Faculty Name**: Computer Science.**

Course Name**: Selected Topics Cs-2.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **ID** | **Full Name**  **[In Arabic]** | **Attendance**  **[Handwritten Signature]** | **Final Grade** |
| **1** | 202000289 | خلود محمد محمد جمال |  |  |
| **2** | 202001047 | يارا السيد محمد علي |  |  |
| **3** | 202000013 | احمد أشرف طه إبراهيم |  |  |
| **4** | 202000283 | خالد عباس عبد الحميد |  |  |
| **5** | 202000876 | مريم حسين حلمي رضوان |  |  |
| **6** | 202001041 | هنيده محمد احمد الراوي |  |  |

**Paper Details:**

* Authors name:
* **Rajaram Yadav**
* **Safal Gautam**
* **Rahul Ratna Das**
* Paper name**: Covid Face Mask Detection Using Neural Networks.**

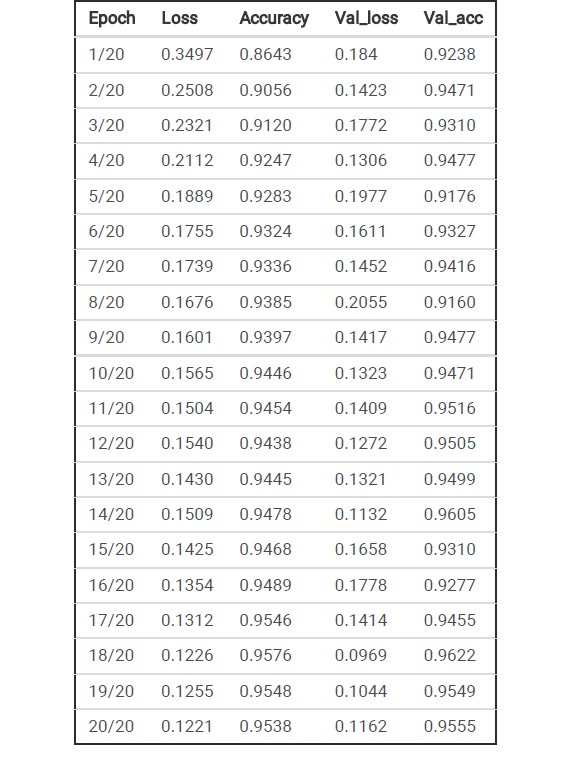
publisher name**: research square.**

* Date of publication**: September 6th, 2022.**
* The dataset used **:**

[**https://www.kaggle.com/datasets/vijaykumar1799/face-mask-detection**](https://www.kaggle.com/datasets/vijaykumar1799/face-mask-detection)

* **The implemented algorithm: MobileNetv2**
* **The result:**

**The result for 20 iterations in checking the loss, accuracy, value loss, value accuracy when training the model is shown in Table.**

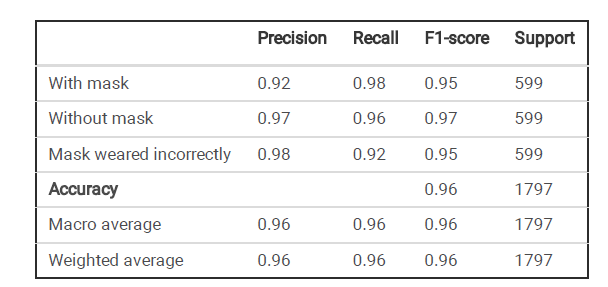
.

From Table 1, we can interpret that the accuracy is increasing at the start of the second epoch, and loss is

seen decreasing after it. At the 20th epoch the loss reduced to 0.1221. Similarly, accuracy reached 0.95.

The process of training into the Deep neural network was much faster than expected. The table is

plotted in the graph shown for better understanding in Fig. 2.



Our proposed model was able to classify the mask-wearing image with faces. The accuracy provided by

the model in such a scenario was near about perfect as the model detected all the faces that had masks

on them accurately along with the accuracy level and it also accurately detected the faces that weren't

wearing masks. Though the model doesn’t give an impression about people with an incorrectly worn

mask, the results at the current level are satisfactory.

**Project Description:**

* **The name of the dataset used: Face Mask detection dataset.**
* **The link of dataset :**

[**https://www.kaggle.com/datasets/omkargurav/face-mask-dataset**](https://www.kaggle.com/datasets/omkargurav/face-mask-dataset)

* **The total number of samples in the dataset: 7553.**
* **the dimension of images: (128,128,3).**
* **Number of classes and their labels: 2 Classes( With Mask , Without Mask).**

**Implementation details:**

* Image pre-processing

First step here begins with the iteration through all the images and storing the labels

of each image (without mask, with mask and wearing mask incorrectly). The image sizes are nonuniform.

We need some standard size so that we can feed it into our neural network. So, it is resized to

224 x 224 x 3 image which is our standard image size. Then the image is converted to a NumPy array.

CV2 is better with BGR channel images, and we are using it in a later phase, so we convert our image into

BGR channel. After the image is preprocessed, it is ready to feed it into our model.

* Splitting data

The data was split into two collections, which are training data named 80 percent, and the rest of the part

(20 percent) is testing data. Each collection consists of all the with-mask, without- mask and wearing

mask incorrectly images.

* Training the model

Steps:

a. Read the dataset and store the image-paths.

b. Iterate through all the images and

1. Store the labels of each image (without mask/ with mask).

2. Convert image to array and preprocess the image.

3. Append the image to data.

c. Convert data and labels to a NumPy array.

d. Transform multiclass levels to binary levels using labelBinarizer.

e. Split the training and testing data. (20% data is used in testing and 80% data is used in training).

f. Create instantiation of mobilenetv2 model and remove the last layer.

g. Add Average pooling 2D, Flatten, Dense (with relu activation), Dropout and Dense (with SoftMax

activation) layers.

h. Assign trainable properties of mobile net v2 base model as false.

i. Declare learning rate (0.001), epochs(20) and batch size(12).

j. Use Adam as optimizer and binary cross entropy as loss.

k. Fit the model giving following parameters:

1. Use image augmentation flow to artificially expand the size of the training dataset. (Use rotation range,

zoom range, width\_shift\_range, etc. as parameters)

2. Steps per epoch: Calculated as length of training data divided by batch size.

3. Validation data: test data.

4. Validation steps: Calculated as length of testing data divided by batch size.

5. Epochs

l. Save the model

m. Test the model with the test splitted data and create classification reports, plots as needed.

* Testing the model

Steps:

a. Read photo path and weights path for Open CVV DNN and create a model using the same parameters.

b. Load the saved trained model.

c. Read the image to be tested.

d. Use blob from image to preprocess image and convert image to the format that has been used by DNN

while training the model.

e. Detect the faces from the model and save the detections.

f. Iterate over the detections and do the following steps:

1. Find the confidence (confidence of detection of faces).

2. Store the start and end coordinates (both x and y coordinates).

3. Convert the image to RGB channel and resize, convert to array, and preprocess the image.

4. Predict the probability of with and without mask using the model loaded initially (our trained model).

5. Draw a rectangle surrounding the face and put text as “Mask” in green color if mask probability is

greater than without mask probability and put text as “Without mask” in red color if without mask

probability is greater than mask probability.

g. For mask detection on video stream, start video stream and Repeat step 5 and 6 for each frame.

* Testing the model

To implement our model in the real world, live video is captured, frame by frame. This video feed is then

fed to our algorithm (model). It can proceed only if a face has been detected. After detecting a face,

required pre-processing tasks are done. Then, the model converts the pre-processed into an array-form

and does the required further processing tasks using MobileNetv2.

Then, for the given frame, it classifies if the subject is not wearing a mask or wearing it incorrectly or

wearing it properly as the model requires them to. The classification result is shown by putting a colored

rectangle around the subject’s face with the probability of that classification in the same color. It uses the

following coloring scheme:

a. No mask: Red

b. Mask weared incorrectly: Blue

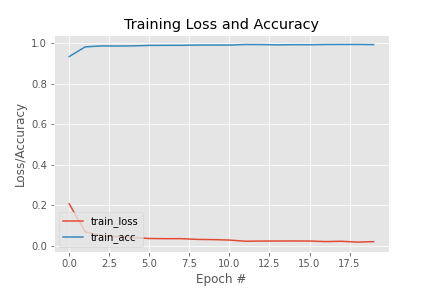
c. Mask: Green

This process is repeated for each video frame. Sample images classified by the model are shown below:

* The ratio used for:
* **Training: 70%**
* **Testing: 30%**
* Number of images in each:
* **3725 Images of Face with Mask.**
* **3828 Images of Face without Mask.**
* The main steps :
* a. Read the dataset and store the image-paths.
* b. Iterate through all the images and
* 1. Store the labels of each image (without mask/ with mask).
* 2. Convert image to array and preprocess the image.
* 3. Append the image to data.
* c. Convert data and labels to a NumPy array.
* d. Transform multiclass levels to binary levels using labelBinarizer.
* e. Split the training and testing data. (20% data is used in testing and 80% data is used in training).
* f. Create instantiation of mobilenetv2 model and remove the last layer.
* g. Add Average pooling 2D, Flatten, Dense (with relu activation), Dropout and Dense (with SoftMax
* activation) layers.
* h. Assign trainable properties of mobile net v2 base model as false.
* i. Declare learning rate (0.001), epochs(20) and batch size(12).
* j. Use Adam as optimizer and binary cross entropy as loss.
* k. Fit the model giving following parameters:
* 1. Use image augmentation flow to artificially expand the size of the training dataset. (Use rotation range,
* zoom range, width\_shift\_range, etc. as parameters)
* 2. Steps per epoch: Calculated as length of training data divided by batch size.
* 3. Validation data: test data.
* 4. Validation steps: Calculated as length of testing data divided by batch size.
* 5. Epochs
* l. Save the model
* m. Test the model with the test splitted data and create classification reports, plots as needed.
* Read photo path and weights path for Open CVV DNN and create a model using the same parameters.
* Load the saved trained model.
* Read the image to be tested.
* Use blob from image to preprocess image and convert image to the format that has been used by DNN
* while training the model.
* Detect the faces from the model and save the detections.
* Page 7/14
* Iterate over the detections and do the following steps:
  + Find the confidence (confidence of detection of faces).
  + Store the start and end coordinates (both x and y coordinates).
  + Convert the image to RGB channel and resize, convert to array, and preprocess the image.
  + Predict the probability of with and without mask using the model loaded initially (our trained model).
  + Draw a rectangle surrounding the face and put text as “Mask” in green color if mask probability is
* greater than without mask probability and put text as “Without mask” in red color if without mask
* probability is greater than mask probability.
* The hyperparameters:

**Depth, Dropout rate, Learning rate, Batch size.**

* The Result:



precision recall f1-score support

with mask 0.99 0.99 0.99 745

without mask 0.99 0.99 0.99 766

accuracy 0.99 1511

macro avg 0.99 0.99 0.99 1511

weighted avg 0.99 0.99 0.99 1511

**Accuracy : 0.9922**

**Loss: 0.0213**