# FACE MASK RECOGNITION

Shreshtha Mankala[1], Aastha Patel[1], Manasa Ivaturi[1]
[1] Indiana University, Luddy School Of Informatics, Data Science

## ABSTRACT

During the COVID-19 pandemic, a lot of restrictions and preventive measures have been put in place, one of which is wearing a face mask. The Project "Face Mask Recognition" aims at determining if the person is wearing a mask or not, and if he/she is wearing then is it the correct way or not. So, there still exists a necessity to wear a face mask in public areas mainly in medical and business setups. Although the pandemic has reduced to becoming endemic, there could be other similar diseases in future, so this kind of model can be used. The model is built using Convolution Neural Networks and performs well on images and real-time videos. For a real-world application, the model has been integrated with voice instructions.

Index Terms - Mask Prediction, Convolution Neural Networks, Voice Recommendation

## 1. INTRODUCTION

The world faced a new challenge in the year 2020, which had a high mortality rate. There are 508,041,253 cases and 6,224,220 deaths to the date according to WHO due to the COVID-19 virus. COVID-19 is a highly contagious disease that causes fever, dry cough, exhaustion, and breathing issues in the majority of cases. Diarrhoea, conjunctivitis, loss of taste or smell, skin rash, or colouring of fingers or toes are some of the minor symptoms. The technology and medical sciences have improved the care of this disease but that hasn't stopped the spread of the virus. The second wave of COVID-19, which began in early 2021, had a negative impact in India, making it difficult to discover infected people from the outside, as most people are asymptomatic or very mildly unwell [1].

The only solution is to take measurements so that the virus can't affect others or can't also enter the body. Most of the time we touch anything, especially in public locations, and we need to apply sanitiser to prevent infection from our fingers. While in a public area, the face mask is highly effective for preventing contamination through the air. There would be less chance to get ill if the mask was wearied properly [1].

Many countries have laws requiring people to wear face masks in public. These rules and laws were created in response to the rapid increase in cases and deaths in many areas. In public spaces, however, monitoring large groups of people is becoming more difficult. As a result, we'll develop a face mask recognition model which detects the faces and predict whether the mask is worn properly, incorrect or without a mask [2].

Here, our model "Face Mask Recognition" is built using CNN. The model can be implemented in public areas with real-time data as our model works with real-time captured video with the Open CV technique.

## 2. RELATED WORK

COVID-19 is a global pandemic that has spread across all disciplines. Nonetheless, this paved the way for researchers. We have seen many research implementations for recognizing if a person is wearing mask or not. Because there are certain inaccuracies in the findings of early laboratory tests, as well as their delays, various options were explored by the researchers.[3]

The approach in [4] makes use of tensorflow, keras, deep learning and MobileNetV2 models. The model was broken down into two phases namely training and deployment. The training phase focuses on loading the dataset, training, serializing, and saving the model. Deployment phase includes loading the model and classifying an image as with_mask or without_mask. The dataset for this model was built from data augmentation by adding the face mask to normal faces(which don't have face masks). Different tasks included during model building were data augmentation, loading the MobilNetV2 classifier which also included fine tuning the model with pre-trained ImageNet weights, building a fully connected layer and backpropagation. Data Augmentation was done using random rotation, zoom, contrast , hue, shift, and flip parameters so as to get better generalization. 99% accuracy was obtained on test set for real time video.

Another approach which was used in [3] is made up of two main parts, initial phase which consists of the training and testing models and next phase where the whole framework is tested. The initial phase contains data in three blocks training, validation and testing. After performing forward propagation, process of calculating errors and updating the network's parameters is performed which is called backward propagation. Different hyperparameters used are learning rate, batch-size, epochs, optimizer, loss function.

The additional implementation in this paper is for maintaining social distancing using the anchor boxes.

Unlike the above two methodologies this paper has proposed to perform pre-processing of the dataset and the output results were directly being sent to the deep learning model. They have taken up some pictures of popular celebrities and manually added face mask to them. The second dataset which was taken from Kaggle is a fine-tuning dataset which works better than the first one. The third dataset used is a detailed dataset that adds fresh entries into the pre-processing and increase the efficiency by adding mode data. The total dataset now has 20,721. In the preprocessing phase they have identified 67 cut points of a face using Euclidian geometrical distance from the nose point to the upper, left, right, lower boundaries of the face, which are then used by the face cut algorithm to only obtain the area below the eyes and till the chin area. Data augmentation like zooming, shearing, cropping is performed further. ResNet50 model is used with 48 layers, one max-pool, and one average pool layer. A dense layer is added at last stage to detect the flatten feature vector from the points. A softmax layer is also added in the last layer so that it can be used to remove to get feature embeddings to plot the visualization. The paper shows the result via Test Accuracy, Average Class-specific accuracy(ACSA), Predicted Positive Value(PPV), loss value, classification report.[1]

The work presented in the above papers uses Keras and TensorFlow modules whereas we have tried implementing the model using pyTorch. Our model also consists of training on image dataset and testing on real time videos. We have got more accuracy than all the models discussed above that is 98% accuracy. Also, our model differentiates by giving 3 labels instead of 2. The labels are 'with mask', 'without mask' and 'wearing mask incorrectly'.

## 3. METHODOLOGY

3.1 Overview and Approach

The idea of this section is that it provides a holistic understanding of the entire process in a brief. It outlines the pipeline of steps taken all the way from importing the dataset to finalizing the results to make informed conclusions and decisions. The steps that include –

CNN's and Architecture
Optimizers
- SGD
- ADAM
- RMSprop

Evaluation metrics and understanding results.
Analyzing, visualizing and summarizing.

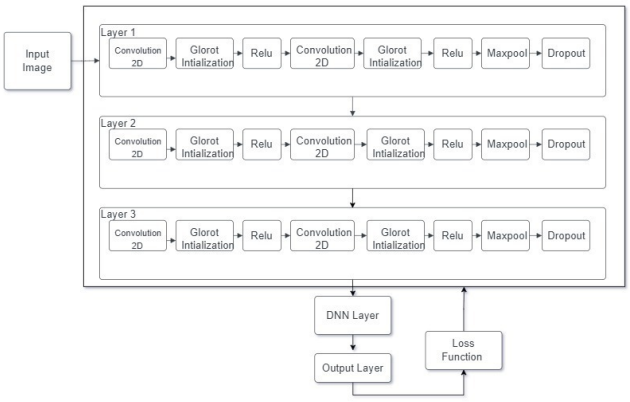
3.2 CNN's and Architecture

CNNs are totally automated image feature extractors that are. A CNN successfully employs neighbouring pixel information to efficiently downsample the image, first through convolution and then through the use of a prediction layer at the end.

Using CNN's for images is tried and tested, especially given the complexity (in other words, the Convolution) involved with data represented as images.

The architecture of this setup is that CNN is made up of 3 layers, with each layer following a sequence of steps as elucidated in the below image.
On a high level, the CNN Architecture we intend to use, consists of the following layers:

- Convolutional Layers – 3
- Dense Layers – 1
- Fully connected layer – 1




Each convolution layer is made up of a conv2d layer, Glorot initialization, and a layer of Rectified Linear Unit (Relu) activation. Also, the previous parts of layers are repeated along with additions to Maxpooling and dropout that happens towards the end of the operations at this layer.

The output of the final layer is connected to a dense layer which is then flattened or linearized – which is the output of the CNN. Images are then passed through this entire network in batches – and this happens for a certain number of epochs. This with time improves the model and enables it to make more accurate predictions depending on further fine tuning and learning with respect to loss occurred at each batch.

3.3 Optimizers & Loss Functions

The very core idea of optimizers is that each optimizer works in a way that it enables the models to learn, predict and perform better. This is done so by minimizing the

loss/error functions. While there are a great number of optimizers that could be tried, considering the familiarity, scope and usability of the models, the optimizers used for this project are – Stochastic Gradient Descent, RMS-Prop (Root Mean Square Propagation) and Adam Optimization Algorithm.

Also optimizers perform as good as the loss function is. In this regard we are counting on one of the most used or prominent loss functions, Cross Entropy Loss.

3.4 Evaluation metrics

As the agenda of this project is to classify images into categories, the evaluation metrics must be able to give a good understanding of the model and its ability to predict.

Hence, the primary metric for the evaluation of the model under consideration is the "accuracy", alongside the loss.

Accuracy here would loosely translate into a model being able to classify a given image into mask worn, not worn or otherwise worn incorrectly.

3.5 Analyzing, visualizing, and summarizing.

For understanding the optimizers, we need a way to empirically measure the performance of the models and their ability to judge an image and classify it accordingly. However, given that the dataset is too huge, it would be impossible to continue without an easier and visual representation of the dataset, its evaluation metrics and performance.

For the same, we are plotting the graphs between the evaluation metrics and epochs.

## 4 Experiment Setup

4.1 Initial setup

When discussing about the Initial setup of this project, it must be understood that after many iterations of alterations to every detail of the code and architecture, the model has been designed to be accurate yet not too specific so as to attain the objective of this project.

During the initial proposal for this project, it was ideated to leverage two dense layers and then continue only with one optimizer that being the ADAM optimizer. However, if we are to continue with only one optimizer, there would be little to no scope for cross validation, comparison and contrast.

Keeping that in mind, the initial setup has been modified to something as discussed in the section 3.2 of this document, where the convolutions are more detailed, more elaborate and well thought. The optimizers are considered with more caution. More detail has been given to loss functions and evaluation metrics.

4.2 Data preprocessing

Dataset that which was taken from the Kaggle, is a collection of images, taken from various sources. Because these images are borrowed from innumerous sources, it is not given that these images follow some if not any standardization, that is in the resolution of images, image sizes, pixel intensities etc., amongst other differences in the dataset.

If the model(s) is/are trained with such unregularized data, the model might not yield desirable results upon attempting to predict for validation/test/real-time data. Hence it is important that the dataset undergoes a process of regularization wherein each image is brought into some standard with uniformity, conformity and regularity.

For this purpose, the first step in pipeline of operations is that there is are transformations to image sizes and normalization of pixel intensities is performed. Thus, making the dataset ready for the next steps, that is for fitting dataset to the models, predicting, visualizing and analyzing the results.

Also, on that note, we intend to perform data augmentation on the images and make them more friendly for the future steps in the ML model.

After experimentation with different sizes and normalization values, it has been found that processing the images at the size of 64*64 are not very resource intensive, are still clearer and helpful for the model to predict.

More specifically, all images have been scaled down to 64x64 pixels. The below image elucidates the exact operation performed in the code.

```
# resizing/rescaling the image to 64*64 alongside normalizing the image
transform = transforms.Compose([transforms.Resize((64, 64)),
                                transforms.ToTensor(),
                                transforms.Normalize((0.5, 0.5, 0.5),
                                                     (0.5, 0.5, 0.5))])
```

4.3 Parameter Fine-tuning, epochs and learning rates

Parameter tuning is critical for any model that is designed to work even predictable misses and inaccuracies.

As discussed in the section 3.2 of this document, it was observed that the CNN/Model is more accurate when there are at least 3 layers, as each layer is convolving twice over the given datapoint. This alongside a kernel size of "3". Similar to the experimentation done in this case, it was also

found that for different optimizers would need different hyper parameters given that the learning is different with each optimizer. However, when the experimentation was done with a higher learning rate, although it appeared that the model is learning faster, in each case the model's accuracy only declined with each epoch – clearly indicating at an overfitted model.

Another important parameter that made a significant impact to the models and their performance is the number of epochs. The number of epochs for each optimizer with a different combination of learning rate was tried. However, given the inaccuracies in prediction data alongside the resource utilization and time constraints have been important factors to consider a uniformity as required across the project scope.

The first set of trails included a learning rate of 0.0001 and 3 number of epochs. The results of this experiment yielded the results as below -

```
# How to predict
prediction_root = os.path.join(test_data_path, "for_prediction")
CATEGORIES = [ "mask.png","28.png", "without.png"]
results = []
for category in CATEGORIES:
    print(category)
    image_path = os.path.join(prediction_root, category)
    results.append( prediction(image_path,transformer))

results
```

```
mask.png
tensor([[-0.1183,  0.6183, -0.2760]], grad_fn=<AddmmBackward0>)
28.png
tensor([[-0.2514,  0.6142, -0.1523]], grad_fn=<AddmmBackward0>)
without.png
tensor([[-0.2231,  0.9798, -0.4265]], grad_fn=<AddmmBackward0>)
[W NNPACK.cpp:51] Could not initialize NNPACK! Reason: Unsupported hardware.
['with_mask', 'with_mask', 'with_mask']
```

As it can be observed from the above image, the classification results are very inaccurate and it can be easily inferred from the same, that the number of epochs alongside the given learning rate are not effective enough to call this as our final model.

The same can be observed from the below image, where loss for training data is huge.

```
num_of_epoch = 3
optimizer_func = torch.optim.Adam
lr = 0.0001
```

```
saved_data, training_results = fit(num_of_epoch, lr, model, train_dl, val_dl, optimizer_func)
```

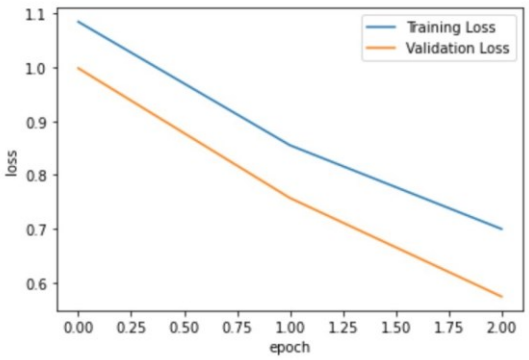
```
Epoch [1/3], Step [10/60], Loss: 1.1022
Epoch [1/3], Step [20/60], Loss: 1.1002
Epoch [1/3], Step [30/60], Loss: 1.1047
Epoch [1/3], Step [40/60], Loss: 1.0860
Epoch [1/3], Step [50/60], Loss: 1.0702
Epoch [1/3], Step [60/60], Loss: 1.0204
Epoch [0], train_loss: 1.0845, val_loss: 0.9982, val_acc: 0.5690
___ 0 ___
Epoch [2/3], Step [10/60], Loss: 0.9827
Epoch [2/3], Step [20/60], Loss: 0.8665
Epoch [2/3], Step [30/60], Loss: 0.8054
Epoch [2/3], Step [40/60], Loss: 0.8561
Epoch [2/3], Step [50/60], Loss: 0.8589
Epoch [2/3], Step [60/60], Loss: 0.7798
Epoch [1], train_loss: 0.8550, val_loss: 0.7569, val_acc: 0.6890
___ 1 ___
Epoch [3/3], Step [10/60], Loss: 0.8185
Epoch [3/3], Step [20/60], Loss: 0.7145
Epoch [3/3], Step [30/60], Loss: 0.6175
Epoch [3/3], Step [40/60], Loss: 0.5890
Epoch [3/3], Step [50/60], Loss: 0.6292
Epoch [3/3], Step [60/60], Loss: 0.6600
Epoch [2], train_loss: 0.6993, val_loss: 0.5737, val_acc: 0.7623
___ 2 ___
```

Considering the same, after taking up a different set of final model is built with 15 number of epochs and a learning rate of 0.0001 which we have found to be optimal, relevant and rational in this context in an all-encompassing manner.

## 5 Results

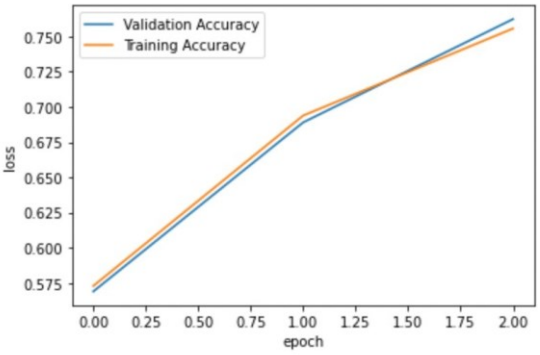
As discussed in the section 3.5 of this paper, visual representation of the results has been done in a graphical manner considering the volume of the data. When experimented with ADAM optimizers with the parameters are mentioned towards the end of section 4.3, the training and validation loss varied as below with the epochs.

```
<matplotlib.legend.Legend at 0x7fa3c94bb1f0>
```




In a similar manner, the training and validation accuracies are as mentioned here –

```
<matplotlib.legend.Legend at 0x7fa388127430>
```

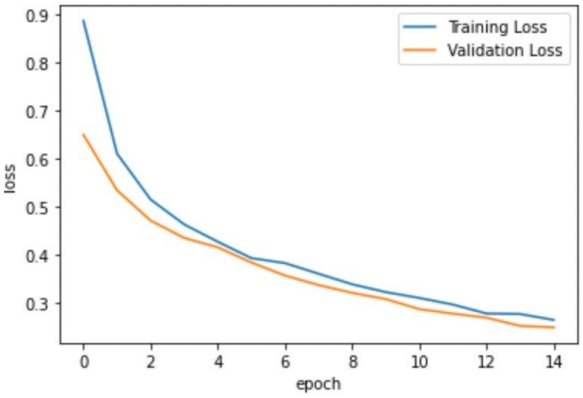



As we can infer from the graphs here, the validation accuracies with the ADAM optimizers are very close to the training accuracies, however, when this model was tested with real data (not very similar to training data), the models were able to predict accurately – indicating a great degree of generalization. The results are as shown below –

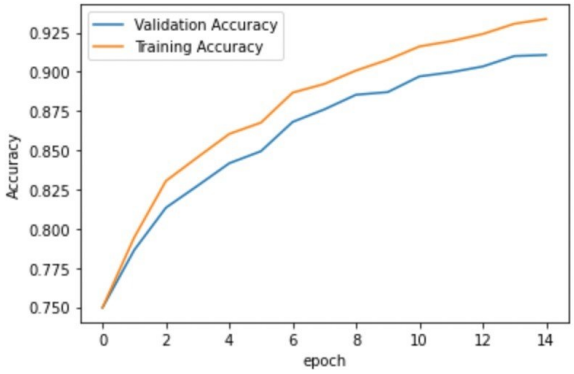
```
# How to predict
prediction_root = os.path.join(test_data_path, "for_prediction")
CATEGORIES = [ "mask.png","mask_weared_incorrect.png", "without.png"]
results = []
print("With Adam Optimiser")
for category in CATEGORIES:
    print(category)
    image_path = os.path.join(prediction_root, category)
    results.append( prediction(image_path,transformer))

results
```

```
With Adam Optimiser
mask.png
tensor([[-1.1815,  3.5160, -3.4236]], grad_fn=<AddmmBackward0>)
mask_weared_incorrect.png
tensor([[ 2.6896,  0.3059, -3.5039]], grad_fn=<AddmmBackward0>)
without.png
tensor([[-1.8678,  0.6074,  0.8043]], grad_fn=<AddmmBackward0>)
['with_mask', 'mask_weared_incorrect', 'without_mask']
```

When experimented with RMSProp optimizer with the parameters are mentioned towards the end of section 4.3, the training and validation loss varied as below with the epochs.




In a similar manner, the training and validation accuracies are as mentioned here –




In contrast to the ADAM optimizer, RMSProp optimizer is more generalizing and this is evident with a diverging graph with the accuracies and loss. This represents an even more generalized model which can be seen in the results during testing as below –

```
# How to predict
prediction_root = os.path.join(test_data_path, "for_prediction")
CATEGORIES = [ "mask.png","mask_weared_incorrect.png", "without.png"]
results = []
print("With RMSprop Optimiser")
for category in CATEGORIES:
    print(category)
    image_path = os.path.join(prediction_root, category)
    results.append( prediction(image_path,transformer))

results
```

```
With RMSprop Optimiser
mask.png
tensor([[-1.1815,  3.5160, -3.4236]], grad_fn=<AddmmBackward0>)
mask_weared_incorrect.png
tensor([[ 2.6896,  0.3059, -3.5039]], grad_fn=<AddmmBackward0>)
without.png
tensor([[-1.8678,  0.6074,  0.8043]], grad_fn=<AddmmBackward0>)
['with_mask', 'mask_weared_incorrect', 'without_mask']
```

We also tested the model on a video clip which is then captured in approximately 300 frames. We tried the model on the frames for which we got good performance. The classification for which is then converted into the form of audio output.

Below image shows the same execution –

```
In [24]: # How to predict

         image1 = "C:\\Users\\hari\\video_images\\frame150.jpg"
         result = prediction(image1,transformer)

         result
Out[24]: 'without_mask'
```

## 6. CONCLUSION

This paper presents a methodology to detect if a person is wearing a face mask or not. It is necessary to still wear face masks in health setups and very crowded areas. Even though the spread of covid has reduced but there is a need to address such issues if a similar disease emerges in the future. It could also help prevent spread of seasonal flu. The motivation for the work stems from people breaking the rules that are in place to prevent the spread of disease. A Convolution Neural Network is used to detect the mask on the face in the system's face mask detection architecture. Labeled image data were used to train the model, with the images being facial images with masks, without masks, and incorrectly wearing masks. The proposed system detects a face mask with 98% accuracy. The model was further tested for real time dataset captured from a video and the audio system was also implemented to restrict people and it eases the burden of manual labor to inform people about wearing masks. This kind of system can be implemented in video surveillance.

To make the model better we can make use of Bag-of-features model in the fully connected layer instead of classic CNN to get better representation. The images can be misleading as a person could cover their mouth with hand and it might be misinterpreted as mask covering. So to solve this kind of a problem the model should be trained with large number of samples with representative images of that kind. These kinds of images were not available

## 7. REFERENCES


[1] Arkaprabha Basu, Md Firoj Ali, “COVID-19 Face Mask Recognition with Advanced Face Cut Algorithm for Human Safety Measures”, IEEE, Kharagpur, India, November 2021.

[2] Mr.Kalla.Kiran, Bokka Vamsi Kiran, Devarapalli Cheswanth Sai, Gaggala Vijay Vamsi, Pitta Rani Salomi, “Face Mask Detection Using Machine Learning”, SSRN, India, September 2021.

[3] Safa Teboulbi, Seifeddine Messaoud, Mohamed Ali Hajjaji, Abdellatif Mtibaa, “Real-Time Implementation of AI-Based Face Mask Detection and Social Distancing Measuring System for COVID-19 Prevention” , Hindawi, V olume 2021, Article ID 8340779, September 2021.

[4] Rajat Sachdeva, Sonam, Harshita Sharma, “Face Mask Detection System”, International Journal of Scientific and Engineering Research, IJSER, India, December 2020