**DATA 255 DEEP LEARNING**

**SPRING 2021**



**Deep Learning Model for Face Mask Detection**

Submitted To: Prof. Shih Yu Chang

Submission Date: 05/13/2021

Submitted By:

Group 4

Samiksha Pandey

HimaBindu Nayani

Varunika Raja

Venkata Sai Kusuma Sindhoora Vankayala Siva

**Objective**

The new Coronavirus outbreak, also known as COVID-19, first appeared in December 2019. It originated in Wuhan, China, and by the beginning of March 2020, the virus had spread rapidly across the world, with the number of confirmed cases and deaths as a result of the virus had increased a lot. As a result of the pandemic, people all over the world are facing challenging times.

According to the World Health Organization (WHO), there have been 150,110,310 confirmed cases and 3,158,792 deaths reported globally as of 2:41 pm CEST, 30th April 2021 [1]. This number is increasing rapidly. When an infectious person sneezes or communicates with another person, water droplets from their nose or mouth disperse into the air and infect other individuals nearby. The WHO has identified fever, dry cough, tiredness, diarrhea, loss of taste, and smell as the main symptoms of coronavirus.

The healthcare system is going through a crisis. To fight the coronavirus, several precautionary steps have been taken. Cleaning hands, keeping a proper distance, wearing a mask, and not rubbing eyes, nose, or mouth are the most important, with wearing a mask. Wearing a mask is one of the many precautionary steps taken to prevent the transmission of this disease.

COVID-19 is a virus that spreads from person to person that can be prevented from wearing a face mask properly. COVID-19 can be kept under control if people keep a strict social distance and wear a face mask. Face mask detection is a method for determining whether or not someone is wearing a mask. Detecting an object from a scene is similar.

**Motivation**

With today’s advancements in the fields of science, technology, and medicine, a pandemic, that too on a global level, should not seem like a daunting or unmanageable threat. Yet, the entire world was devastated when it was hit by the first pandemic it has seen in decades starting in January 2020 when a mysterious death caused by pneumonia related to the next biggest and most deadliest viruses was reported in Wuhan, China. COVID-19, or Coronavirus Disease- 19, as it has since been termed, immediately spread throughout China and other parts of the world like wildfire. It made its way overseas and on January 21, 2020, the United States reported its first Covid case as well, just 12 days within its inception.

While our advancements in STEM allowed us to develop working vaccines within 1 year of lockdown that are still actively being distributed to the public, the world was not at all prepared for this pandemic. While travel bands, lockdowns, and stay-at-home orders were issued all around and social distancing was mandated, the rules varied between counties, states and countries even. The one universal precaution that was consistently enforced was the usage of masks while in any public place. Since the coronavirus is an airborne virus, the necessity of a mask to both protect oneself and others around them is paramount.

There have been a myriad of projects done to assess the spread of the virus and analyze its projection globally. There have also been projects related to how to effectively protect against it or to determine hospitals’ capabilities of supplying care to their patients. While immense efforts have been taken by all to get through the pandemic, the known truth is that the ‘New Normal’ with respect to everything in a person's life has been altered. This also leads to the new phase of the pandemic where we must explore life post-vaccination. While public venues can once again open up, safety measures such as the mask will still continue to be in effect.

Our vision with this project was with the forward thinking of this next phase that has not yet been explored in detail. Our aim was to apply the deep learning skills that we have learned all semester in order to come up with our own working solution that can be used in grocery stores, restaurants, malls, and movie theaters. We designed and developed code to perform face mask detection on crowds of people by incorporating various deep learning models in order to compare and contrast their performances as well as to explore the techniques of object detection and classification.

**Literature Review**

Classification and Object Detection are basic tasks that can be achieved with Deep Learning techniques. The combination of both allow for improved data analysis as now the detected objects can also be sorted into meaningful categories thus giving the data and the models used within a project to have further value in providing insights. There are various approaches that have been taken in order to achieve this combined task as well as those in relation to face mask detection that were researched thoroughly in order to develop the methodology for this project.

The Literature Survey matrix below shows a brief overview of the most relevant papers that were studied for the purpose of this project. Each paper had a slightly different approach, whether it be the objective or the model, however they all showed results with strong accuracy and precision scores. This led us to define our own approach based on what seemed to work and weed out the techniques that did not.

The first paper explored detection of face masks in real-time based on processed CCTV footage. Any model that must take in real-time input data is a challenging one to build as the data must be collected, pre-processed all within seconds of being sent through the main model as well to deliver results. The data scientists who worked with it used the model Single Shot Multibox Detector with MobileNetV2 Architecture and got a competitive accuracy of 92.4%. From this paper, we learned about this novel usage of the model and decided to take from it the usage of MobileNet architecture for our own. We did not use the Single Shot Multibox Detector (SSMD) as our input data was static images rather than real-time video imagery. Also, the model only performed a binary classification of mask or no mask.

Classification is one of the basic tasks in Machine Learning as well. In the second paper, we saw that a Hybrid approach was taken to perform the classification using the machine learning models of Decision Tree and SVMs and then object detection was done using the deep learning model of ResNet50. In machine learning, Decision Tree is one of the oldest and most reliable models when it comes to accurate classification of properly processed data. This foundation led to the predictably strong accuracy score of 99%. While the approach in this paper was more easier to comprehend in comparison to the others, we did not want to incorporate machine learning into the model as using a singular deep learning approach simplifies the coding process.

Table 1. Literature Survey Matrix

|  |  |  |
| --- | --- | --- |
| Objective | Model | Accuracy |
| Real-Time Face Detection and binary mask classification. | Single Shot Multibox Detector with  MobileNetV2 architecture (SSMDNV2) | F1 score - 93%  Accuracy - 92.4% |
| Hybrid DL/ML model for detection. | Decision Tree, SVM, ResNet50 | RMFD - 99.64%  SMFD - 99.49%  LFW - 100% |
| Face Detection and Binary Mask Classification | Sequential Convolutional Neural Network | Accuracy - 95.77% and Accuracy - 94.58% (respectively on two different datasets) |
| Serverless Edge Computing Detection | Edge computing (YOLO), NCNN, WebAssembly | Average Precision (AP) = 89% |
| Retina Face Mask | Lightweight NN MobileNet | Face/mask detection - 2.3% and 1.5% higher  Recall = 11.0% and 5.9% higher |
| Real- Time Recognition with CNN | VGG16 CNN | Accuracy - 96% |

**Approach**

The final approach for this project was an amalgamation of portions derived from all the papers that were researched for the literature survey that resulted in improving the key performance metrics of each model. The final model takes in images of individuals or crowds as input data and is able to not only detect the mask and classify the individual(s) as being safe or not, but it can also detect the presence of a mask that is worn correctly or incorrectly. This combination of a binary detection and classification results in a novel final product that can be utilized effectively in public places as a measure of upholding safety measures and having a meaningful impact.

As Data Science students, we did not feel that any one model could encompass the full scope of this project and we wished to explore different techniques as well. This is why we went for three Deep Learning models in total that would all perform the same task in order to compare and contrast their performance and determine which one was best suited for our purposes based on their respective classification reports. The three models chosen are: Basic Convolutional Neural Networks, MobileNet, and VGG16. The Basic CNN was the most straightforward choice for a simple binary classification and serves as the baseline model for this project. Mobile Net was chosen for its performance in the real-time detection and classification and the Retina Face Mask papers as it noticeably outperformed the baseline models. VGG16 is also a viable model approach as it has been shown to increase the accuracy when combined with CNN. In addition to all three model approaches, hyperparameter tuning and ensemble techniques were also utilized to observe how the models reacted in different settings and how this affected the resulting accuracy for detection and classification.

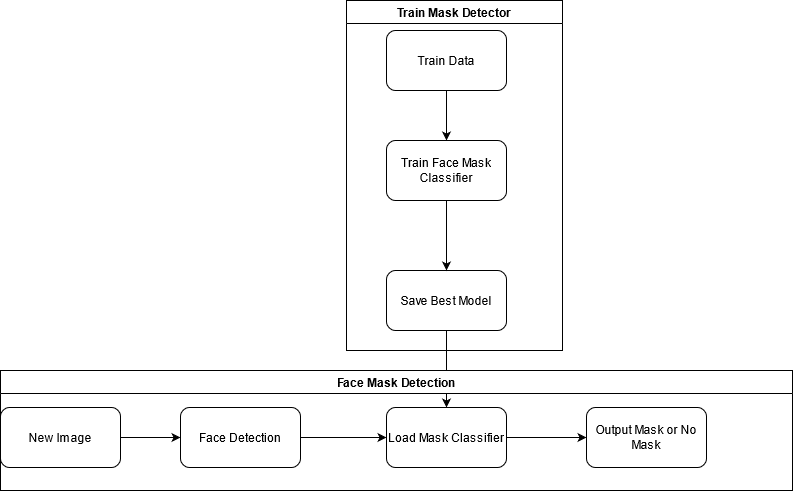


Figure 1. Process Flow

**Data Processing**

Based on the proposed approach for this project, all the models proposed are built to take in the same source dataset. The input data is in the form of a collection of images of individuals and crowds both wearing, not wearing, and/or incorrectly wearing their masks. Unlike a regular dataset in a flat file format that might contain numerical or categorical values for fields, image data cannot be processed with regular descriptive data analysis alone prior to being used in Deep Learning models. Before this image dataset can be fed into the models, as with all input data, it must go through a series of data processing steps that allow us to convert it into a usable form that can be analyzed and dissected into components that can be used for extracting further valuable information.

There are a lot of methods by which image data can be processed, however the simplest and most effective way is to utilize python libraries such as OpenCV, a popular Computer Vision library. With this library, we were able to refine the quality of the images by performing basic cleaning such as reducing noise, adjusting colors, and increasing image sharpness. We also were able to segment the images and isolate the face via functions meant for face/feature detection such as Haar Cascades function or the Caffe Model. The ultimate goal of image processing was to convert the newly adjusted and analyzed images into multi-dimensional arrays that can now be easily processed by the models. The arrays represent each image and contain values that correspond to features of the image such as RGB values and segments.

For this project, the source dataset was obtained from Github. The repository contains a dataset consisting of 2605 with mask images and 1937 without mask images. To begin with, basic descriptive analysis was performed including normalization and data augmentation. Not all the images were of great quality so areas of error were likely to be higher in darker images or images that were noticeably more blurry.

Dataset link :

<https://drive.google.com/drive/folders/1dmeNSjiHNKIN0AMMBa1Tsd1xRi8smfsd?usp=sharing>

**Solution Implementation**

**Train Mask detector**

First task in our project flow is to train mask detector models and identify the best model with highest accuracy and f1 scores. To identify masks in the image we have used Convolutional Neural Network (CNN) and its different architectures. We trained mask detector models with multiple images with single faces with labels as mask and no mask. Below are the details of the models we trained to detect masks in the face.

1. **Basic CNN :**

We developed our own basic CNN model using Keras library to classify images as mask or no mask. For training we have used images with a single person and hence it is a single object detection task. CNNs scan the image with learnable “filters” and extract more and more abstract features at each layer. Filters in early layers may for example detect edges or color gradients, while later layers may register complex shapes.

Our model consisted of 2 convolution layers with max pooling layer and relu activation functions. First CNN layer had 200 kernels and the second had 100 kernels , followed by a fully connected neural network with 100 neurons in the hidden layer.

Model was tuned using grid search hyperparameter tuning. With cv as 3 and were trained for 30 epochs each. We tried different parameters for the model. We started by developing a baseline model with 1 CNN layer and compared the results with this baseline model. Dropout layer was also added in the final model to avoid overfitting. Figure 2 shows the architecture of the final basic CNN model for mask detection.

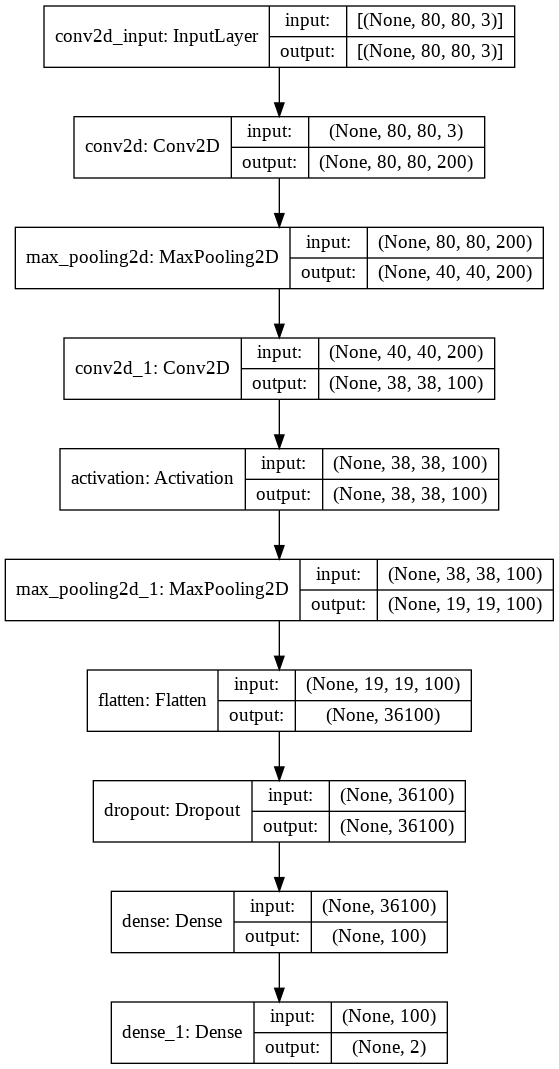


Figure 2. Basic CNN

1. **VGG16:**

The third model we developed is the VGG16 model which is a Convolutional Neural Network(CNN) architecture proposed by Karen Simonyan and Andrew Zisserman from the University of Oxford in 2014. The acronym VGG stands for Visual Geometry Group. The VGG16 model was used to win the ImageNet Large Scale Visual Recognition Challenge ([ILSVRC](http://www.image-net.org/challenges/LSVRC/)) competition in 2014. The VGG-16 is a 16-layer convolutional neural network. The model uses a set of pre-trained weights from ImageNet. In ImageNet, a dataset of over 14 million images belonging to 1000 classes, the model achieves 92.7 % top-5 test accuracy. The default RGB image input size for the VGG16 model is 224 x 224 pixels for three channels.

The network structure of VGG16 has thirteen convolutional layers, five Max Pooling layers, and three Dense layers, with a total of twenty-one layers. As a result, there are 16 layers of tunable or learnable parameters, i.e., 13 convolutional layers and 3 fully connected layers. So, VGG16 is the model's name for this reason. As shown in figure 3 the architecture of VGG16 follows the same design flow, the convolution and max pool layers are arranged. The most unique aspect of VGG16 is that, rather than having a huge number of hyper-parameters, they focused on having 3x3 filter convolution layers with a stride 1 and still used the same padding and maxpool layer of 2x2 filter stride 2.

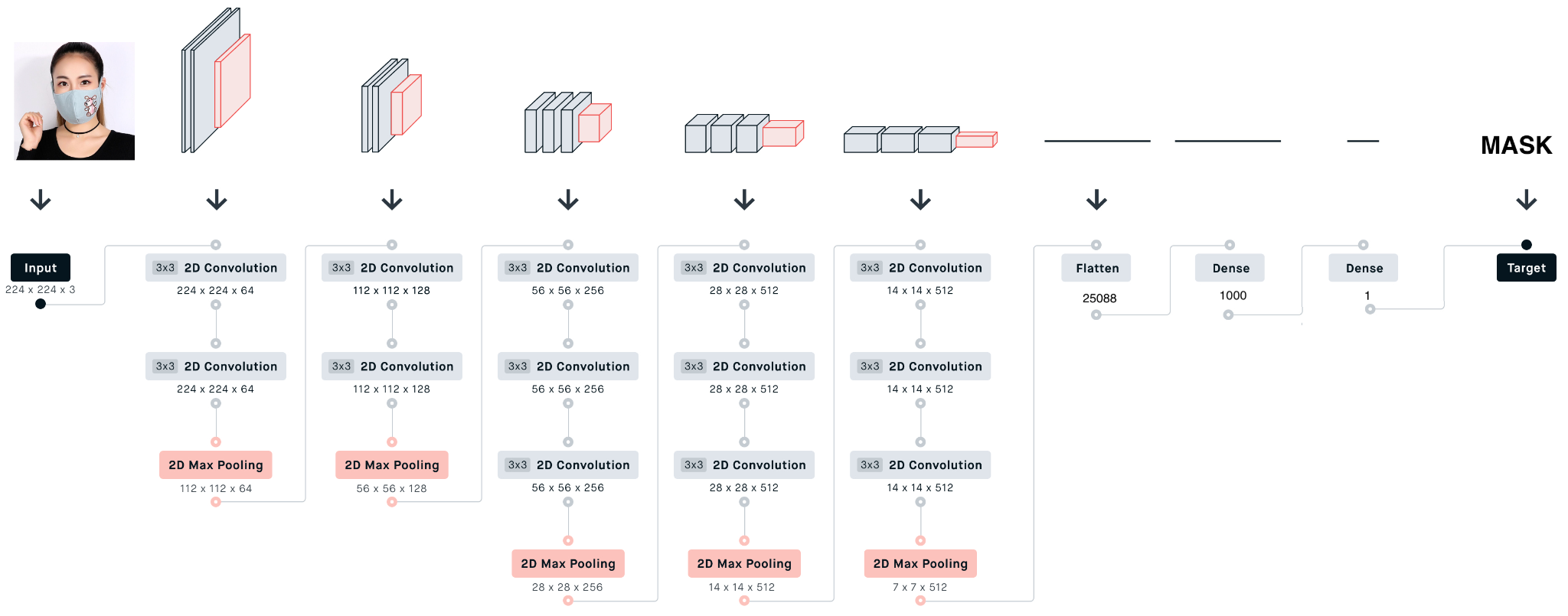


Figure 3. VGG16 Architecture

The VGG16 architecture is as follows, the input image shape 224x224x3 is passed through the first two layers are 3x3 convolutional layers, while the first two layers use 64 filters, resulting in a volume of 224x224x64 since the same convolutions are used. Filters are always 3x3 with a 1 stride. After that, a pooling layer with a max-pool of 2x2 size and stride 2 was used to reduce the volume's height and width from 224x224x64 to 112x112x64. After that, there are two more convolution layers with 128 filters. The new dimension is 112x112x128 as a result of this. The volume is reduced to 56x56x128 after using the pooling layer. Two more 256-filter convolution layers are added, followed by a downsampling layer that shrinks the size to 28x28x256. A max-pool layer separates two more stacks, each with three convolution layers. The 7x7x512 volume is flattened into a Fully Connected (FC) layer with 25088 channels and a softmax output of 1 classes after the final pooling layer [10].

For Mask detection, the keras library includes a pre-trained model that can be loaded and used for a variety of tasks, including transfer learning, image feature extraction, and object detection. We loaded the library's VGG16 model architecture and then added all of the weights to the appropriate layers. Before using the pretrained VGG16 model, data preprocessing is done as mentioned in the Data Processing stage. After 80-20% split of data for training and testing again from 80% data training data is split into 80-20% of training and validation data. So, as of total data - 64% is training data, 16% validation data and the rest 20% is used for testing data. The training and validation dataset of images are saved under the Dataset folder for further use of the datasets.

A pretrained VGG16 model is used from Keras applications till the last max pooling layer with volume 7x7x512. Before adding new layers to the base model in keras, each layer has a parameter called “trainable”. For freezing the weights of a particular layer, we set this parameter to False, indicating that this layer should not be trained because during backpropagation no need to train the base model. Deep Convolutional Neural Networks will take days to learn and require a significant amount of computing power. To solve this, we integrated VGG16 with Keras using Transfer Learning.

Transfer learning is a method of using a deep neural network algorithm that was previously trained on a similar problem to design a new model. The new model incorporates two or three layers from the earlier trained model. Here for the VGG16 model we have added 3 layers to the base model into Fully Connected layers. Following a stack of convolutional layers, three Fully-Connected (FC) layers are added: the first one is flattened which has 25088 channels, the second and third are Dense layers. The second layer has one 1000 channels with the RELU layer as activation while the third layer has one channel with the Sigmoid activation function and therefore has a total of 1001 channels. In all networks, the fully connected layers are configured in the same way.

To compile the VGG16 model using the compile function, it expects three main parameters: the optimizer, the loss function and the metrics of performance of the model. So, binary\_crossentropy for loss function as we are performing binary classification for mask detection and SGD is used as an optimizer for the model. It's important to use an optimizer to adjust the model's weights during the training process so that predictions are as accurate and optimized as possible. Stochastic Gradient Descent (SGD) is the classical optimization algorithm. The gradient of the network loss function is computed in SGD with respect to each individual weight in the network. Every forward pass through the network produces a parameterized loss function, and we use the gradients we've generated for each of the weights, multiplied by a learning rate, to move our weights in the direction of the gradient.

The decision on how many epochs a model should be trained is a common issue that occurs when training a neural network. A model with too many epochs may be overfitted, whereas a model with too few epochs may be underfitted. To tackle this problem, the concept of Early Stopping is used. We define an arbitrary large number of training epochs and stop training until the model output on a hold out validation dataset stops improving. Keras has a callback EarlyStopping that allows it to stop training early. When the EarlyStopping callback is activated, it will interrupt testing, but the model at the end of training might not be the best on the validation dataset. A second callback is required to save the best model found during training for later use called ModelCheckpoint callback. It can be used in a number of ways, but will just use it in this case to save the best model observed during training as specified by a performance measure on the validation dataset. The weights are stored in the “VGG16\_face\_mask\_detection\_model.h5” file so that there is no need to retrain the model in the future; instead, we may simply load the weight into the model [11].

1. **Modified MobileNetv2:**

MobileNetv2 is a deep learning model based on convolution neural networks. This is already trained on Image net dataset which is a collection of 1.4M images and can efficiently perform classification of images into 1000 classes. MobilenetV2 architecture has 53 convolution layers and 1 average pooling layer. It contains the fully convolution layer with 32 filters, followed by 19 residual bottleneck layers. It has mainly two components: inverted residual block and bottleneck residual block. There are three convolution layers in the bottleneck residual block. They are depth wise convolution, 1x1 convolution with Relu and 1x1 convolution without any linearity. There are stride 1 and stride 2 blocks and the internal layers of two blocks are as follows in figure. Each layer has an activation function (ReLU6) and a batch normalization layer except for the projection layer. The expansion factor used for most of the layers is six and c is the number of output channels[12]. A dropout ratio is set in the hidden layers to avoid overfitting. For an inverted residual block, a contrary procedure of above is followed and additionally the layers of these blocks are compressed. We can make use of all this knowledge and create a base model from mobilenetv2 for training our face mask detector. This process of reusing the existing pre-trained model and customizing it for our task is called transfer learning. The advantage of using this technique is you do not have to do it from scratch because the model has already learned feature maps quickly as it is already trained on the large dataset.

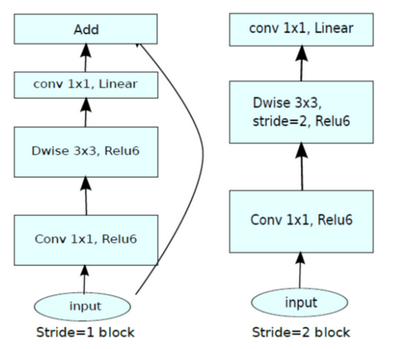


Figure 4. Stride 1 and Stride 2 blocks

Load the pretrained weights of ImageNet dataset using Keras. By applying transfer learning techniques, we are going to reuse the bottom layers of the base model. First layer is considered as base\_input and the last three layers of base output we have replaced it with a fully connected layer. Then, we have added new trainable head layers on top of the base model, and these layers are trained on the collected dataset so that it can determine the features to classify whether a person is wearing a face mask or not. The base layers of the pre-trained model are set as non-trainable so that weights will not be updated during the process of backpropagation whereas head model layers will be tuned. We first added a flatten head layer to convert pooled feature maps to single dimension features. With 0.5 as dropout ratio is added in the dropout layer and sigmoid activation function is used in the output layer since it is a binary classification problem. Finally, the base and head model are merged, and this fine-tuned model is used for training our face mask detection. Initialized the default parameters of learning rate=1e-4, number of epochs to train=20 and batch size as 32. Split the dataset into 65% training, 15% validation and 20% for testing. We used Adam optimizer and binary cross entropy as loss function for compiling the model. We also used callbacks to set early stopping and model checkpoints to save the best model while training. This is mainly to stop the training if loss starts to increase. It also avoids overfitting. The model is trained for 20 epochs and results are: loss: 0.1374, accuracy: 0.9846, val\_loss: 0.1460, val\_accuracy: 0.9889. The model is saved and is stored in "mask\_detection\_model.h5" for future use.

**Solution Evaluation**

All the models were evaluated on a test dataset and results are highlighted in the table below. We had a fairly balanced dataset for training and hence models were evaluated using accuracy, f1 score, precision and recall.

Table 2. Summary of Mask Detector models (Test Data)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **F1 score** | **Precision** | **Recall** |
| CNN using Keras | 0.9579 | 0.96317 | 0.96504 | 0.9613 |
| MobileNetV2 | 0.9922 | 0.9932 | 0.9922 | 0.9941 |
| VGG16 | 0.9767 | 0.9798 | 0.9732 | 0.9864 |

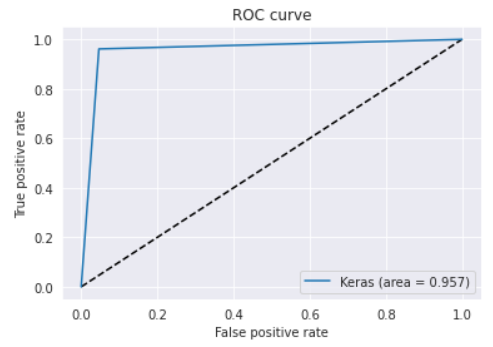
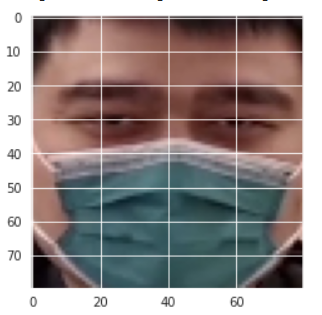


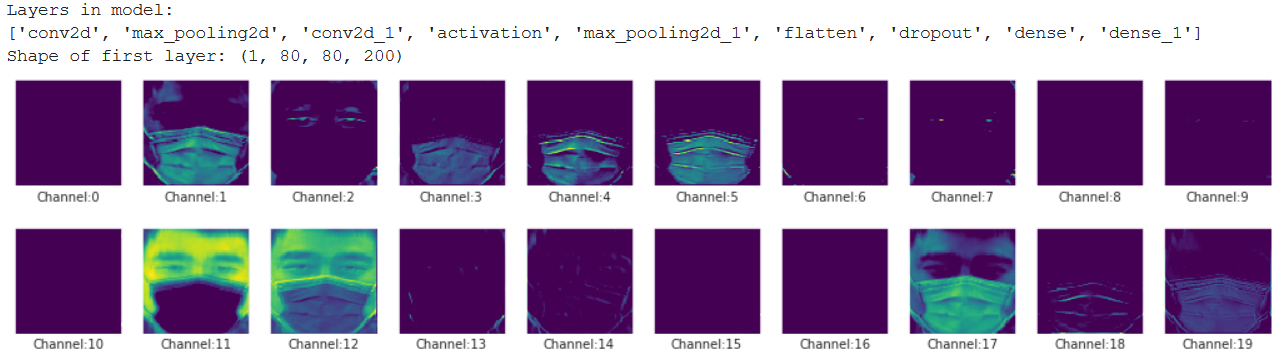
Figure 5. ROC for CNN Model

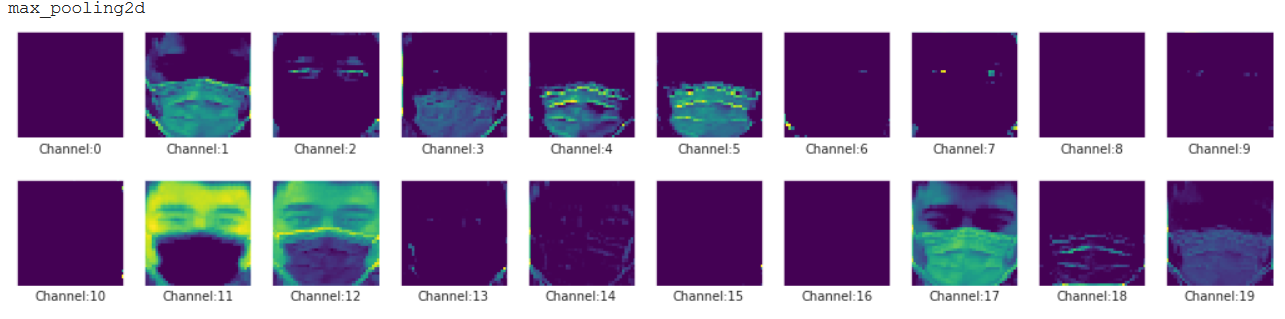
For model evaluation and validation the dataset was split into train, test and validation sets based on the 60-20-20 rule. Grid search method was used for tuning the basic CNN model. After adding an additional CNN layer into the initial baseline model, performance increased significantly. Additionally training epochs were increased to enhance the score of the model. Models converged fast after increasing kernel size and adding more CNN layers with drop out. Best model after parameter tuning gave 94% accuracy on test data, when we used sigmoid function in the final output layer as its a binary classification problem.We changed the model to use softmax function in the output layer, as it is just expansion of sigmoid for multi class problems. We found that performance of the model improved by 1%. Hence our best basic CNN model for mask detection had softmax in the outer layer and gave 95% accuracy on test data. Additionally we analysed the result of the basic CNN model by visualizing output of intermediate CNN layers for correct and wrong predictions. Sample output for first two and last layer of CNN for correct classification with mask on is:

Original Image:



Output of CNN layers for basic CNN model:

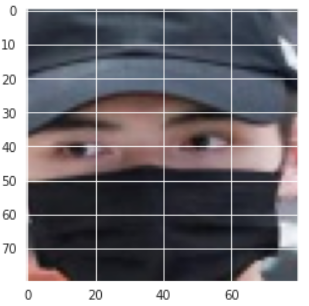




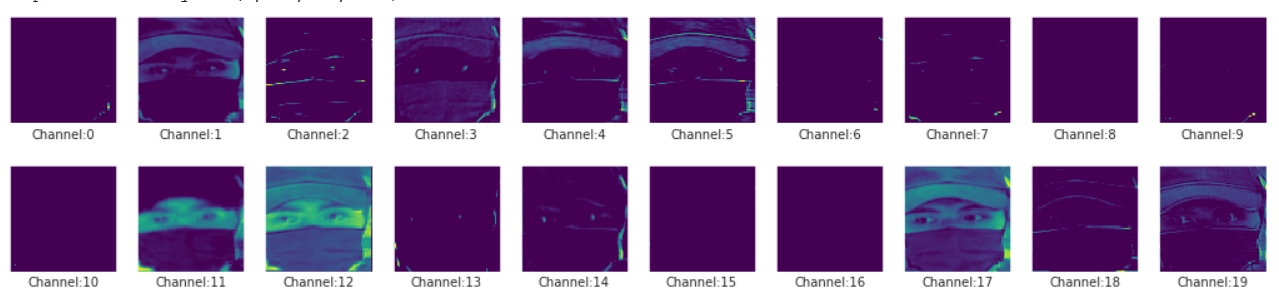


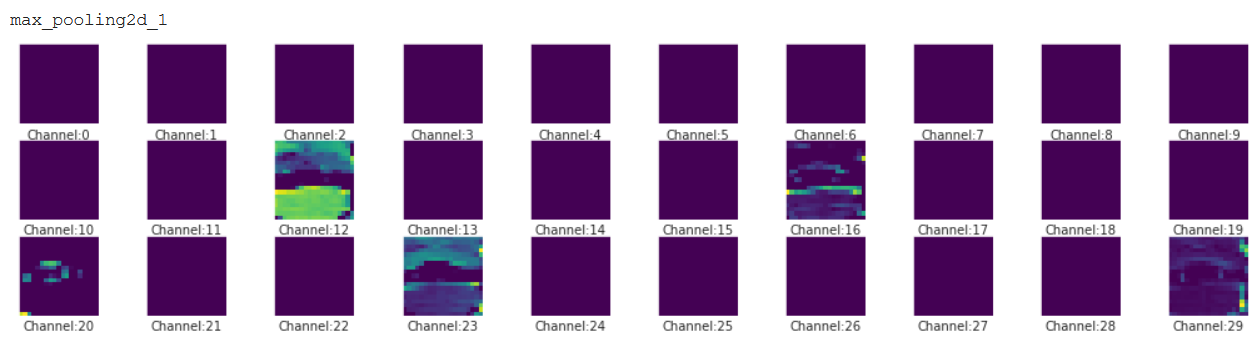
We can see that as we go down in the neural network, the output face is less recognisable and the model extracts more low level details in the image. Last image shows that the model was able to identify the mask in the image and its focus is on the mask in lower layers. Next we visualized misclassified data for first and last layer of CNN with mask on:

Original Image:



Output of CNN layers:





We can see that in the last visual of the last layer, the model can not separate the mask in the image and hence misclassifies it as no mask on. This evaluation and visualization helped us in selecting our next model. We used pre-trained models on our dataset using transfer learning and found significant improvement in performance. Ambiguities due to image resolution and image orientations were handled in pre-trained models.

For the VGG16 model, the model evaluation and validation of the whole dataset collected was split into 64-16-20% of training, validation and testing dataset. As the pre-trained VGG16 model was used with transfer learning on Fully connected layers with one flatten and two dense layers. The model was saved and the model was fit to generate the accuracy on the training and the validation data using 30 epochs. The accuracy on the training data was 98.2% and on the validation data it’s 99.17%. The Learning curve was plotted on the training and the validation loss function as log\_loss on Y-axis to the 30 epochs on X-axis as shown in figure 6. As shown in figure 6 the learning curve, the best model was found at 30 epoch.

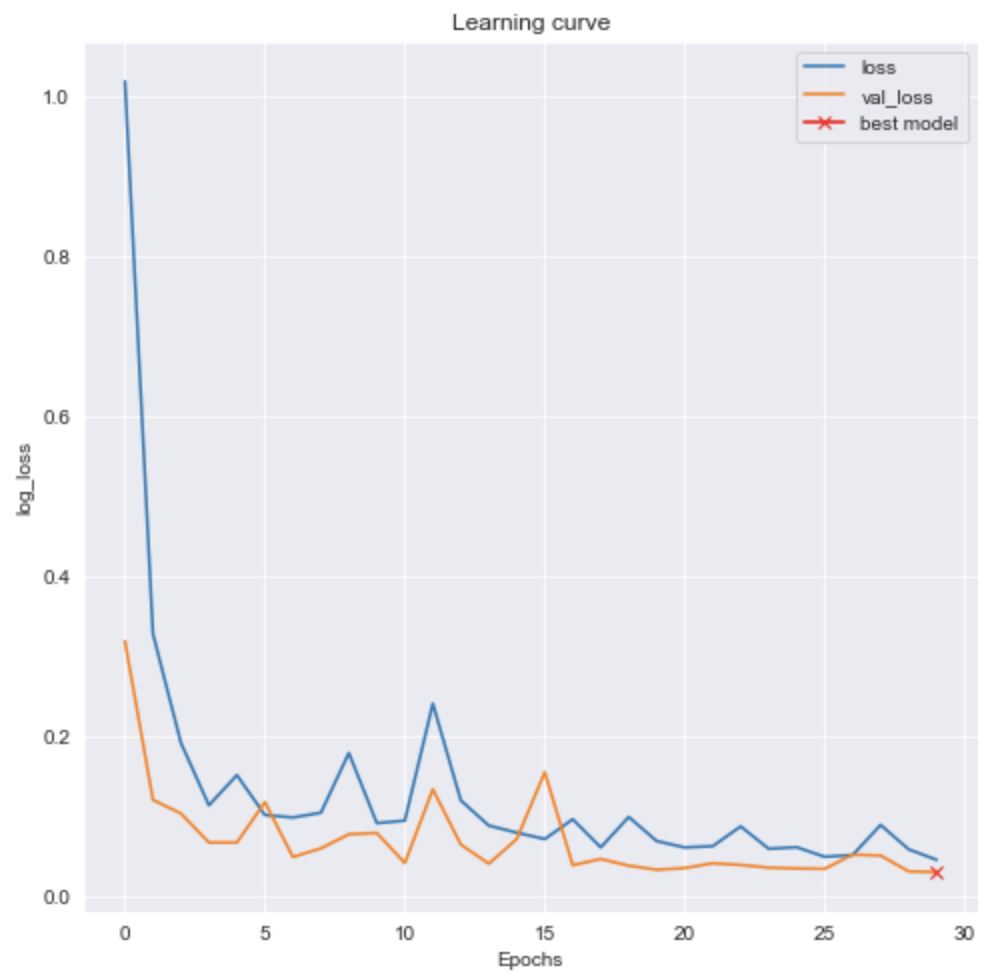


Figure 6. Learning Curve of VGG16 model

The VGG16 model was next evaluated on test data and the loss is 0.0632 and accuracy is 0.976. Again saving the model after finding the accuracy on train and test data where we used it in the future and no need to again train the model. From the confusion matrix predicted on the test data the number of correct predictions are 883 (True Positive and True Negative), whereas the number of misclassified predictions are 21 (False Positive and False Negative). The classification report predicted on the test data is accuracy with 97.67%, f1-score with 97.98%, precision with 97.32%, Recall with 98.64% and the Area Under the Receiver Operating Characteristic Curve (roc\_auc\_score) is 97.5%. The Receiver Operating Characteristic (ROC) curve is plotted as shown in figure 7. A ROC curve is a graphical plot used to show the diagnostic ability of binary classifiers. A ROC curve is constructed by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR). The true positive rate (TPR = TP/(TP + FN)) is the proportion of all positive observations that were correctly predicted to be positive. Similarly, the false positive rate (FPR = FP/(TN + FP)) is the proportion of negative observations that are wrongly predicted to be positive.

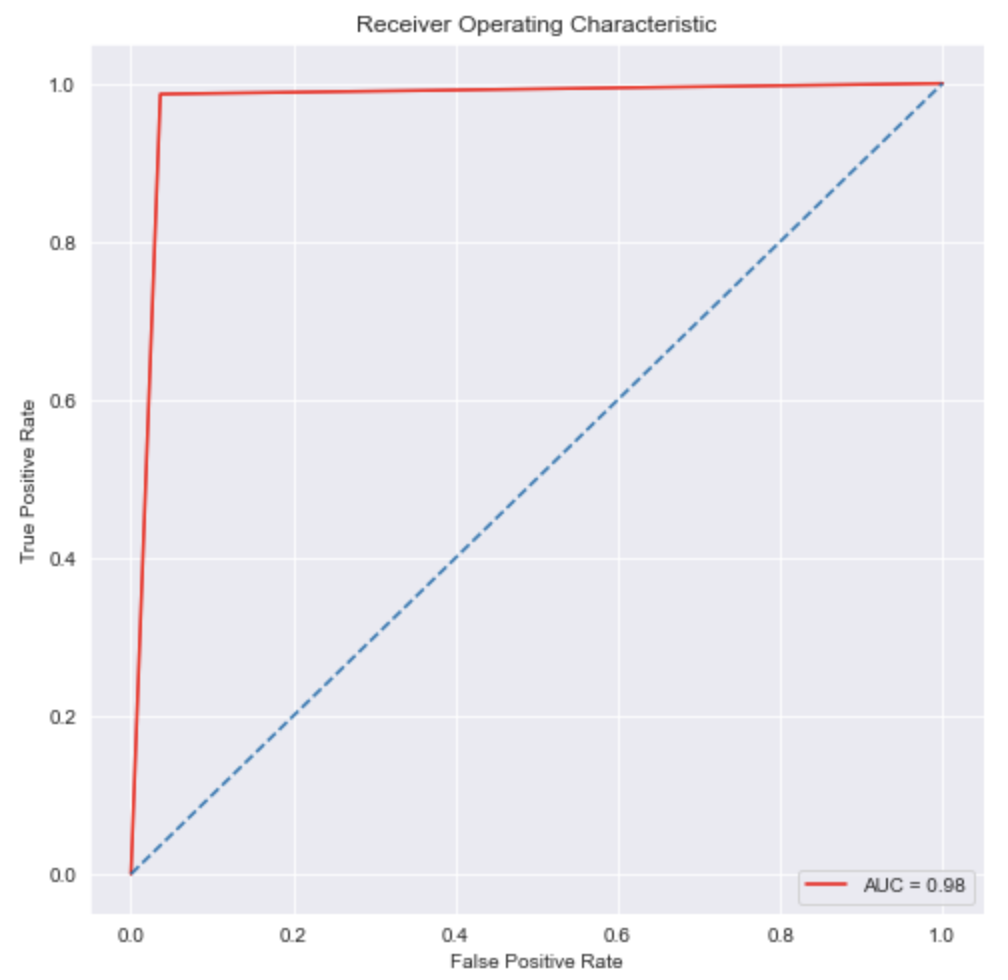


Figure 7. Receiver Operating Characteristic (ROC) curve

The VGG16 model had predicted correctly on test data images. And, also the model predicts the new images and it predicts correctly for the binary classifier of mask and no mask as shown in below figure 8.

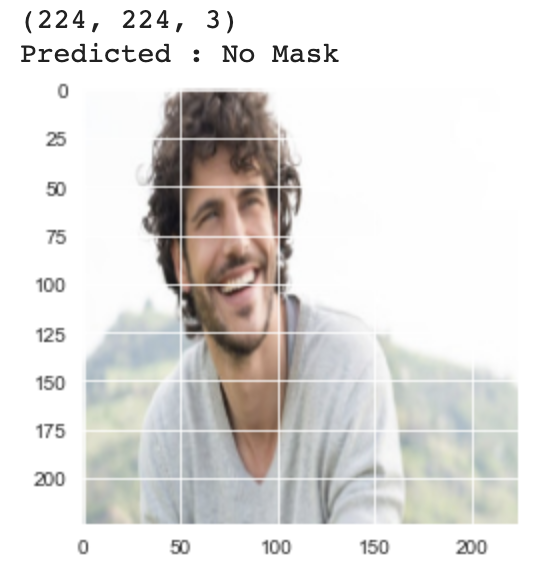
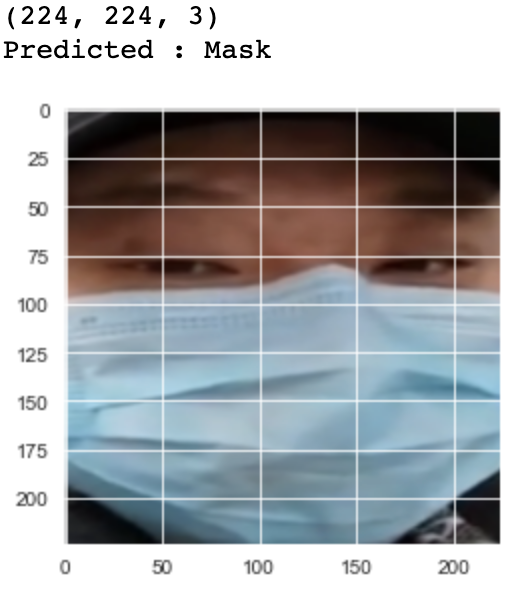


Figure 8. Prediction on New images

Even though the accuracy of the VGG16 model is good with 97.6% better than the basic CNN model using keras, there are two major drawbacks of the model. They are: the model takes too long to train the model and the network architecture weights themselves are quite large. So the MobileNetV2 model was used to predict the mask detection as the model is faster and also an effective feature extractor for object detection and segmentation.

To estimate the training and validation performance of the modified Mobilenetv2 model, the learning curves are plotted using matplotlib library. This metric is used to compare the loss function of training data against the loss function of validation data and checks if the model suffers from variance or bias error upon training with increasing samples[13]. The blue curve in the figure shows training accuracy and loss whereas the orange curve depicts the validation accuracy and loss. From the diagram 9, we can see that both training and validation scores converged to an optimal value. The accuracy curves are smooth, and the model gave a pretty good accuracy of 0.984, 0.988 respectively for training and validation. Similarly, the loss of training and validation curves are decreased gradually, and the model gave a relatively lower loss of 0.13, 0.14 after training iteratively for 20 epochs.

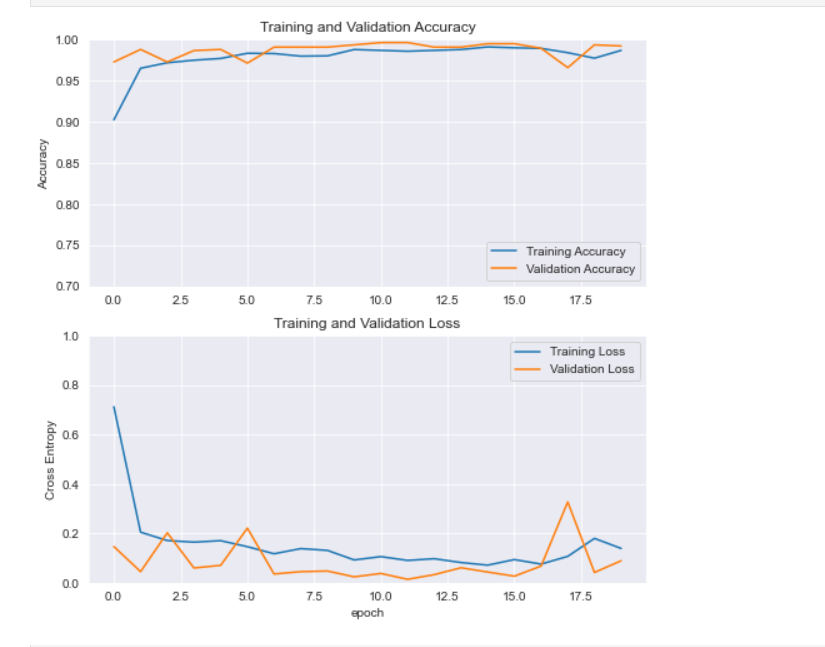


Figure 9. Learning Curves

Another common metrics used for evaluating a balanced dataset is accuracy, precision, recall and f1 scores. The classification report given by model on test data is 0.9922 - accuracy, 0.9932 - precision, 0.9922 - recall and 0.9941 - f1 score. Figure 10 shows the heatmap of the confusion matrix on predicted and actual labels. It has successfully predicted 383 true positives, 4 false negatives, 3 false-positive, and 514 true negative out of 904 images used for testing.

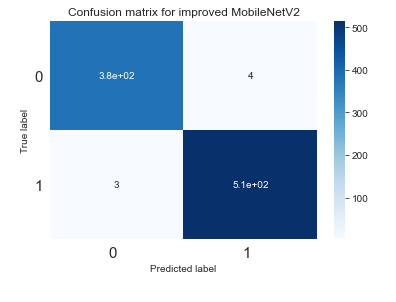


Figure 10. Confusion matrix

The robustness of a classification model is often measured by using Area under the curve (AUC) metric of a ROC curve (Receiver Operating Characteristics). Figure 11 compares the model’s false positive rate(fpr) with true positive rate(tpr) as shown below. The auc score ‘0.992’ of the roc curve is very close to the ideal roc curve but not exactly the same.

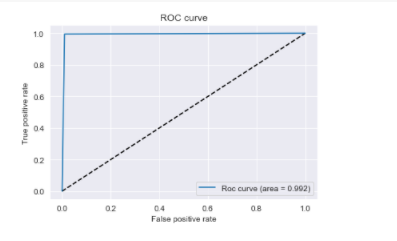


Figure 11. ROC curve

Figure 12 shows the visualization of correct predictions on test data. The model was able to classify the person wearing a face mask as ‘With Mask’ and not wearing a mask as ‘No mask’ correctly.

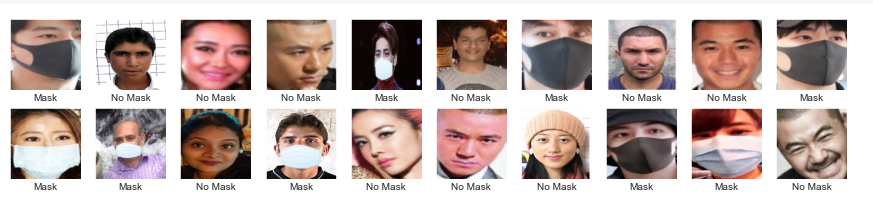


Figure 12. Correct predictions

**Face Mask Detection**

Finally, the best model was selected and used along with a pre-trained face detector model. We used a Single Shot MultiBox detector (SSD) model for face detection in live images and faces detected in each image were passed as input to the saved mask detector model to predict the bounding box around faces in image with label as mask or no mask. We tried testing on a few multiple face images. Figure 13 shows that our integrated model was able to predict the bounding boxes with the correct label for each of the faces detected in the image. Figure 14 is tested using a single face image and the model gives the accurate bounding box and classification result for this particular image. Finally, we have deployed our integrated model on the Flask Local as shown in Figure 15, so that it will be easy for others to access through an application programming interface (API) to make predictions.

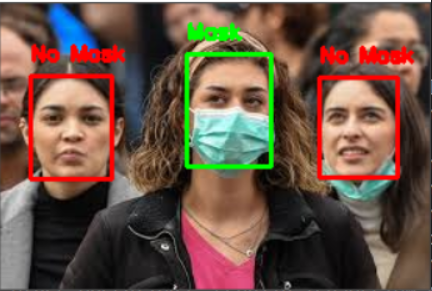


Figure 13 : Testing on multiple faces



Figure 14 : Testing on single face

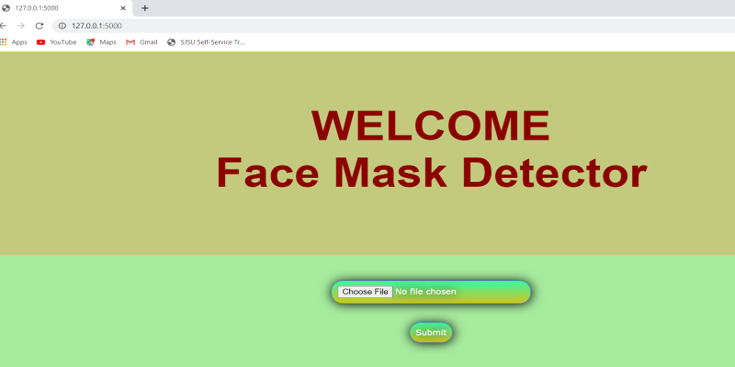


Figure 15 : Model output using Flask Local

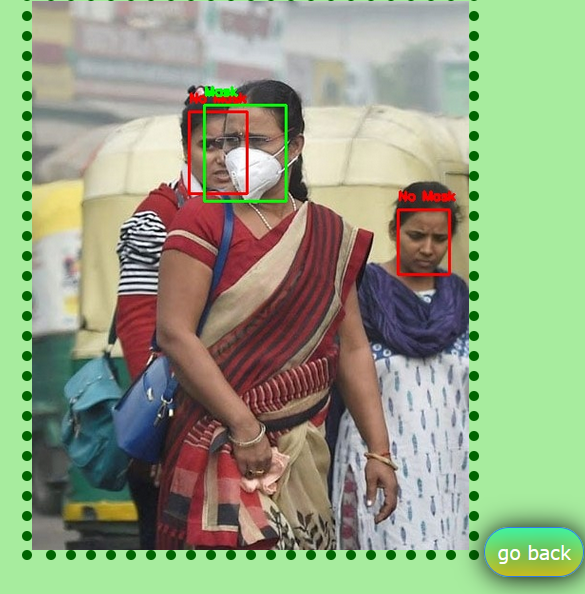


Figure 16. Result Using Flask

Figure 16. Shows the result using Flask application. This image is outside of our dataset which we picked from the internet. This result shows that the model is able to detect even overlapping faces and is able to predict correctly if a mask is present or not.

**Impact**

As we move into the next phase of the COVID-19, life post-pandemic is the main focus. There is a necessity for every individual and every institution to make efforts on their part in order to ensure that society is taking necessary precautions for safety and cleanliness in order to reduce the number of related cases and deaths until they are numbers of the past. This is a call to action for all to change behaviors and attitudes they may have held before to now suit this ‘New Normal’ by taking part in practices such as social distancing, frequent hand washing/ hand sanitizing, and wearing masks in all crowded public places--especially if those places are indoors. While this sense of responsibility may come naturally to many, there are always those who oppose or those who no longer feel the need for such measures after receiving vaccinations.

There are many individuals across all ages that are ultimately unable to receive the Coronavirus vaccines due to pre-existing conditions or other illnesses that prevent them from physically being able to take them. These individuals rely on what is known as Herd Immunity to protect them, a situation where all those that can be vaccinated receive them so that they cannot pass on the virus to those who cannot. If members of society who oppose the vaccine or do not see the point in taking them fail to do their duty, they put in harm not only themselves but also those individuals who are more prone to being infected by them. This is why the need for tools that can help ensure the safety of all in shared places that are meant for all to enjoy and use are paramount going forth.

We developed three cohesive models that are all able to detect face masks, or the lack thereof, on individuals and crowds and their effectiveness based on whether they are worn correctly or not. The three models chosen were basic Convolutional Neural Network (CNN), MobileNetV2, and VGG16. Based on the key performance indicators we saw that the basic CNN model had an accuracy of 95.8% and a precision of 96.5%. The MobileNetV2 model had an accuracy of 99.2 % and precision of 99.2%. The VGG16 model has an accuracy of 97.6% and a precision of 97.3%. These are strong accuracies across the board for all three models individually which proves to suit the purposes of this project. However, when considered under comparison, we see that the MobileNetV2 model has the highest accuracy and precision levels therefore resulting in being the best model as well.

With such strong performance metrics, we can confidently conclude that our models are ready to be handed over to being used as a tool that can be put in place in public areas such as grocery stores, malls, movie theaters, and even hospitals to help monitor public safety and bring the pandemic well into control. While time is the key factor post-pandemic to help reduce the number of cases and Coronavirus related deaths that every country in the world is dealing with, this tool is key in helping promote the notion that the goal is beyond simply trying to end this pandemic. The ultimate goal is to ensure that the world is not distraught by another one any time soon. With measures such as this project and the combined efforts of everyone in society, even if there were to be an outbreak of a new pandemic, this time, the world would be deemed more than prepared.

Comprehensive Final Project Demo Link: <https://youtu.be/ee-YQ3L9uac>

**References**

1. World Health Organization. (2021). WHO Coronavirus (COVID-19) Dashboard. Retrieved April 30, 2021, from https://covid19.who.int/
2. Nagrath, P., Jain, R., Madan, A., Arora, R., Kataria, P., & Hemanth, J. (2021). SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2. *Sustainable cities and society*, *66*, 102692. <https://doi.org/10.1016/j.scs.2020.102692>
3. Loey, M., Manogaran, G., Taha, M. H. N., & Khalifa, N. E. M. (2021). A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic. *Measurement*, *167*, 108288. https://doi.org/10.1016/j.measurement.2020.108288
4. A. Das, M. Wasif Ansari and R. Basak, "Covid-19 Face Mask Detection Using TensorFlow, Keras and OpenCV," *2020 IEEE 17th India Council International Conference (INDICON)*, New Delhi, India, 2020, pp. 1-5, doi: 10.1109/INDICON49873.2020.9342585.
5. Wang, Z. (2021, January 4). WearMask: Fast In-browser Face Mask Detection with Serverless Edge Computing for COVID-19. Retrieved April 3, 2021, from https://arxiv.org/abs/2101.00784
6. Singh, S., Ahuja, U., Kumar, M. *et al.* Face mask detection using YOLOv3 and faster R-CNN models: COVID-19 environment. *Multimed Tools Appl* (2021). <https://doi.org/10.1007/s11042-021-10711-8>
7. Pooja S., Preeti S. (2021) Face Mask Detection Using AI. In: Khosla P.K., Mittal M., Sharma D., Goyal L.M. (eds) Predictive and Preventive Measures for Covid-19 Pandemic. Algorithms for Intelligent Systems. Springer, Singapore. <https://doi.org/10.1007/978-981-33-4236-1_16>
8. Dataset: https://github.com/balajisrinivas/Face-Mask-Detection/tree/master/dataset
9. Dataset: <https://github.com/prajnasb/observations/tree/master/experiements/data>
10. Prabin Nepal. (2020, July 30). VGGNet Architecture Explained. <https://medium.com/analytics-vidhya/vggnet-architecture-explained-e5c7318aa5b6>
11. Sachin Mohan. (2020, August 10). Keras Implementation of VGG16 Architecture from Scratch with Dogs Vs Cat Data Set. https://machinelearningknowledge.ai/keras-implementation-of-vgg16-architecture-from-scratch-with-dogs-vs-cat-data-set/
12. A. Mohan, “Review On Mobile Net v2,” *Medium*, 01-Jul-2020. [Online]. Available: https://medium.datadriveninvestor.com/review-on-mobile-net-v2-ec5cb7946784.
13. “3.4. Validation curves: plotting scores to evaluate models¶,” *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/learning\_curve.html