FACE MASK DETECTION USING CONVOLUTIONAL NEURAL NETWORK

Mohd Razif Bin Shamsuddin
Faculty of Computer and Mathematical Sciences
Universiti Teknologi MARA
Shah Alam, Selangor, Malaysia
m.razif@uitm.edu.my

Muhammad Hafiz Bin Mohd Nasarudin
Faculty of Computer and Mathematical Sciences
Universiti Teknologi MARA
Shah Alam, Selangor, Malaysia
hafizn24@gmail.com

***Abstract*— COVID-19 virus has affected people around the world and it is known for its potential deadly symptoms if not treated or prevented using vaccines. This virus spreads very fast and easily as it can be airborne. One of the practical methods and solutions to stop the virus from spreading is simply using a facemask. However, many people have the tendency to wear a face mask incorrectly or in a worse case scenario they do not wear it at all. Our work proposes a way to implement a surveillance like system to identify and recognize people wearing facemask. The development of this surveillance system is in hope that it can be used in premises so it can help authorities identify people who is wearing facemask incorrectly. The system is able to detect a face mask by implementing a CNN classification algorithm. This study focuses two CNN model with different optimizer trained with an image dataset that is labelled in two different class labels where one classes identifies as people who wears facemask correctly and the other one identifies it as people who wears facemask incorrectly. Furthermore, parameter tuning is implemented to increase the accuracy of the CNN model. Finally, based on the tuned model's performance, the model with highest accuracy was selected to be used in the surveillance system. In this study, the highest accuracy is produced by the CNN model with Adamax optimizer that achieves 99% training accuracy.**

Keywords—convolutional neural network, machine learning, image recognition, face mask detection (*key words****)***

I. INTRODUCTION (*HEADING 1*)

The first recorded coronavirus case was in 1960. A study conducted in 2001 by a Canadian study found that about 500 patients with coronavirus had the same symptoms like flu. The report published in 2003 stated that coronavirus is widespread in many countries. The United States, Hong Kong, and some south Asian countries are some countries that are included (P. Sharma et al., 2021). As of early 2020, the World Health Organization (WHO) announced that a new coronavirus named 2019-nCoV had spread worldwide, and a pandemic had been declared by WHO [1].

COVID-19 can be transmitted from person to person. The virus travels through a small liquid particle that has been released into the air from coughing, sneezing, speaking or breathing. Experts currently identify seven types of coronavirus that humans can infect. The common types are 229E (alpha coronavirus), NL63 (alpha coronavirus), OC43 (beta coronavirus), and HKU1 (beta coronavirus) (P. Sharma et al., 2021). COVID-19 common symptoms are fever, cough, and fatigue. According to recent studies, most patients that COVID-19 has infected are middle-aged men with underlying diseases[1]. The COVID-19 pandemic caused substantial negative impacts on the world economy. Loss of income and high unemployment are some of the impacts of this pandemic. Another impact of this pandemic is high mortality rates, especially in countries with low GDP (Gross Domestic Product) per capita, like India and Mexico. follow.

Health measures published by WHO stated that people need to keep the physical distance from other persons at least 1 metre apart; avoid visiting crowded places that may be hotspots of the spread of coronavirus. Wash hands with soap and water, or use hand sanitiser when contacting foreign objects; Wear face masks when going to public places to avoid spreading the virus [2]. As of November 2021, Malaysia is still in the recovery phase from this pandemic. More than 75% of Malaysians already have taken the vaccine and the number of Malaysians that take the vaccine is still growing. The government takes many initiatives to minimise the spread of coronavirus in Malaysia.

One of the earliest initiatives required any persons entering the establishment to have their temperature checked; Only allow specific industries to operate on the total lockdown durations. The government also introduced an application called mySejahtera. This application tracks the Malaysian movement to notify their risk status and as digital certificates from taking the vaccines. From what we experience, we can conclude that the COVID-19 pandemic is the largest health crisis of this generation (Ministry of Health Malaysia, 2021). Artificial neural networks are techniques of simulating a learning process from biological organisms. The components of artificial neural networks are inspired by the components of the nervous system of the brain. The process started in the input layer where the neurons receive the information. The information went through multiple layers of neurons to produce a result. Artificial neural networks help computers to learn and decide on their own.

Convolutional neural networks is a technique used mostly to learn unstructured data like image, voice, or audio input. CNN consists of three major layers. The first layer is the convolution layer; this layer adds filters into input data to make the information in uniform manner. The second layer is the pooling layer; the layer averages the value for the square pixel area of the data. The information from the pooling layer has fewer variables to be considered. The information goes through neural networks layers to be processed [3].

II. RELATED WORKS

Before continuing with the research, we tried to identify few related works on face mask detection algorithm that uses machine learning as its classification model. Here are a few notable researches that has a very interesting approach and

produce good results in its classifications. The propose model for this study is the Convolutional Neural Network (CNN) technique.

A. Convolutional Neural Network

CNN is widely used for image recognition because of its ability to recognize object in images and video feeds. CNN has three main components for processing images. Each layer of neurons in a CNN performs a nonlinear operation on the linear outputs of the layer before it. There are a lot of convolutional and pooling layers in this algorithm. Pooling layers convert activation using a fixed function while convolutional layers include weights that must be learnt. Hence, below are the main components of CNN model to process and recognize object in images [4].

1) Convolutional Layer : A CNN's main building block is a convolutional layer. It consists of a set of filters (or kernels), the parameters of which must be learned throughout the training process. Filters are more compact in terms of size and weight than the input volume itself is. The input volume is convolved with each filter to produce a neuron activation map. The filters are typically smaller in size than the actual image. Each filter interacts with the image to generate an activation map. The filter is slid over the image's height and breadth, and the dot product between each filter element and the input is determined at every spatial point. Each time the filter is moved across the width and height of an input, it does a dot product calculation at each spatial location. To acquire the output volume of the convolutional layer, the activation maps of all filters along the depth dimension must be stacked one on top of another. This means that every neuron in the activation map is only linked to a tiny area of input volume because of the design of each filter. In other words, each neuron has a tiny receptive field, which is matched by its filter size. When convolution is used for activation mapping, the filter's parameters are shared for all local locations. To improve expressiveness, learning, and generalisation, the number of factors is reduced by weight sharing. In the convolutional layers, the information of the image is represented in the form of a parameter. These layers reduce the parameter by sharing the exact weight and bias across the connected neurons into the same channel. This layer reduces any noise from the image dataset by reducing the parameter. Finally, this layer automatically tunes the image dataset based on the image itself [4].

2) Pooling Layer : The convolutional layer is followed by a pooling layer. A nonlinearity (such as ReLU) is applied to the feature maps generated by a convolutional layer after they have been processed. Pooling layers simply reduce the size of an image data, and there is no parameter training performed on it. This layer reduces image data size by combining a number of kernels into a single kernel by using a predefined function. A popular pattern for layer ordering in a convolutional neural network is to add a pooling layer after the convolutional layer. This pattern can be repeated several times in a given model. Each feature map in the pool is processed independently by the pooling layer, resulting in a new set of the same number of combined feature maps. Feature maps may be filtered by applying a pooling operation, which is similar to applying a filter. The pooling operation or filter is generally often applied with a stride of 2 pixels, which is smaller than the size of the feature map. Since each feature map's dimensions are halved and its pixels and values are cut in half, the pooling layer reduces the overall size of each feature map to one-fourth its original size. When a pooling layer is added to a feature map with a resolution of 66, the resulting pooled feature map has a resolution of 33. (9 pixels).

Pooling is stated rather than taught. The most common pooling functions are max-pooling and average-pooling, the following are examples of the two often used functions:

- **Max-pooling**: Each patch on the feature map should have an average value calculated.
- **Average-pooling**: The maximum value for each patch of the feature map is calculated using maximum pooling (or max pooling).

Max-pooling works by calculating the maximum values from the combining of the kernels. For the average pooling, the average values from the combining of the kernels is calculated. Reducing the size of an image helps reduce the processing power to train the model (Planche, B., & Andres, E., 2019).

3) Fully connected Layer : Fully Connected Layer is simply, feed forward neural networks which are nonlinear models. Fully Connected Layers constitute the last few layers of the network. These layers help determine which category of the data image resulted belong. The output from the last pooling or convolutional layer is passed into the fully connected layer, where it is flattened before being applied. A few fully connected layers that are similar to artificial neural networks and carry out the same mathematical processes are then linked to this flattened vector. Fully connected layers have three main components. The three components are initialization, activation, and weight training. The data is reshaped into column vector form before the processed image goes through fully connected layers (Planche, B., & Andres, E., 2019).

B. Face Mask Detection Methods and Techniques: A Review

In this article, the author covered a several similar studies on facial mask identification. According to the author, detecting masks is an extremely difficult process. Applications of facial mask detection include anti-spoofing, monitoring and identifying criminals, and stopping the transmission of the Corona Virus, among others. Each of the reviewed publications employs a different sort of algorithm, strategy, or approach, but they all aim to recognize faces and facial characteristics like eyes, noses, and brows as well as determine whether or not a person's face is hidden by a mask. After thoroughly examining each algorithm, the author have come to the conclusion that each method has advantages and disadvantages of its own. However, when compared to the other algorithms, the YOLO algorithm provides superior results with more precision and is more effective in practical applications.

C. *Multiscale parallel deep CNN (mpdCNN) architecture for the real low-resolution face recognition for surveillance*

This study focuses on creating a CNN model for face recognition using a low-resolution input image. Images for criminal investigation tends to have low-resolution image quality. This drawback cause difficulty in recognising the criminal. Low-resolution images have much noise that may affect the model to recognise the criminal inside the image. The authors used the low-resolution images dataset for this study rather than the downsampled dataset. This is because the downsampled dataset has many features for the model to recognise compared to the low-resolution dataset. Using a low-resolution dataset helps the model to learn from the real-world scenario. This study concludes that the proposed model has 86% accuracy on the low-resolution test image and achieves 99% accuracy if the model recognises the image from the normal to the high-resolution dataset [5].

TABLE I. TRAIN ACCURACY AND TEST ACCURACY OF THE DIFFERENT CLASSIFICATION AND SEGMENTATION MODELS

CNN Optimizers	Recognition accuracy in %			
	ELU	ReLU	tanh	Leaky ReLU
SGD	87.13	80.75	85.66	74.67
Adagrad	85.11	72.6	69.85	84.01
Adam	85.29	70.22	84.74	86.4
Adadelta	88.6	80.88	85.29	88.6
Adamax	88.6	80.15	86.58	88.6
RMSProp	86.76	77.94	84.01	87.13

[a.] excerpt from (N. K. Mishra et al, 2021).

[b.]

III. METHODOLOGY

The process for creating the model and the system for classifying face masks will be covered in this section. This section will include a list of all the procedures necessary to complete the work in order to meet its goals. Face recognition without a mask is easier, but face recognition with a mask alone is more challenging due to the difficulty of masked face feature extraction compared to conventional face feature extraction. The covered face lacks several facial features, including the chin, lips, and nose. Our study will train eight different models based on two optimizers

A. *Data Preparation*

For this project, Kaggle and GitHub were used to find the suitable dataset structure for model training. Having viewed numerous datasets related to face mask identification on Kaggle, one dataset that stood out had three classes (with mask, without mask, and wrongly wearing mask); nonetheless, the dataset was severely unbalanced and uncleaned. To enhance this dataset, photos have to be updated such that each class has an equal distribution of images and noisy images that may be regarded outliers were removed. As a result, the dataset is a hybrid of an existing dataset that has been cleansed and distributed evenly throughout each classes. We would like to acknowledge that the dataset for the training of our model is taken from Kaggle. The dataset obtained in this work can be referred to this url:

- https://www.kaggle.com/datasets/vijaykumar1799/face-mask-detection
- https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset
- https://www.kaggle.com/andrewmvd/face-mask-detection

The Kaggle dataset consists of approximately 3,000 images for each class. The classes are “face image”, ‘wearing face mask”, and “wearing incorrectly”. The Kaggle dataset provides highly imbalanced and low-resolution images to be used alone for the model training. The diagram below shows an example of the images from the Kaggle dataset.

Fig. 1. Example of images of “wearing incorrectly” class from the Kaggle dataset training samples

The model will only be trained with “Incorrect wearing” and “Face mask-wearing” classes since the program will be capable of detecting human faces without using the model. The number of images used is 3,000 for each class. One thousand five hundred from the Kaggle dataset and 1,500 from the GitHub dataset for the “Face mask-wearing” class. For the “Incorrect wearing” class, 3,000 images from the GitHub dataset only since the Kaggle dataset for the “Incorrect wearing” class is so unclean and unbalanced.

B. *CNN Model using Tensorflow with Keras*

Keras has been designated as the official high-level API for TensorFlow as of TensorFlow 2.0. It is an open-source program that has been included into TensorFlow to accelerate the development of deep learning models. It may be found at 'tf.keras'. This is what we will use in our works and model training. The model training is executed in python3 environment and having tensorflow as its imported library. Assuming that tensorflow has been correctly imported and loaded the dataset is loaded in the code as follows.

```
1  url = pathlib.Path().parent.resolve()
2  dir_name = os.path.join(url, 'dataset')
3
4  data_dir = pathlib.Path(dir_name)
5  batch_size = 20
6  img_height = 128
7  img_width = 128
8
9  train_ds=tf.keras.preprocessing.image_dataset_
10 from_directory(
11    data_dir,
12    validation_split = 0.2,
13    subset = 'training',
```

```
    seed = 100,
    image_size = (img_height, img_width)
)
```

Fig. 2. Python code to load data and set training parameters

From what is shown in figure 2. The data is loaded in the path directory with the image size set at 128pixel x 128 pixel. This will be crucial as it will determine the input tensor to the CNN later. The code also utilize other parameter to ease in any changes to the model such as batch size, validation split and number of seed. The convolutional neural network is then defined. The convolution, pooling, and flattening layers will be applied using keras.Sequential method. The first layer is called 'Conv2D.' It is specified by the following parameters:

- 256 output filter with input_shape (128,128,3) for the first convolutional layer
- Feature detector is set at 3 by 3
- a `pool_size` of (2, 2) that defines the size of the pooling window

The pooling window has two strides, which dictate the amount of steps it takes. Keep in mind that we may construct our network we set and fine tune all these parameters. We only need to monitor the stats and change the design until find the one that produces the greatest results. Another convolution and pooling layer is built in this situation. The flatten layer comes next, with the results being transmitted to the dense layer. Because the dataset comprises two classes, the final layer has two units. Below is an example of how the sequential layers are set up in the python code as shown in figure below.

```
model = keras.Sequential(
    [
        data_augmentation,
        layers.experimental.preprocessing.Rescaling
(1./255, input_shape=(img_height, img_width, 3)),
        layers.Conv2D(256, (3,3), activation='relu',
input_shape = (img_height, img_width, 3)),
        layers.MaxPooling2D((2, 2)),

        layers.Conv2D(128, (3,3), activation='relu'),
        layers.MaxPooling2D((2, 2)),

        layers.Conv2D(128, (3,3), activation='relu'),
        layers.MaxPooling2D((2, 2)),

        layers.Conv2D(256, (3,3), activation='relu'),
        layers.MaxPooling2D((2, 2)),

        layers.Conv2D(128, (3,3), activation='relu'),
        layers.MaxPooling2D((2, 2)),

        layers.Flatten(),
        layers.Dropout(0.25),
        layers.Dense(256, activation='relu'),
        layers.Dense(2),
    ]
)
```

Fig. 3. Python code to set up the CNN layers using keras.Sequential method

Because the pictures are 128 by 128, the 128 output filters and 3 by 3 feature detector'same' padding result in even padding for the input input shape of '(128, 128, 3)'. 3 informs the network that pictures are coloured using the 'relu' activation function in order to achieve non-linearity. Dropout regularisation is a frequent method for improving the performance of deep learning models. During the training phase, a certain number of connections are discarded. This challenges the network to discover patterns from data rather than memorise it. Overfitting is reduced as a result of this. This may be accomplished in Keras by putting a Dropout layer within the network. All this is set up in the sequential layer set up as shown in figure 3.

Finally, the model must now be compiled. Because the labels are not encoded in a single pass, the Sparse Categorical Cross-Entropy loss is utilised. If you wish to encode the labels, you'll need to utilise the Categorical Cross-Entropy loss function. The example of the compiled model is as shown in figure 3. This section of code allows us to choose our optimizer as we would train our CNN model later. In this example the optimizer is Adamax.

```
model.compile(
  optimizer=keras.optimizers.Adamax(),
  loss=tf.keras.losses.SparseCategoricalCrossentropy
(from_logits=True),
  metrics=['accuracy']
)
```

Fig. 4. Python code to set up the optimizer and the metrics parameter

The next step is to fit the data to the training set. Because the callback monitors the validation set, the validation set is also passed along the model as it trains. The importance of learning curves lies in their ability to indicate whether a model is learning or overfitting. If the validation loss increases substantially or the validation accuracy decreases dramatically, your model is likely overfitted. Since the model has been preserved in a history variable, you may use it to plot the losses and precision.

```
history = model.fit (train_ds, validation_data =
val_ds, epochs = 3)
```

Fig. 5. Training the model

The model development is a crucial component of this study. It will help create models capable of classifying given input accurately. The CNN models are created by training the models with the preprocessed datasets. The sequential model will be trained with different parameter tunings to increase the variety of produced models, hence increasing the chances of finding an appropriate model for the system. Following is the list of parameter tuning of model development:

- Optimizer (Adam, Adamax)
- Split Ratio (80:20, 70,30)
- Number of Epochs (3,6)

IV. Results and Findings

For the trained image classification model, several evaluation methods are used to evaluate the accuracy of the models. For this study the evaluation approach will be use a

confusion and classification matrix and model prediction accuracy. From these evaluation method, we can choose the suitable model for the system based on the evaluation results. To test the models' performance in the confusion and classification matrix, a new dataset is prepared for evaluation. The dataset follows the same execution based on the training dataset with the remaining images from the Kaggle dataset. The following results are shown in the table below.

TABLE II. TRAINING ACCURACY OF THE DIFFERENT CLASSIFICATION MODELS OF THE CNN ARCHITECTURE

Model	Optimizer	Split ratio	Epoch	Accuracy
1	Adam	80:20	3	0.976
2	Adam	80:20	6	0.998
3	Adam	70:30	3	0.988
4	Adam	70:30	6	0.992
5	Adamax	80:20	3	0.990
6	Adamax	80:20	6	0.994
7	Adamax	70:30	3	0.989
8	Adamax	70:30	6	0.985

Based on the table above, we can conclude that all model are quite good to achieve accuracy above 97%. Probably due to the result of similar created models, the difference of accuracy doesn't make much difference. Further evaluation is needed to find a suitable model from the trained models. We decided to evaluate model 5 as it has the highest training and testing accuracy. Refer to table 2 and table 5 for both training and testing accuracy of the models. Consider a binary classifier with 6000 entries to categorise. 3000 of these objects belong to A (wearing incorrectly), while 3000 belong to B (wearing incorrectly). The following are the results:

TABLE III. CONFUSION MATRIX OF MODEL 5

	wearing incorrectly (prediction)	wearing correctly (prediction)
wearing incorrectly (actual)	2975	25
wearing correctly (actual)	32	2968

Those results will be referred to as class A a negative one and class B a positive one (0). So there were 2975 true negatives, 32 false positives, 25 false negatives, and 2968 genuine positives in the total sample size of 6000. The entire confusion matrix can be viewed as a sample from a multinomial distribution. It is possible to both bootstrap and assume priors on it. A Beta distribution can be assumed to be the realisation of a Binomial distribution, thus we'll use that as our prior for recall. Observe that the F1-score takes into consideration both precision and recall, which means that it also accounts for FPs and FNs. The greater the F1-score, the more accurate and accurate the results. A score of 0 to 1 is called an F1-score. The better the model, the closer it is to 1.

TABLE IV. PRECISION, RECALL F1-SCORE OF THE 5TH MODEL

	Precision	Recall	F1-score	Support
Mask weared incorrect	0.99	0.99	0.99	3000
With mask	0.99	0.99	0.99	3000
Accuracy			0.99	6000
Macro avg	0.99	0.99	0.99	6000
Weighted avg	0.99	0.99	0.99	6000

By examining the prediction accuracy for each model, we can choose the suitable model for the system based on the model's accuracy in classifying the image. For the prediction accuracy, the model is tested with three different images representing the "Incorrect wearing", "Face mask-wearing" and "Not wearing a mask" classes. Even though the models be trained for the "Incorrect wearing" and the "Face mask-wearing" classes, the capability of the models to know the "Not wearing a mask" class is a must. The expected result from this evaluation is that the model is capable of classifying the class with high accuracy for the "Incorrect wearing" and the "Face mask-wearing" classes and has low accuracy on the "Not wearing a mask" class. The following diagram shows the images used for the model prediction accuracy.

TABLE V. ACCURACY OF THE DIFFERENT CLASSIFICATION MODELS OF THE CNN ARCHITECTURE

Model	Actual Prediction	
	Wears correctly	Wears Incorrectly
1	99.36	86.42
2	98.74	93.95
3	80.78	94.54
4	99.74	98.53
5	90.30	98.66
6	93.49	74.61
7	86.13	95.55
8	73.86	98.44

We can conclude from the accuracy prediction test that some models do not classify the images correctly. The hypothesis from the occurrence is that some models have overlearned in the training process hence the inability to classify images correctly. With that said, we can see a few models that pass the criteria on how the suitable model should work. Based on the further evaluation of the results, we can conclude that Model 5 will be suitable for the system. From the table below, the model achieves astronomical accuracy in the trained classes and low accuracy in the newly represented class. The next step for this chapter is to implement the model inside the system. To implement the model for real-time classification, the model must be implemented with the system. The system work by capturing each frame from the

video stream as input. From the input, the system preprocesses the input images to create a suitable specification for the model. After preprocessing, the input goes to the model to be predicted image class and the accuracy of the prediction. With that information, the system displays the result on the screen. The diagram below shows the code of the system. The system puts a threshold for the "face" class to determine if the image is in the "face" category. Figure 6 and figure 7 shows the model implementation result on the system. From the diagrams, we can conclude that the model successfully classifies images in the real environment. Even though the result for diagram 4.9 is not 90% accurate, the model manages to classify it correctly. Overall, the model is useable for the classification in real-time.

Fig. 6. Live detection of wearing mask improperly

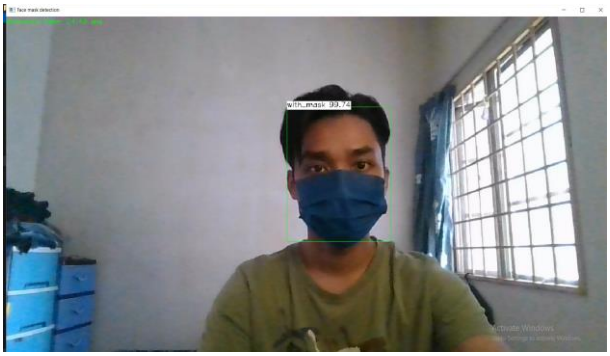

Fig. 7. Live detection of properly wearing a mask

V. SUMMARY

Further in the evaluation process, some models struggle to classify the image correctly. The hypothesis is that the model was overlearned during the training process, thus unable to make a correct classification. The hypothesis is back with the flawless result on confusion and classification matrix on the models. From the model prediction accuracy, we finally found a suitable model for the system. The model consists of high accuracy in the trained classes and low accuracy in the newly introduced class. Implementing the selected model on the system shows that the model can classify the input in real-time. Even though the system can work with real-time input, some flaws were present. Even though the selected CNN model for the system is the most suitable among created models, in terms of predicting and classifying from real-time stream input, the model does not have low accuracy in the "not wearing face mask" class. This problem made it difficult to determine the threshold for classifying the input for the "not wearing face mask" class since the system has different accuracy values for the "wearing face mask" class for different people. Another limitation of this study is that the system automatically shuts down when the camera captures high-speed objects on the screen. The final limitation in this study is that the system has difficulty classifying the human face from a far distance. This limitation made it difficult to implement the system for real-world use. several improvements can be made to improve the system performance for the future of this study. Further research might explore a better understanding of manipulating the video stream input that can help the model predict the input more accurately. Different size resolutions of the image dataset could improve the model's classification of human faces from far distances is highly recommended in the future investigation of the dataset. The final improvement is to allocate more time to perfect the system for any error that may occur.

VI. ACKNOWLEDGEMENTS

A special appreciation for Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA for supporting the publication of this paper

REFERENCES

[1] Sikaroudi, M. K., Zonooz, S. R., Ebrahimi, Z., Jebraili, H., Farsi, F., Talebi, A., & Masoodi, M. (2021). Assessment of anorexia and weight loss during the infection and recovery period of patients with coronavirus disease 2019 (COVID-19). Clin Nutr Open Sci, 40, pp. 102-110.

[2] Sharma, P., Gupta, S., Goel, N., Gupta, A., Saini, V., & Sharma, N. (2021). A review: novel coronavirus (COVID-19): an evidence-based approach. In Biomedical Engineering Tools for Management for Patients with COVID-19.

[3] Chollet, F. (2021). *Deep Learning with Python*. Manning Publications.

[4] Loey, M., Manogaran, G., Taha, M. H. N., & Khalifa, N. E. M. (2021). Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection. Sustain Cities Soc, 65, 102600.

[5] Mishra, N. K., Dutta, M., & Singh, S. K. (2021). Multiscale parallel deep CNN (mpdCNN) architecture for the real low-resolution face recognition for surveillance. Image and Vision Computing, 115.