithm and Case-based reasoning.

The next learner, eager learners, will develop a classification model based on a training dataset before receiving a test dataset. Contrary to lazy learners, eager learner will take more time in learning and less time in prediction. The example of this learner is Decision Trees, Naïve Bayes, and Artificial Neural Network (ANN).

B. Convolutional Neural Network

A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data. A neural network is a machine learning algorithm created to imitate the working operation of neurons in the human brain. Traditional neural networks are not ideal for image processing and must be fed images in reduced-resolution pieces.

CNN "neurons" is arranged like the frontal lobe area, the area that's responsible for processing visual stimuli in humans. The layers of neurons are organized in such a way to cover the entire visual field avoiding the image processing problem of traditional neural networks. A CNN uses a system much like a multilayer perceptron that has been designed to reduce processing requirements. The layers of a CNN consist of an input layer, an output layer, and a hidden layer that includes multiple convolutional layers, pooling layers, fully connected layers, and normalization layers. This results in a system that is more effective, and simpler to trains limited for image processing and natural language processing.

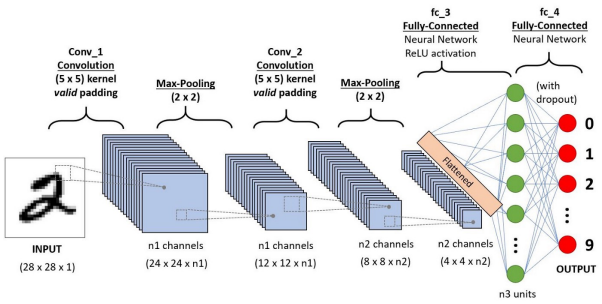

Fig. 1. Convolutional Neural Network *(Source:* Towards Data Science — A Comprehensive Guide to Convolutional Neural Networks*)*

III. FACEMASK CLASSIFICATION

The facemask classification will consist of two parts, preprocessing the image, and creating the model. For facemask classification, this implementation will use the Facemask Detection Dataset 20000 Images [1]

A. Image Preprocessing

To preprocess the image, we are going to use the ImageDataGenerator module from TensorFlow. To do that, we need to create a training and testing directory, and inside those directories we need to create a directory for each class, in the facemasks detection case the labels are 'mask' and 'no mask'. After the directories are created, we need to split the data to training and testing datasets. In the program, we use 90% as training data and 10% as testing data, so the datasets are as follow:

TABLE I. DATA SPLITTING RESULTS

	Class	
	Mask	**No Mask**
Training	9000	9000
Testing	1000	1000

Those data are going to flow from the directory in batches and will be augmented by the ImageDataGenerator. To do that, we'll use the code as follow:

```
def train_val_generators(TRAINING_DIR, VALIDATION_DIR):
 train_datagen = ImageDataGenerator(rescale = 1.0/255.,
                    rotation_range = 45,
                    width_shift_range = 0.2,
                    height_shift_range = 0.2,
                    shear_range = 0.2,
                    zoom_range = 0.2,
                    brightness_range = [0.2,0.2],
                    horizontal_flip = True,
                    fill_mode = 'nearest')
 train_generator =
train_datagen.flow_from_directory(directory=TRAINING_DIR,
                                  batch_size=32,
                                  class_mode='binary',
                                  color_mode='grayscale',
                                  target_size=(150, 150))
 validation_datagen = ImageDataGenerator(rescale = 1.0/255.)
 validation_generator =
validation_datagen.flow_from_directory(directory=VALIDATION_DIR,
                                       batch_size=32,
                                       class_mode='binary',
                                       color_mode='grayscale',
                                       target_size=(150, 150))
 return train_generator, validation_generator
```

The ImageDataGenerator class will augment the data based on the given parameters. On the code above, the training image will be normalized, rotated, shifted in weight & height, sheared, zoomed, had a brightness shift, and flipped horizontally. This augmentation is done to get various training data. That's why on the validation_datagen we only normalize the image, not transforming the data. Next, the flow_from_directory function will specify the location of the image source and will send the (150 x 150) sized image through 32 batches where the class is binary ('mask' and 'no mask') and the color is greyscale because the training data is monochrome.

B. CNN Model

After the image is processed, we then define the CNN model that will receive the processed image. To do that, we'll use the code as follow:

```
def create_model():
 model = tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(150, 150,
1)),
  tf.keras.layers.MaxPooling2D(2,2),
  tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
  tf.keras.layers.MaxPooling2D(2,2),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(1, activation='sigmoid')
 ])
 model.compile(optimizer=RMSprop(learning_rate=0.001),
          loss='binary_crossentropy',
          metrics=['accuracy'])
```

```
  return model

model = create_model()
history = model.fit(train_generator,
            epochs=5,
            verbose=1, validation_data=validation_generator,
            )
```

The first layer is the Convolutional layer (Conv2D). The parameters are the number of filters in the convolution, the kernel size or the height and width of the 2D convolution window, and ReLU as the activation function. The input size is the same as the previous, which is (150 x 150) and 1 because the image is grayscale. The Pooling layer will down samples the input along its spatial dimensions (height and width) by taking the maximum value over an input window for each channel of the input. Beside Convolutional & Pooling layers, there is also Flatten layer that will flatten the 2D Array to 1D. Those 1D Array will then be fed into the Dense Layer with specified units. The last layer is an output layer. It consists of 1 unit and use sigmoid function because the problem is binary classification. The model then compiled using Root Mean Squared optimizer, Binary Cross entropy loss function, and measure its accuracy. Then train the model using the generator that was defined on the previous part and train it for 5 epochs.

IV. Conclusion

The results of the training as follow:

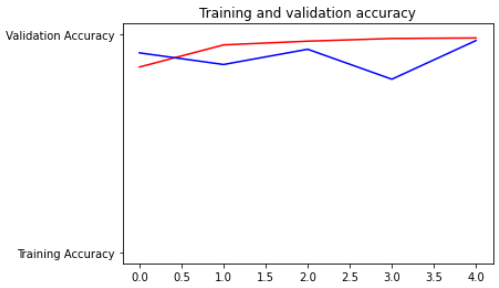

Fig. 2. Training and Validation Accuracy Results

From the figure above, after 5 epochs, the training accuracy reached 0.9842 and validation accuracy reached 0.9725. This is a pretty good training result but let's try the prediction of the model using unseen image.

Fig. 3. Facemask Classification Model Prediction Results

As seen above, there are 3 pictures of people using mask that is misclassified. This happens because the training data only consist of forward-looking face, thus the side looking is misclassified. This could be resolved by adding more various data on the dataset

Acknowledgment

First, I would like to say thanks to God that I am able to finish this paper, and for everything I have been given in life and the gift of life itself. Also, thanks to both of my parents, who always support me through the ups and downs. And, to all of my lecturer, especially Dessi Puji Lestari, S.T., M.Eng., Ph.D who has guide me in the study of Socio-informatics and Professionalism this semester, may all the lessons learnt, will be useful for years to come. Lastly, to all my friend, whose acts of kindness and support, no matter how small, shall not go unnoticed.

Link

The source code of the program can be accessed through: https://colab.research.google.com/github/raihanastrada/facemask-image-classifier/blob/main/facemask_image_classifier.ipynb

References

[1] Singaraju, J.P. and Lavik, J. (2019). Facemask Detection Dataset 20000 Images, Retrieved May 1, 2021 from https://www.kaggle.com/datasets/pranavsingaraju/facemask-detection-dataset-20000-images.

[2] Contributor, T. (2018, April 27). *convolutional neural network*. SearchEnterpriseAI. Retrieved May 1, 2022, from https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network

[3] Saha, S. (2021, December 7). *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. Medium. Retrieved May 1, 2022, from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[4] *Classification Algorithm in Machine Learning - Javatpoint*. (n.d.). Www.Javatpoint.Com. Retrieved May 1, 2022, from https://www.javatpoint.com/classification-algorithm-in-machine-learning

[5] *All symbols in TensorFlow 2 | TensorFlow Core v2.8.0*. (n.d.). TensorFlow. Retrieved May 1, 2022, from https://www.tensorflow.org/api_docs/python/tf/all_symbol