**Implementation**

A test video of a parking lot is fragmented into frames of images to be tested by our model, Mask RCNN. As Mask RCNN uses Tensorflow version 1, due to the updated Tensorflow version 2, we had to edit the codes of the model Mask RCNN to support the newer version of Tensorflow. Now, we had to individually mark the parking spots so that we can calculate the intersection of a detected car with the parking spot to determine if that spot is occupied. To do this, we incorporated a different approach than manually calculating points and marking the spots with Matplotlib of python. We assumed and thus recorded the video so that in the first two seconds all the parking spots were occupied. The first 60 frames (two seconds) were then taken and run through OpenCV to detect cars. OpenCV detects objects by region, meaning it can detect a rectangular region within which the object is present. The rectangular regions of the cars, that were consistently detected in the first two seconds, were saved in a list variable and we will use these rectangles as parking spots for the rest of the video. Another problem that may occur during this operation is the background movement of cars falsely detected as parking spots. To ensure this does not happen, while running the loop of code for detecting parking spots, we used functions of OpenCV (cvtColor, GaussianBlur, dilate, erode, drawContours etc.) to ‘blacken out’ any pixels that would be different from other frames. This means any moving objects would get blackened and blurred so OpenCV cannot detect them as objects. This solves the false parking spot detection problem, since only the cars that are in place and not moving is detected.

Now, the loop to detect if cars are present in the parking spots is run. This time Mask RCNN is used to detect cars. Before that we had to convert the color scheme of the frames from GBR(used by OpenCV) to RGB(used by Mask RCNN). Mask RCNN can detect objects with an accurate border detection. Meaning the region detected is in the shape of the object. The intersection of this region with the saved parking spot variable (computed with mcrnn utils library function compute\_overlaps) is used to detect if the spots are empty of not. An intersection value of 0.15 is used to determine this clause.

During the computation of the detection twilio api was used to send a sms to our (a user’s) personal number when a spot was empty. Also, the color of the rectangles around the parking spots were drawn to be red if they were not empty and green if they were empty , using OpenCV rectangle function, for each and every frames.

Finally, these frames were defragmented back to a video and saved to our google drive path.

**App**

For a visual representation of our project we built a UI app using flutter that would present a different use case of our project. The app was made using standard StatelessWidget of flutter for three different screens: Home, Maps, Video. The Home Screen prompts a map icon which upon clicking would redirect to the Maps screen. To incorporate this screen Google Map API was used. The style of the Map was changed using JSON code from Google’s platform. To incorporate the UI of the Map screen, google\_map\_flutter 2.0.6 was used. A custom marker was placed with longitude, latitude values of a garage that presumably hosted out services. Upon tapping that marker an info window pops up, and upon tapping the info window, user is redirected to the CCTV footage (a video for our prototype) of the parking garage, to see if spaces are available and where. This video is imported from the video that our model generates with all the predictions. To incorporate the video player in flutter video\_player 2.1.2 package was used.

Ideally, a live video was planned to be used here that would be directly imported from a hosted server where our model would continuously send in frames as video which the hosted server would take in a buffer and send back to the app. Due to the inability to find access to a live feed of a commercial parking lot we settled with a pre-made video.

**Challenges**

The biggest challenge was to find a suitable video of a parking spot with multiple cars that were moving. Due to the lockdown, it was not possible for us to record a video of a public parking area. Upon requesting officials of popular malls in Dhaka, we were not granted access to a live feed or even a pre-recorded video of their parking lot. With so solution at hand, a team member recorded a personal video with their car in their garage that we would later use in our model.