Face Mask Detection Using MobileNetV2

Marita Brichan 40138194
Khalil Nijaoui 40092653

Github Link:
https://github.com/maritabrichan/Face-Mask-Detection-MobileNetV2/blob/master/Mask%20detectionNotebook.ipynb

Abstract - When the COVID-19 pandemic hit back in 2020, the entire world changed. Everyone went from living a stress-free life about viruses and infections to having to wear a mask everywhere they go in order to protect themselves and others from covid-19. Even though for the past 2 years, it has been required by the law to wear a mask in every closed space, many people choose to retaliate and not wear their masks indoors. This is why the need for a face mask detection and alert system to be implemented using image processing techniques. A deep convolutional neural network would perform classification in order to know if the image contains a mask and a human face and if that mask is worn properly. This will be performed by using MobileNetv2. Furthermore, in such a problem, we will be transferring learning using deep trained models from a very large dataset on Kaggle will be useful.

I. INTRODUCTION

The COVID-19 pandemic has created a lot of problems and pressure on the health system. It has been more than two years since the virus was first detected, and yet it is still contagious as ever. The only means to protect a person from catching this virus is to wear a mask, especially indoors. Wearing a mask properly will at the end save many lives and prevent this virus from spreading any further. Currently, in order to make sure that everyone wears their masks indoors, human employees are present to make sure that everyone is following the rules. In order to do that, the human mind first detects a human face, and then it will look for a mask on that face, and then ensure if that mask is covering both the mouth and the nose. If the person is not wearing a mask, or their mask is not worn properly, the human mind will know that and will alert those people and ask them to wear their masks properly.

However, this system is not perfect and has actually proven to sometimes be risky for the employees. Some people go as far as assaulting employees when asked to wear their masks. Not to mention, that this system is not perfect either, these employees' jobs often include many other things than making sure that everyone is wearing their masks properly, so people could sneak in and put everyone in that indoor place at risk of exposure to the covid virus. So, a better solution would be to use image segmentation in order to detect if people are wearing their masks. The implementation of this system will have many advantages, among which would be to spread awareness about the pandemic and force people to wear their masks properly in order to keep themselves and everyone around them healthy and

safe. Plus, stop any further spread of COVID-19. It is to make sure employees feel protected in their working places and make sure that the spread of the covid-19 virus decreases. This system could also be used in hospitals or research labs after this pandemic is over, in case there are any protected floors that require all staff and visitors to wear a mask for protection purposes.

Using deep learning frameworks in order to teach the computer what a person wearing a mask looks like and what a person not wearing their mask or wearing it improperly looks like. This will have to also be done from various different angles to make sure that the computer learns. Deep learning sometimes exceeds the performance at the human level. The deep learning networks used in this project will be Keras which is built on top of Tensorflow. Tensorflow is an end-to-end open-source platform for machine learning while Keras is a framework that can scale to large clusters of GPUs. The used convolutional neural network is MobileNetv2 which is 53 layers deep.

In order for the computer to identify a person who is properly wearing their mask versus a person who is not, the network will have to acquire a very large dataset in order to be trained. The very large set of labeled data that will be used was found on Kaggle.

Instead of going with a new model, a pre-trained model will be used which will allow for learning transfer. This means that the knowledge from one learned task will be transferred to a new task. In this case with the use of a convolutional neural network, learned features will be transferred quickly to this new project which will allow us to reduce the number of images needed to train the new network. This method has proven to be more time-efficient than training a model from scratch.

This project relies entirely on computer vision which is the computer being able to analyze images given to it and derive meaningful information from them. [1] This project will be using OpenCV.

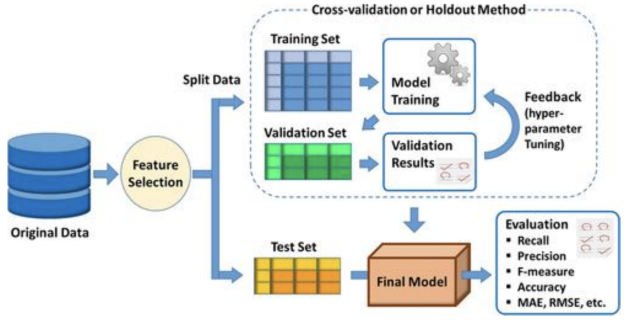

Figure 1: Image Classification Pipeline [8]

This paper will talk more in detail about how different image processing steps are used in order to create software that would detect if a person is wearing a mask properly or not.

II. RELATED WORK

With this pandemic affecting the entire healthcare system and changing everyone's life, many trained professionals started dedicating some of their work and studies towards methods to help reduce the spread of covid-19. Among these, many took an image processing approach where there was some face mask detection involved. There are various techniques, datasets and neural networks to get such a result. However, the existing methods are not efficient enough to be used in public spaces. Furthermore, the datasets for such a problem seem to be limited.

In the paper Single Camera Masked Face Identification, they talk about using a single camera to identify and detect a masked face using two approaches. First, by using a pre-trained

YOLO-face/trained YOLOv3 model. Second, by using a pre-trained one with RetinaFace in order to place the face and mask on it, and then VGGFace2 for mask verification. It seems to have good precision to be used in public spaces. [2]

In the paper Joint Face Detection and Alignment Using Focal Loss-based Multi-task Convolutional Neural Networks, they explore the use of MTCNN which is a combination of CNN which stands for Convolutional Neural Network and the traditional cascade structure which has proven to be effective. Furthermore, in order to improve face classification performance, it introduces Focal Loss. This method has been shown to be simple and effective and its performance was good on the public dataset. [3]

Furthermore, the research paper Facial Mask Detection using Semantic Segmentation uses face segmentation using Predefined Training Weights of VGG and a fully convolutional network to perform semantic segmentation on facial features. It is capable of identifying multiple face masks in one frame which makes it faster and able to identify if a group of people walking together are all wearing masks or not without missing anyone. It can also detect non-front faces. [4]

III. PROPOSED METHODOLOGY

In order to detect if a person is wearing a mask on their face properly, the model will first need to be trained using a dataset. In this case, the dataset was taken from Kaggle. Then preprocessing is performed so that the data can be cleaned and split into training and testing datasets. As the name indicates, the training dataset will train the model resulting in a fully trained model, while the testing dataset will test the model resulting in a tested model. After this, the model should be fully trained and tested in order to be able to detect if a person is wearing a mask or not on its own. This is when the validation dataset will be used to make sure that the system is accurately working.

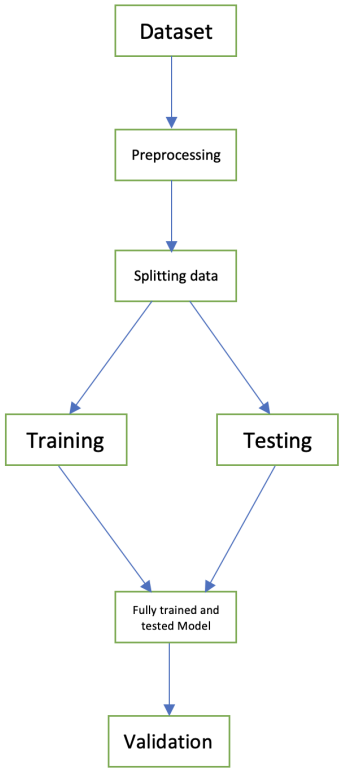

Figure 2: The proposed methodology

A. Dataset

The dataset used in this project was taken from Kaggle. [5] It contains 3 directories, Test, Train and Validation. Test contains 50 images with masks and 50 others without a mask. Train contains 600 images in total, 300 with masks and 300 without. As for Validation it contains 153 images with people with masks on and 153 images of people without masks on. In total, the dataset contains 1,006 images. A sample of some of the

images taken from the Kaggle dataset is in Figure 3.

Figure 3: Different images of people with and without masks [5]

B. Preprocessing

To make the most of our training set, we performed a number of preprocessing and data augmentation techniques.

When we load the images, we set the color mode to rgb, the images are resized to 224 width and Height. This is critical as it makes the learning of the machine faster for smaller images and many models require that the image size is the same across collected images.

The load_img step also performs an interpolation which is used to reassemble the image if the target size is different from the size of the loaded image. The selected algorithm is 'bicubic' which is a non-adaptive algorithm extending the cubic interpolation. It considers 16 pixels(4x4) compared to the bilinear interpolation which only takes 4 pixels (2×2) which might affect its speed.

Once the images are loaded and resized, we convert each PIL image to a Numpy array to use it to perform the next steps as we need to understand the features of the image.

Next, we use the preprocess_input method form mobilenet_v2 to adapt the image to the format that the model requires (The input pixel values are scaled between -1 and 1, sample-wise).

The processed images are now augmented to generate better performance and outcome of the machine learning model through the creation of more data for the training process.

Using the Keras ImageDataGenerator we augment the images by combining different techniques such as rotation, shifting, flipping, brightness change, shearing, and varying zoom.[6]

Keras ImageDataGenerator generates augmented images while the model is still in the training stage meaning the model will receive new variations at each epoch.

The data augmentation techniques used:

- **Rotation_range**: randomly rotate the images through degree range 60.
- **Random Shifts**: height_shift_range and width_shift_range to shift the image vertically and horizontally.
- **Random Flips**: horizontal_flip to randomly flip the image along the horizontal axis.
- **Random Brightness:** brightness_range to vary lighting conditions
- **Random Zoom**: zoom_range to randomly zoom in or out on the image
- **Random shear**: shear_range to randomly apply shearing transformations
- **Fill mode:** fill_mode the strategy used for filling in newly created pixels, which can appear after a rotation or a width/height shift.

C. Dependencies

This project uses Keras which is built on top of Tensorflow. Tensorflow is an end-to-end open source platform developed by Google for machine learning. It is the most popular machine learning framework today and it is being used by many big companies. Keras is a framework that can scale to large clusters of GPUs. It is also an industry strength framework. It has low-level flexibility for research implementation while also having high-level convenience features in order to make everything go faster. [7]

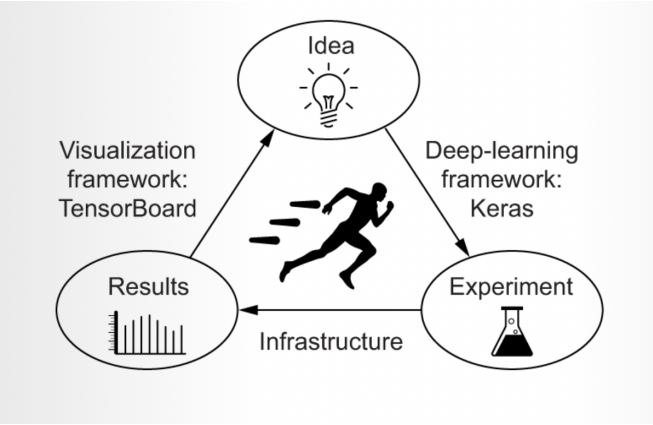

Figure 4: Keras framework [7]

D. Model Architecture

A convolutional neural network convolves learned features with input data. It uses 2D convolutional layers. A CNN usually has two main parts, the first being a convolution/pooling mechanism that breaks the image into features in order to analyze them; and second, a fully connected layer taking the output of the last part to predict the best label to describe the image.

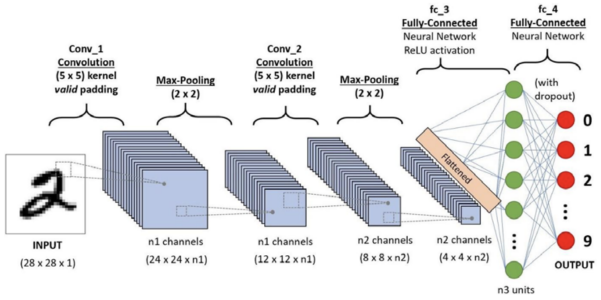

Figure 5: CNN Architecture [8]

The convolutional neural network has two main layers: feature learning layers and classification layers. The feature learning layer performs many operations in order to alter the images with the intent of learning the different features. While the classification layer is the one that classifies and puts a label on the image.

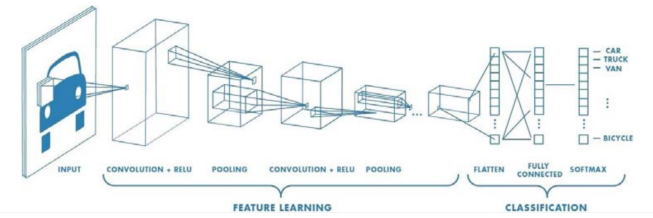

Figure 6: A convolutional Neural Network [8]

In this project, the convolutional neural network architecture of MobileNetV2 is used which seeks to perform well on mobile devices. Figure 7 shows the building block of the MobileNetV2 architecture.

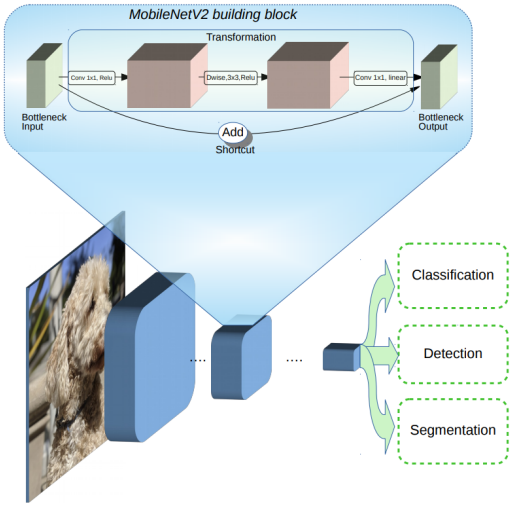

Figure 7: MobileNetV2 building block [8]

In order to construct the model, the following functions were used.

- **Average Pooling 2D** is an average pooling operation for spatial data in order to downsample the input's height and width by taking the size defined by pool_size which in our case is fixed at 7 for both height and width. So they will both be downscaled by a factor of 7.

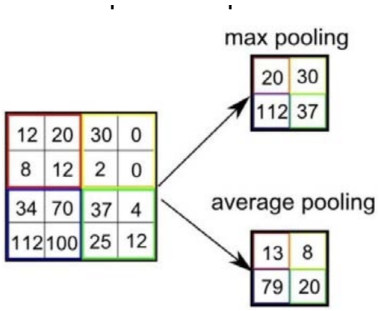

Figure 8: Pooling [8]

- **Flatten** flattens the input without affecting the batch size
- **Rectified Linear Unit** activation on the preceding output which outputs an array of shape (None, 128)
- **Dropout** is used in order to reduce overfitting, layers will be dropped at a rate of 0.5.

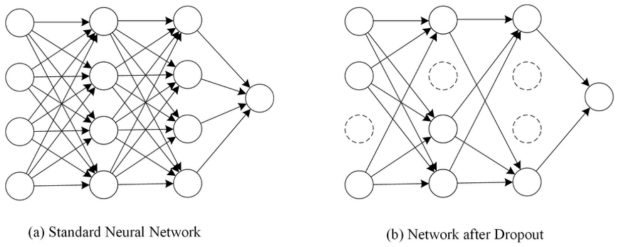

Figure 9: Dropout [8]

- **Softmax** activation on the previous output which outputs an array of shape (None, 2)

IV. EXPERIMENT AND RESULTS

1. **System environment:**

The model is trained locally on a system with the following

- **CPU**: Intel(R) Core (TM) i7-5500U CPU @ 2.40GHz
- **RAM**: 8.00 GB

2. **Hyperparameters:**

If omit the preprocessing and data augmentation steps the results may become misleading and the accuracy degrades a lot.

Although a 256x256 size image might generate better results as its higher resolution, the experiments showed that 224x224 images are enough to achieve great accuracy in reasonable time.

- **Initial learning rate:** 1e-4
- **Number of epochs to train for:** 50
- **batch size:** 32
- **image size:** 224x224

In the data augmentation step the following parameters:

rotation_range=60
zoom_range=[0.4,1.0]
width_shift_range=0.3
height_shift_range=0.3
shear_range=0.15
horizontal_flip

Note:
- **Vertical flip** make the model less accurate as in real life we wouldn't find people upside down.
- **Brightness range**: it makes the model less accurate.

3. Evaluation Criteria:

There are various ways to check the performance of our model:

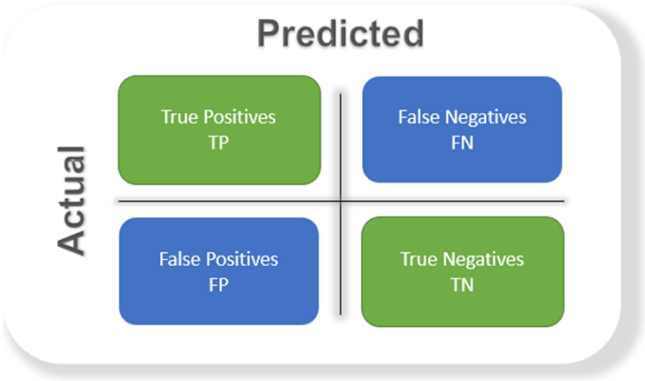

Figure 10: Confusion matrix[10]

- **Accuracy**

Accuracy is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions. [9]

$$\text{accuracy} = \frac{\text{correct predictions}}{\text{all predictions}}$$

- **Precision:**

Precision is defined as the fraction of relevant examples (true positives) among all the examples which were predicted to belong in a certain class.

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

- **Recall:**

Recall is defined as the fraction of examples which were predicted to belong to a class with respect to all of the examples that truly belong in the class.

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

- **The F-score:**

The F1 score is the harmonic mean of precision and recall. This takes the contribution of both, so higher the F1 score, the better. The more generic score applies additional weights, valuing one of precision or recall more than the other. [10]

$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{tp}}{\text{tp} + \frac{1}{2}(\text{fp} + \text{fn})}.$$

4. Results:

We used the early stopping method to stop the training once the model performance stops improving.

The first epoch: the accuracy: 0.7465 and at the end of the fist epoch the value of the accuracy: 0.9510

The model was early stopped after 11 epochs where the loss value didn't improve from 0.0897

-The time taken by the model: 23 minutes and 39.21 seconds

The training loss and accuracy plot according to the epochs:

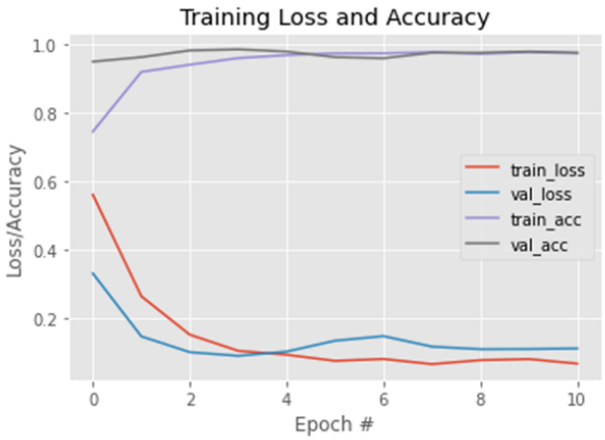

Figure 11: Training Loss and Accuracy

Our model achieved 98.69% **accuracy** which indicates high levels of performance making it accurate and trustworthy. However, not only accuracy is important as the other metrics indicate how well our model is performing. The following table summarizes the precision and recall and the f-score of our model:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	153
1	0.99	0.99	0.99	153
accuracy			0.99	306
macro avg	0.99	0.99	0.99	306
weighted avg	0.99	0.99	0.99	306

Figure 12: Precision and F-score of the model

The confusion matrix:

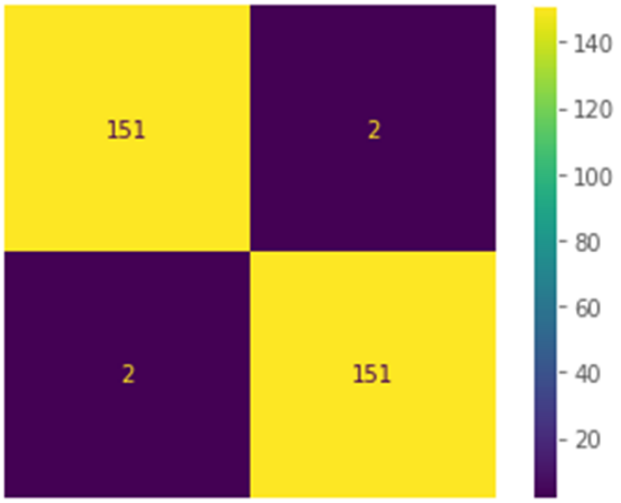

Figure 13: Confusion Matrix

Our test dataset had 306 images, 304 were predicted correctly 151 to have mask and 151 not to have a mask. Only 4 incorrect predictions were produced where 2 were predicted as wearing a mask while they were not, and 2 were wearing a mask and our model predicted that they aren't wearing one.

In our case, a false negative is almost as important as false positive as there is not much of a consequence as opposed to a cancer detection algorithm where a false negative might lead to missing the detection of cancer which might cause death.

V. CONCLUSION & FUTURE WORK

To conclude, using the convolutional neural network of MobileNetV2 with the dataset from Kaggle, and using Keras has proved to be working with 98.69% accuracy rate. From what has been shown, this solution is an accurate and trustworthy one with high levels of performance.

This approach is great because with such a high accuracy rate, we are one step closer to implementing this system on cameras indoors in order to make sure that everyone wears their masks properly and keep everyone around them including themselves healthy and safe from this pandemic. This implementation of this system will help reduce the stress of the health system and spread awareness by encouraging people to wear masks everywhere.

VI. REFERENCES

FOR THE RESEARCH PAPER

[1] "What is Computer Vision?," *IBM*. [Online]. Available: https://www.ibm.com/topics/computer-vision. [Accessed: 14-Apr-2022].

[2] V. Aswal, O. Tupe, S. Shaikh, and N. N. Charniya, "Single Camera Masked Face Identification," *IEEE Xplore*, 23-Feb-2021. [Online]. Available: https://ieeexplore.ieee.org/document/9356313. [Accessed: 14-Apr-2022].

[3] R. Wang, J. Tian, and C. Jin, "Joint face detection and alignment using focal loss-based multi-task convolutional neural networks," *SpringerLink*, 01-Jan-1970. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-31456-9_30. [Accessed: 15-Apr-2022].

[4] T. Meenpal, A. Balakrishnan, and A. Verma, "Facial mask detection using semantic segmentation," *IEEE Xplore*, 31-Oct-2019. [Online]. Available: https://ieeexplore.ieee.org/document/8888092. [Accessed: 15-Apr-2022].

[5] S. Kumar Mohanty, "Face mask detection & alert system MOBILENETV2," *Kaggle*, 30-Jun-2021. [Online]. Available: https://www.kaggle.com/code/sidharth178/face-mask-detection-alert-system-mobilenetv2/notebook. [Accessed: 03-Apr-2022].

[6] aniruddha, "Image augmentation keras: Keras imagedatagenerator," *Analytics Vidhya*, 11-Aug-2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatagenerator/. [Accessed: 12-Apr-2022].

[7] K. Team, "Simple. flexible. powerful.," *Keras*. [Online]. Available: https://keras.io/. [Accessed: 14-Apr-2022].

[8] A. Ben Hamza, "Image Processing Slides."

[9] J. Jordan, "Evaluating a machine learning model.," *Jeremy Jordan*, 21-Jul-2017. [Online]. Available: https://www.jeremyjordan.me/evaluating-a-machine-learning-model/. [Accessed: 16-Apr-2022].

[10] K. Nighania, "Various ways to evaluate a machine learning models performance," *Medium*, 30-Dec-2018. [Online]. Available: https://towardsdatascience.com/various-ways-to-evaluate-a-machine-learning-models-performance-230449055f15. [Accessed: 15-Apr-2022].

FOR THE CODE

The code used in the notebook was taken from S. Kumar Mohanty on Kaggle and improved

[5] S. Kumar Mohanty, “Face mask detection & alert system MOBILENETV2,” *Kaggle*, 30-Jun-2021. [Online]. Available: https://www.kaggle.com/code/sidharth178/face-mask-detection-alert-system-mobilenetv2/notebook. [Accessed: 03-Apr-2022].

[11]“Tf.keras.preprocessing.image.imagedatagenerator,” *TensorFlow*. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator. [Accessed: 05-Apr-2022].

[12] J. Brownlee, “How to configure image data augmentation in Keras,” *Machine Learning Mastery*, 12-Apr-2019. [Online]. Available: https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/. [Accessed: 05-Apr-2022].

[13] K. Team, “Keras Documentation: Image data preprocessing,” *Keras*. [Online]. Available: https://keras.io/api/preprocessing/image/. [Accessed: 05-Apr-2022].