Application Development for Mask Detection and Social Distancing Violation Detection using Convolutional Neural Networks

Gokul Sudheesh Kumar and Sujala D. Shetty
Department of Computer Science, Birla Institute of Technology & Science, Pilani, Dubai Campus, Academic City, Dubai, U.A.E.

Keywords: Machine Learning, YOLO Object Detection, Firebase, TensorFlow.

Abstract: This project aims to detect face masks and social distancing on a video feed using Machine Learning and Object Detection. TensorFlow and Keras were used to build a CNN model to detect face masks and it was trained on a dataset of 3800 images. YOLO Object detection was used to detect people in a frame and check for social distancing by calculating the Euclidean distance between the centroids of the detected boxes. Developed an Android app named "StaySafe" where the user will be notified and can monitor the violations. For this purpose, Firebase was used as the backend service. If a violation is detected it will upload the image to a Firebase Cloud Storage with a notification, and the user will be able to view these images on their Android app along with the date and time. Firebase Cloud Messaging service was used to send notifications which will be handled in the android app. The app offers various features like viewing history, saving the image to the device, deleting the images from the cloud etc. A heat map can also be viewed which highlights crowded regions which can help officials identify the regions that need to be sanitized more often.

1 INTRODUCTION

The year 2020, has brought us a lot of challenges, especially in the working sectors. Many establishments are switching to a work from home-based environment. There are still some businesses that opened after lockdown and continuing its normal operation like restaurants, few schools and universities, construction sector, and a few offices all ensuring workplace safety such as wearing masks and social distancing.

The health authorities are working hard on ensuring that these businesses follow all the protocols for keeping the employees safe. They are conducting regular inspections and even shutting down these establishments if they keep violating the safety protocols.

Many of these establishments are working towards the automation of detecting such violations, thereby reducing the time and labour spent for the same. The main aim of this project is to detect violations such as not wearing a mask or not following social distancing in a workplace and notify the officials through an Android app.

Technology has advanced tremendously over the past century, everything starting from the Internet of Things (IoT) (Raj, 2021) to machine learning and deep learning. CNN is used in various fields like medical (Duran-Lopez, 2019), marine science (Sung, 2017) and many other applications (Hansen, 2017) and has become a prominent domain of machine learning.

This project was implemented using Keras and TensorFlow where the CNN (Convolutional Neural Network) model was trained for face mask detection and the YOLO Object Detection was used for social distancing detection. An android app named "StaySafe" was developed with the help of Firebase which is the backend service implemented for pushing notifications and storing these detected images, from where the user will be able to view them on their app.

The rest of the paper is arranged as follows: Section 1 gives a brief introduction to the project. Section 2 reviews some work related to this project. Section 3 explains the methodology and implementation of the project. Section 4 talks about Social Distance violation detection. Section 5 talks about Firebase which is the back-end service used to store detected images and send notifications to the android app. Section 6 talks about the Android App. Section 7 discusses the applications. Section 8 is

Kumar, G. and Shetty, S.
Application Development for Mask Detection and Social Distancing Violation Detection using Convolutional Neural Networks.
DOI: 10.5220/0010483107600767
In *Proceedings of the 23rd International Conference on Enterprise Information Systems (ICEIS 2021) - Volume 1*, pages 760-767
ISBN: 978-989-758-509-8

conclusion and future scope of work. Section 9 is references.

2 RELATED WORK

This section reviews some of the related works that implements CNN with the help of TensorFlow, Keras and YOLO Object detection and improvements in object detection.

Akanksha Soni et al. (Soni, 2020) developed a model that detects whether a person is wearing a helmet in real time thereby, detecting any violations. This project was also implemented with the help of TensorFlow, Keras and OpenCV. Their proposed model showed major improvements when compared to some previous models that gave wrong predictions whenever a rider wears clothes over their face. They achieved an overall accuracy of 98% when tested.

S Chen et al. (Chen, 2020) implemented a model with the help of TensorFlow to identify ID card numbers. With the help of OpenCV the image of an ID card is preprocessed and the number on the ID card is recognized and given as output with the help of a trained CNN model. When tested it was observed that training speed is fast and the accuracy is high.

Emily Caveness et al. (Caveness, 2020) developed TensorFlow Data Validation (TFDV) which offers a scalable solution for data analysis and validation for machine learning. It is deployed in production which is integrated with TensorFlow Extended (TFX), which is an end-to-end ML platform. Their system has gained a lot of traction ever since they open sourced their project. Other open-source data validation systems such as Apache Spark were also heavily inspired from their project. Apache Spark packs with built-in modules for streaming and has a fast, easy to use system for big data processing.(Nair, 2018)

Yonghui Lu et al. (Lu, 2020) proposed an efficient YOLO Architecture, YOLO-compact for a real time single category detection. As we know in most practical applications, the number of categories in object detection is always single and the authors aimed to make detections faster and more efficient for these scenarios. By performing a series of experiments, the authors were able to come up with an efficient and compact network with the help of YOLOv3. It was observed that YOLO-compact is only of 9MB size, about 26 times smaller than YOLOv3, 6.7 times smaller than tiny-yolov2 and 3.7 times smaller than tiny-yolov3. The average precision of YOLO-compact is 86.85% which is significantly higher than other YOLO models.

M. B. Ullah (Ullah, 2020), proposed a CPU-based YOLO object detection model that is intended to run on non-GPU computers. In the proposed method, the author optimized YOLO with OpenCV in a way that real time object detection can be much faster on CPU based computers. Their network architecture comprises 2 Convolutional layers each followed by pooling layers and 3 fully connected layers. Their model detects objects from videos in 10.12 to 16.29 FPS with 80-99% confidence in CPU-based computers.

3 METHODOLGY AND IMPLEMENTATION

Figure 1 depicts the project architecture.

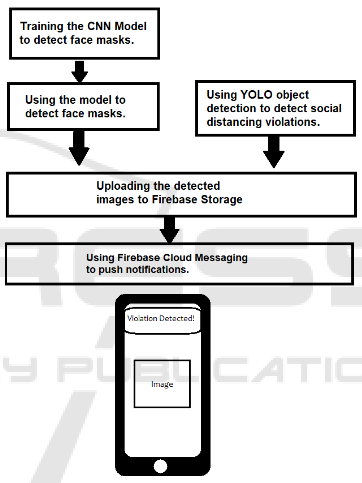

Training the CNN Model to detect face masks.
Using the model to detect face masks.
Using YOLO object detection to detect social distancing violations.
Uploading the detected images to Firebase Storage
Using Firebase Cloud Messaging to push notifications.
Violation Detected!
Image

Figure 1: Project Architecture.

3.1 Face Mask Detection

This project uses TensorFlow and Keras to train a CNN model for detecting face masks.

3.2 TensorFlow and Keras

TensorFlow is an open-source platform that is used for Machine Learning, created by the Google Brain team. It is explicitly used for complex numerical computation, that packs together a bunch of machine-learning and deep learning models and algorithms.

It can be used for a variety of applications such as classifying handwritten digits, object detection, image recognition, natural language processing (Natraj, 2019) by training and running deep neural networks.

Keras which acts as an interface for TensorFlow is an open-source library that provides an efficient way of implementing neural networks. It consists of useful functions such as activation functions, and optimizers.

3.2.1 How Does TensorFlow Work?

With the help of TensorFlow, developers can create dataflow graphs which are structures that show how data passes through the graph, or a series of nodes. Think of each node as a mathematical operation and each edge representing a multidimensional data array or a tensor.

This can be easily implemented in python where these nodes and tensors act as objects. However, the mathematical operations are performed in C++ binaries which shows an optimal performance. Python takes care of directing the traffic and combines them to work together as a unit.

TensorFlow can be run on multiple platforms such as in a cloud, a local machine, CPUs or GPUs, iOS, and Android devices. It can also be run on Google's custom TensorFlow Processing Unit (TPUs). The trained models can be run on any system for predicting results.

TensorFlow 2.0 which was released in October 2019 made many significant changes from user feedback. It works more efficiently and is more convenient with simple Keras API for training models and better performance. With the help of TensorFlow Lite, it is possible to train models on a wide variety of devices.

3.3 Training the Face Mask Detection Model

In this project a convolutional neural network is used to detect face masks. The neural network takes in the input image as a frame from the video, processes it and classifies it under two categories: mask and no mask. The model was trained using 3800 images, 1900 images each for "with mask" and "without mask" categories.

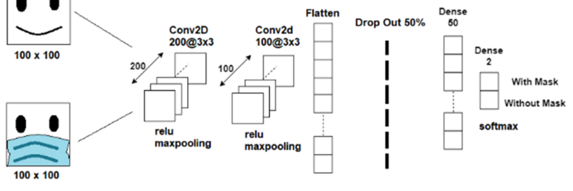


Figure 2: Network Architecture.

The network consists of two convolution layers each followed by a relu activation function and a max pooling layer. Figure 2 shows the network architecture.

Relu function (Rectified Linear Unit) is introduced for non-linearity in the convolutional network. It is shown as:

$$f(x) = \max(0, x) \tag{1}$$

Pooling layers will reduce the number of parameters if the images are large. Spatial pooling also referred to as sub-sampling or down-sampling helps to reduce the dimensionality of each map making sure important information is retained. It can be of different types:

- Sum Pooling
- Max Pooling
- Average Pooling

Max pooling will take the largest element from the rectified feature map.

The data is then flattened by converting it to 1-dimensional array which is passed as the input to the final output layer. To help prevent overfitting, the network ignores a certain percentage of neurons during training. In this case the network drops out 50% of neurons. These units are not considered during some forward or backward pass.

The final output layer takes the values and transforms them into a probability distribution, this is achieved with the help of softmax function. This function is helpful when it comes to classification problems or when dealing with multi-class classification problems. The final predicted class is the item in the list whose confidence score is the highest.

It is represented as:

$$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2}$$

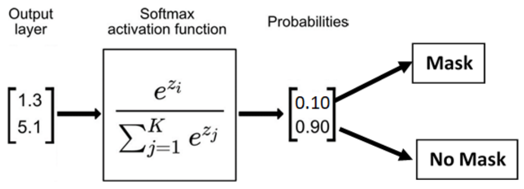

Output layer
Softmax activation function
Probabilities
1.3
5.1
$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
0.10
0.90
Mask
No Mask

Figure 3: Softmax Activation Function.

The final prediction will be based on the class that has the highest probability. The model was then compiled using Adam optimizer and trained for 20 epochs with a learning rate of 0.001 and an accuracy of 96.35% was observed on the validation set.

Figure 4 plots the accuracy and loss, respectively. (Goodfellow, 2016) (Belciug, 2020)

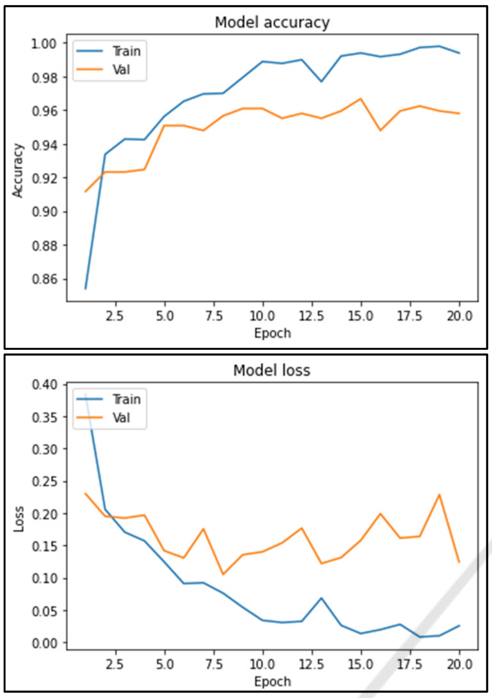


Figure 4: Accuracy and Loss.

3.4 Detection

OpenCV was used to capture the video and the trained model was loaded using TensorFlow.

A pre-trained model was used to detect faces in a video. The weights were initialized from the configuration file with the help of OpenCV.

After getting the bounding boxes for the faces in the frame, that region of interest is cropped out from the main frame, reshaped, and is then passed to the model.

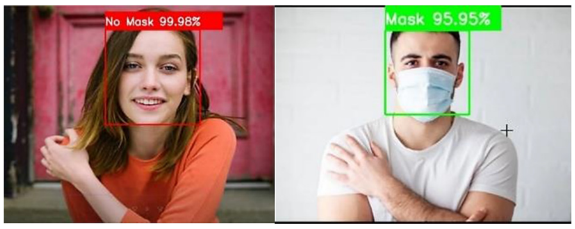


Figure 5: Detecting Face masks (https://en.wikipedia.org/wiki/File:Victoria_Pedretti.jpg).

Every time a violation is detected that frame will be saved locally which will later get uploaded to Firebase Storage.

4 SOCIAL DISTANCING DETECTION

This project uses YOLO Object detection for detecting people in a frame and finds the Euclidean distance between the centroids of the detected boxes.

4.1 YOLO Object Detection

You Only Look Once (YOLO) is an effective real time object detection system. It considers object detection as a regression problem and finds the class probabilities for each of the bounding boxes. In one evaluation the neural network predicts bounding boxes and class probabilities from the image, hence the name YOLO. The base model detects images at an astonishing speed of 45 frames per second whereas a smaller version called Fast YOLO detects at 155 frames per second. It performs better than other detection methods such as Deformable Part Models (DPM) and Region-based convolutional neural networks (R-CNN). There are two types of algorithms that work towards object detection- Algorithms based on classification, and Algorithms based on regression. YOLO falls under the latter category.

4.1.1 How Does YOLO Work?

A single neural network is used to combine separate parts of object detection. It takes in features from an image to predict each bounding box. The detection is modeled as a regression problem where an image is divided into SxS grids. In Fig. 7, the authors set the value of S as 7. This can be changed in the YOLO configuration file. Each grid predicts bounding boxes (B), their confidence scores, and their class probabilities (C). These predictions are encoded as an S x S x (B*5 + C) tensor.

Each bounding box has 5 predictions: (x, y, w, h) and confidence. Each grid cell predicts conditional class probabilities (C) as P (Classi | Object).

The architecture for this Convolutional Neural Network was inspired from GoogLeNet model that is used for image classification. Just like GoogLeNet their model was implemented using 24 convolutional layers that helps to extract features from the image followed by 2 fully connected layers to predict the output probabilities and coordinates. They also came up with a faster version on YOLO named Fast YOLO that uses a CNN with less layers (9 instead of 24) and less filters for those layers. Other than that, the training and testing parameters were the same between YOLO and Fast YOLO.

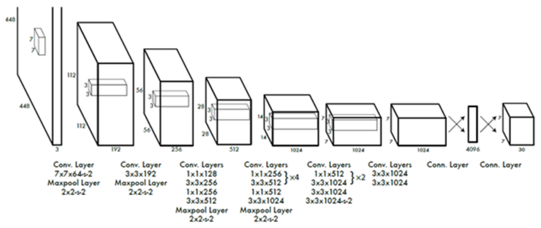

Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 ×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 ×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

Figure 6: The YOLO Architecture (Redmon, 2016).

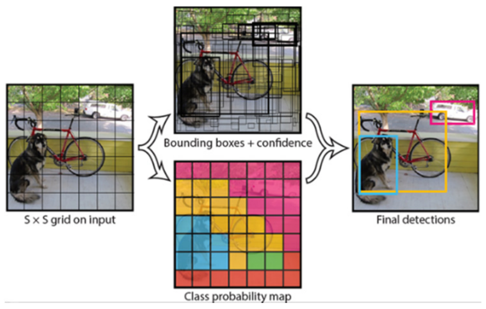


Figure 7: The YOLO Model (Redmon, 2016).

4.2 Detection

This project uses YOLO v3 to detect people. OpenCV python package was used to read the weights and set up the model using the configuration file.

```
net=cv2.dnn.readNet("./YOLO/yolov3.weig
hts", "./YOLO/yolov3.cfg")
```

The classes were read from "coco.names" file which contains the names of 80 different classes such as person, bicycle, car etc. Open CV was used to capture the video as shown below. This project was tested on some sample videos, on the webcam and on an IP camera as shown below.

```
cap=cv2.VideoCapture('./Videos/Example
4.mp4')
cap = cv2.VideoCapture(0)
cap=cv2.VideoCapture("http://192.168.2.
2:8080/video")
```

In each frame, the people are detected, and boxes are drawn along with their centroids. The centroids for each box were calculated and appended to a list. Red boxes indicate that the person is violating social distancing as shown in Fig. 8 below.

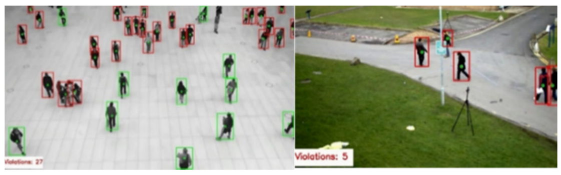


Figure 8: Detecting Social Distancing Violation.

Using Euclidean distance, the distance between the centroids can be calculated with the help of the scipy package.

```
D = dist.cdist(centroids, centroids,
metric="euclidean")
```

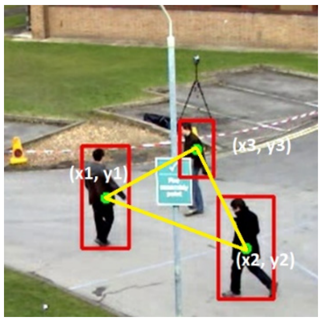


Figure 9: Calculating the Euclidean Distance between centroids.

Where D is a 2-D array as shown in the table below:

Table 1: 2-D array containing the Euclidean distance between each point.

	(x1, y1)	(x2, y2)	(x3, y3)
(x1, y1)	0	5	3
(x2, y2)	5	0	5
(x3, y3)	3	5	0

Where each element in the array is calculated according to the formula mentioned below:

$$dist\left((x_i, y_i), (x_j, y_j)\right) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (3)$$
$$i, j = 1, .. N\ detections$$

From the above equation, i and j represents rows and columns in the 2-D array, respectively.

If this distance is less than a certain threshold value, the person is violating social distancing.

We can estimate this threshold value by finding the ratio as follows:

Assuming the pixel density to be 96 dpi, we can deduce that 1 pixel = 0.26458333 mm or 1 mm = 3.7795 pixels.

Taking the ratio,

$$Ratio = \frac{\text{Real Height of the person (mm)}}{\text{Pixel Height of the person in the frame (mm)}} \quad (4)$$

Now to estimate the pixel distance for 1 meter (1000 mm):

$$1000/_{Ratio} \times 3.7795 \quad (5)$$

This threshold value may or may not be accurate and it needs to be increased or decreased accordingly.

Every time a violation is detected that frame will be saved locally which will later get uploaded to Firebase Storage.

4.3 Heat Map

Heat map is used to detect if people are crowding in a specific area and hence violating social distancing norms. By looking at the heat map on their app, it can help officials identify the places that need to be sanitized more often.

To plot the heat map, the first frame is read and every iteration the boxes are drawn on that frame. Red boxes indicate densely populated and sparsely populated regions are shown by blue boxes. When the program terminates this image is saved which is then uploaded to Firebase storage.

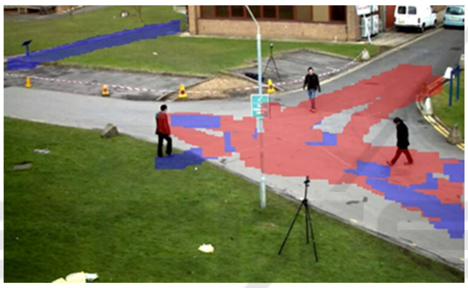


Figure 10: Heat Map showing densely and sparsely populated areas.

5 FIREBASE

5.1 Introduction

Firebase is a useful platform for mobile and web applications. It can be a back end for many services such as user authentication, data storage, static hosting, real-time database etc. It also offers a machine learning kit which has face detection, image labeling, object detection, text recognition, digital ink recognition and pose detection. It provides a user-friendly platform for building mobile and web apps.

This project uses cloud storage and cloud messaging to store detected images and push notifications to the app.

5.2 Cloud Storage

It is a simple, powerful, and cost-effective storage service that is built for Google scale. The Firebase SDKs for cloud storage adds security to uploads and downloads for the app regardless of the network quality. These SDKs can be used for storing audio, video, images etc. Google Cloud Storage can be used to access the same files on the server.

The uploaded files can be accessed from both Firebase and Google cloud and each file is stored in a Google Cloud Storage Bucket. This allows the developer to download and upload files from mobile clients through Firebase SDKs and makes it possible to do server-side processing using Google Cloud.

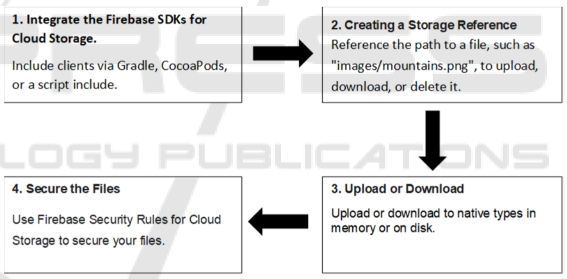
This project uses the Pyrebase python package to initialize the storage reference.

A configuration key is used to link the firebase project with the python program. This key can be retrieved from the Firebase service. A storage reference is created using which files can be uploaded or downloaded.

The images which were saved earlier by the mask and social distancing detection are uploaded to the cloud storage.

The local path and path to be uploaded to the cloud storage are passed as arguments:

```
storage.child(path_to_cloud_storage).pu
t(path_to_local_storage)
```



1. Integrate the Firebase SDKs for Cloud Storage.
Include clients via Gradle, CocoaPods, or a script include.

2. Creating a Storage Reference
Reference the path to a file, such as "images/mountains.png", to upload, download, or delete it.

3. Upload or Download
Upload or download to native types in memory or on disk.

4. Secure the Files
Use Firebase Security Rules for Cloud Storage to secure your files.

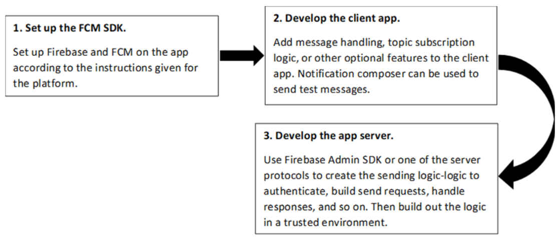
Figure 11: Implementation Path – Cloud Storage (https://firebase.google.com/docs).

Firebase was then added to the Android project by registering the app on Firebase and setting up the configuration files. A storage reference is again created, and the image is downloaded from the cloud storage. The image then gets updated to the image view. This can be saved locally to the Android device as well.

5.3 Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) provides an easy platform for sending messages. With the help of FCM, we can notify a client app via the Firebase Admin SDK or the FCM server protocols.

Using the firebase admin SDK, a notification can be pushed to the app. While setting up, the credentials



1. Set up the FCM SDK.

Set up Firebase and FCM on the app according to the instructions given for the platform.

2. Develop the client app.

Add message handling, topic subscription logic, or other optional features to the client app. Notification composer can be used to send test messages.

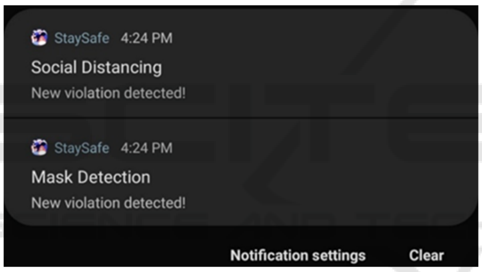
3. Develop the app server.

Use Firebase Admin SDK or one of the server protocols to create the sending logic-logic to authenticate, build send requests, handle responses, and so on. Then build out the logic in a trusted environment.

Figure 12: Implementation Path – Firebase Cloud Messaging (https://firebase.google.com/docs).

are verified, and the app is initialized through a '. json' file. A custom notification can be sent with a title, message, and topic. The topic determines which user will get the notification. The app will only get notified depending on this topic which acts as the username as each user is subscribed to a particular topic.

In the Android app, a Firebase Message handling class was added which takes care of handling the incoming messages and a notification builder for creating notifications. The app gets notified every time a violation is detected.



StaySafe 4:24 PM
Social Distancing
New violation detected!

StaySafe 4:24 PM
Mask Detection
New violation detected!

Notification settings Clear

Figure 13: Notifications from StaySafe App.

6 ANDROID APP

The app was developed in Java with the help of Android Studio which is built on JetBrains' IntelliJ IDEA software.

The main interface of the app has the recently updated image along with the date and time. Two buttons are used to switch to mask detection and social distancing detection mode. A save button is used to save the image locally. Refresh button is used for updating the image view to the recently added image on the cloud. "View history" button takes the user to a new activity for checking the history. The view for history is implemented as a RecyclerView. RecyclerView makes it easy to display large sets of data. The data is supplied along with how each item looks and the RecyclerView library dynamically creates the elements when needed.

When an item scrolls off the screen, the RecyclerView recycles the individual elements instead of destroying the view. By reusing the previous views that were scrolled off screen, it reduces power consumption and improves the app's performance and responsiveness. (Fatima, 2020)

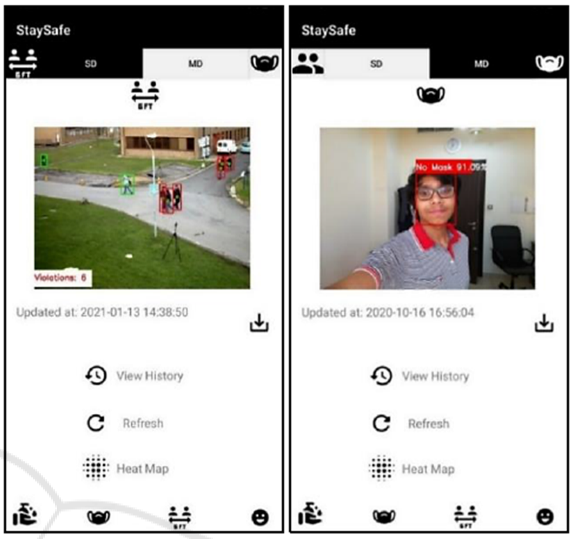


Figure 14: Main Interface.

7 APPLICATIONS

Hospitals and Clinics: This is an important domain that needs special attention during a pandemic. To safeguard the health of doctors and nurses, this system can be implemented to detect whether a patient is wearing a mask. A CCTV camera can be placed to detect mask and social distancing violations which can in turn help reduce crowding while patients are waiting in queues for their appointments.

Airports: Many of the destinations around the world have started easing travel restrictions. This system can be implemented in airports where people are waiting in queues at check-in facilities, security clearance gates, passport control, waiting lobbies etc.

Shops / Workplaces: As many businesses are opening after lockdown this system can be implemented to ensure that customers and employees are following all the COVID 19 safety protocols.

8 CONCLUSION AND FUTURE SCOPE

This project has been developed to come up with an efficient way for detecting and notifying officials when a person does not follow the COVID 19 safety protocols in a workplace, business establishments etc. In this work, we have trained a model for face mask detection using TensorFlow and Keras and used YOLO Object detection for detecting social distancing. The proposed CNN architecture comprises two convolutional layers followed by relu activation function and a max pooling layer. YOLOv3 was used to detect people in a frame and find the Euclidean distance between them. With the help of OpenCV we were able to capture the video feed from different sources like webcam, video file or an IP camera. An android app was developed which will get notified every time a violation is detected, and the detected images can also be viewed through the app. This was achieved with the help of Firebase service. As a future study, we can work on finding a pattern to detect or predict the time at which it gets crowded the most and the heat map can be plotted in a more accurate manner.

REFERENCES

Soni, A., & Singh, A. P., 2020. Automatic Motorcyclist Helmet Rule Violation Detection using Tensorflow & Keras in OpenCV. In *2020 IEEE International Students Conference on Electrical, Electronics and Computer Science (SCEECS)*.

Chen, S., Wei, Y., Xu, Z., Sun, P., & Wen, C., 2020. Design and Implementation of Second-generation ID Card Number Identification Model based on TensorFlow. In *IEEE International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA)*.

Caveness, E., C., P. S., Peng, Z., Polyzotis, N., Roy, S., & Zinkevich, M., 2020. TensorFlow Data Validation: Data Analysis and Validation in Continuous ML Pipelines. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*.

Lu, Y., Zhang, L., & Xie, W., 2020. YOLO-compact: An Efficient YOLO Network for Single Category Real-time Object Detection. In *2020 Chinese Control and Decision Conference (CCDC)*.

Ullah, M. B., 2020. CPU Based YOLO: A Real Time Object Detection Algorithm. In *2020 IEEE Region 10 Symposium (TENSYMP)*.

Raj, A., Maji, K., & Shetty, S. D., 2021. Ethereum for Internet of Things security. In *Multimedia Tools and Applications*.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A., 2016. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Natraj, L., & Shetty, S. D., 2019. A Translation System That Converts English Text to American Sign Language Enhanced with Deep Learning Modules. In *International Journal of Innovative Technology and Exploring Engineering Regular Issue*, 8(12), 5378-5383.

Khawas, C., & Shah, P., 2018. Application of Firebase in Android App Development-A Study. In *International Journal of Computer Applications*, 179(46), 49-53.

Fatima, N. S., Steffy, D., Stella, D., & Devi, S. N., 2020. Enhanced Performance of Android Application Using RecyclerView. In *Advances in Intelligent Systems and Computing Advanced Computing and Intelligent Engineering*, 189-199.

Nair, L. R., Shetty, S. D., & Shetty, S. D., 2018. Applying spark based machine learning model on streaming big data for health status prediction. In *Computers & Electrical Engineering*, 65, 393-399.

Duran-Lopez, L., Dominguez-Morales, J., Amaya-Rodriguez, I., Luna-Perejon, F., Civit-Masot, J., Vicente-Diaz, S., & Linares-Barranco, A., 2019. Breast Cancer Automatic Diagnosis System using Faster Regional Convolutional Neural Networks. In *Proceedings of the 11th International Joint Conference on Computational Intelligence*.

Sung, M., Yu, S., & Girdhar, Y., 2017. Vision based real-time fish detection using convolutional neural network. In *OCEANS 2017 - Aberdeen*.

Hansen, D. K., Nasrollahi, K., Rasmusen, C. B., & Moeslund, T. B. (2017). Real-Time Barcode Detection and Classification using Deep Learning. In *Proceedings of the 9th International Joint Conference on Computational Intelligence*.

Goodfellow, I., Bengio, Y., & Courville, A., 2016. *Deep learning*, Cambridge (EE. UU.). MIT Press.

Belciug, S., 2020. *Artificial intelligence in cancer: Diagnostic to tailored treatment*, London, United Kingdom. Academic Press.