

# Snagging Parking Spaces with Mask R-CNN and Python

**LIKHITHA SADAVALA, (B.TECH-CSE),**

SRI PADMAVATI MAHILA VISVAVIDYALAYAM, Tirupati, Andhra Pradesh, 517502, INDIA

**ABSTRACT** This model utilizes Deep Learning to detect when a parking space becomes available and straightly notify you by a text message. We first use SOTA pretrained model for detection, such as Mask R-CNN, in order to detect the static cars in the video. No training is even needed. Git clone it. We are now able to conclude from their detected bounding boxes the potential spaces for parking. We will then constantly compare between cars and parking spaces, to look for opportunities. We will do it easily by calculating the Intersection over Union (IoU) on the car's bounding box, to see if it is overlapping with a parking spot's bounding box. Finally, we will use message API like Twilio to send a real-time message to our phone when a parking spot becomes available.

**INDEX TERMS** API, Box Refinement, COCO(Common Objects In Context), Convolutional Neural Networks, Deep Learning, Fast R-CNN, HOG(Histogram of Oriented Gradients), IOU( Intersection Over Union), Keras, Mask R-CNN, Multi-GPU, Neural Networks, SOTA, Tensorflow, Twilio, YOLO

## I. INTRODUCTION

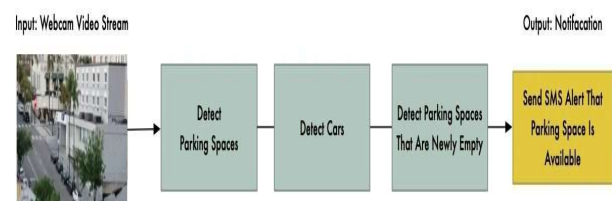
Searching for a free parking could be a night mare. They are usually taken, and once not, they quickly get stolen. The idea of this model is to locate a camera which will constantly record, and let a DL model to monitor the image and notify the person by text message instantly when a slot is released.

### A. AGENDA

1. The input to the machine learning pipeline is simply a video stream from a normal webcam pointed out the window.
2. The first step in the pipeline is to detect all possible parking spaces in a frame of video. Obviously we need to know which parts of the image are parking spaces before we can detect which parking spaces are unoccupied.
3. The second step is to detect all the cars in each frame of video. This will let us track the movement of each car from frame to frame.
4. The third step is to determine which of the parking spaces are currently occupied by cars and which aren't. This requires combining the results of the first and second steps.
5. Last step is to send a notification when a parking space becomes newly available. This will be based on changes in car positions between frames of video.

### B. PIPELINE

**DETECTING CARS** We need a model who can detect and segment cars in the image - which will be equivalent to parking spots when the cars are static.



There are old-fashioned options such as:

**HOG (Histogram of Oriented Gradients)** - object detector and slide it over our image to find all the cars. This older, non-deep-learning approach is relatively fast to run, but it won't handle cars rotated in different orientations very well.

**CNN (Convolutional Neural Network)** - object detector and slide it over our image until we find all the cars. This approach is accurate, but not that efficient since we have to scan the same image multiple times with the CNN to find all the cars throughout the image. And while it can easily find cars rotated in different orientations, it requires a lot more training data than a HOG-based object detector.

So we will go for Mask R-CNN, Faster R-CNN or YOLO!

Those combine the accuracy of CNNs with clever design and efficiency tricks that greatly speed up the detection process. This will run relatively fast (on a GPU) as long as

we have a lot of training data to train the model.

In general, we would like to choose the simplest solution that will get the job done with the least amount of training data and not assume that we need the newest, flashiest algorithm. But in this specific case, **Mask R-CNN** is a reasonable choice despite being fairly flashy and new.

The **Mask R-CNN** architecture is designed in such a way where it detects objects across the entire image in a computationally efficient manner without using a sliding window approach. In other words, it runs fairly quickly. With a modern GPU, we should be able to detect objects in high-res videos at several frames a second. That should be fine for this project.

In addition, **Mask R-CNN** gives us lots of information about each detected object. Most object detection algorithms only return the bounding box of each object. But Mask R-CNN will not only give us the location of each object, but it will also give us an object outline (or mask), like this:



### C. TRAINING MASK R-CNN

To train Mask R-CNN, we need lots of pictures of the kinds of objects that we want to detect. We could go outside and take pictures of cars and trace out all the cars in those images, but that would take a couple days of work. Luckily, cars are common objects that lots of people want to detect, so several public datasets of car images already exist.

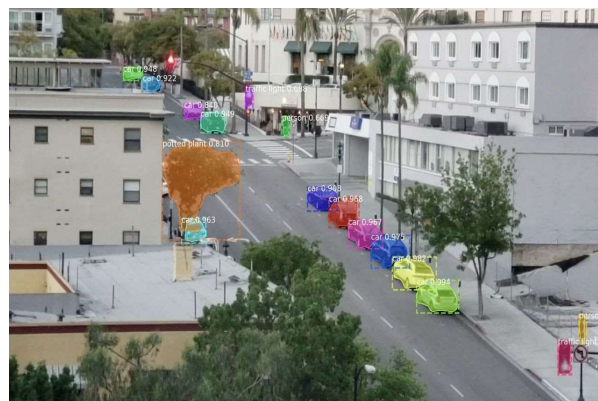
There's a popular dataset called COCO (short for Common Objects In Context) that has images annotated with object masks. In this dataset, there are over 12,000 images with cars already outlined.

But wait, things get even better! Since it's so common to want to build object detection models using the COCO dataset, lots of people have already done it and shared their results. So instead of training our own model, we can start with a pre-trained model that can detect cars out of the box.

For this project, we'll use the great open source Mask R-CNN implementation from Matterport which comes with a pre-trained model.

Side note: Don't be afraid of training a custom Mask R-CNN object detector! It's time consuming to annotate the data, but it is not that difficult. If you want to walk through training a custom Mask R-CNN model using your own data.

If we run the pre-trained model on our camera image, this is what is detected out of the box:



For each object detected in the image, we get back four things from the Mask R-CNN model:

- The type of object that was detected (as an integer). The pre-trained COCO model knows how to detect 80 different common objects like cars and trucks.
- A confidence score of the object detection. The higher the number, the more certain the model is that it correctly identified the object.
- The bounding box of the object in the image, given as X/Y pixel locations.
- A bitmap "mask" that tells which pixels within the bounding box are part of the object and which aren't. With the mask data, we can also work out the outline of the object.

So we basically know how to detect cars now,

We can move to our next target => detecting empty slots!

#### 1) DETECTING EMPTY PARKING SPACES

We actually know the pixel location of each car in our image. And by looking at multiple frames of video in succession, we can easily work out which cars haven't moved and assume those areas are parking spaces. But how do we detect when a car leaves a parking space?

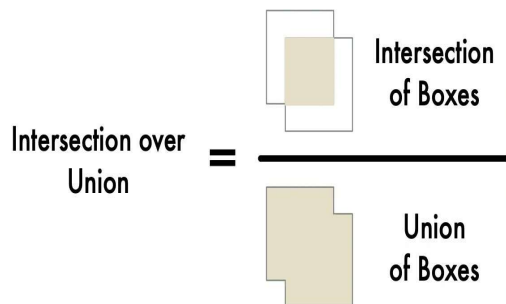
The problem is that the bounding boxes of the cars in our image partially overlap.

So if we assume that each of those bounding boxes represents a parking space, it's possible that the box can be

partially occupied by a car even when the space is empty. We need a way to measure how much two objects overlap so we can check for “mostly empty” boxes.

The measure we will use is called Intersection Over Union or IoU.

IoU calculated by finding the amount of pixels where two objects overlap and dividing it by the amount of pixels covered by both objects, like this:

$$\text{Intersection over Union} = \frac{\text{Intersection of Boxes}}{\text{Union of Boxes}}$$


This will give us a measure of how much a car’s bounding box is overlapping a parking spot’s bounding box. With this, we can easily work out if a car is in a parking space or not. If the IoU measure is low, like 0.15, that means the car isn’t really occupying much of the parking space. But if the measure is high, like 0.6, that means the car is occupying the majority of the parking space area so we can be sure that the space is occupied.

Even though Mask R-CNN is highly accurate, occasionally it can miss a car or two in a single frame of video. So before flagging a parking space as free, we should make sure it remains free for a little while — maybe 5 or 10 sequential frames of video. That will prevent the system from incorrectly detecting open parking spaces just because object detection had a temporary hiccup on one frame of video. But as soon as we see that we have at least one parking space free for several sequentially frames of video, we are ready to send a text!

#### D. SENDING TEXT NOTIFICATION

The last step of our pipeline is to send an SMS alert when we notice that a parking space has been free for several frames of video.

Sending an SMS message from Python is very simple using Twilio. Twilio is a popular API that lets you send an SMS message from basically any programming language with a few lines of code. Of course, if you prefer to use a different service SMS provider, you could use that instead.

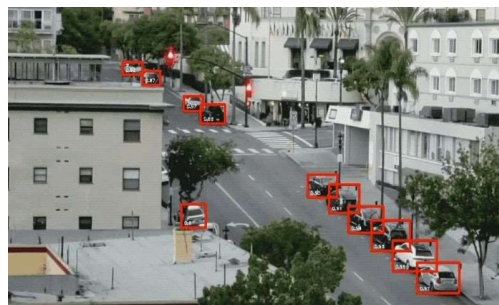
To use Twilio, sign up for a trial account, create a Twilio phone number and get your account credentials. Then, you need to install the Twilio Python client library.

#### E. NOTE

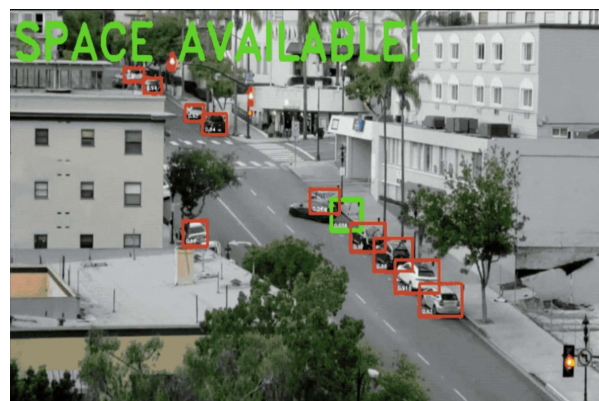
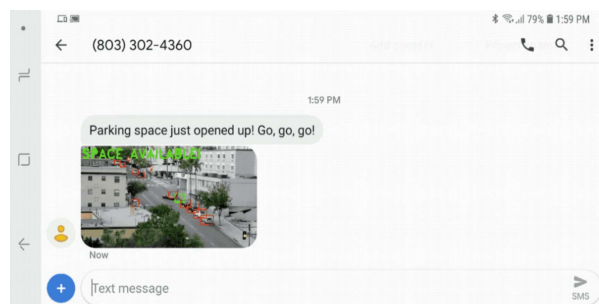
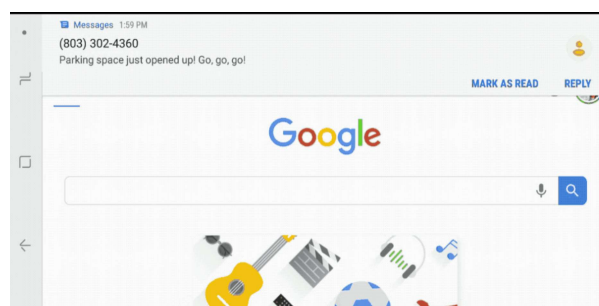
This model requires TensorFlow version 1.15.3 and Keras 2.2.4.

## II. RESULTS

### A. LIVE CAMERA DEMO:



### B. SAMPLE SMS:



## III. CONCLUSION

Mask R-CNN with Python model can be used for detecting parking lots in public places and it saves time by just sending an SMS to the person’s phone. Mask R-CNN can also be trained for identifying the masks, animals etc.

---

#### IV. ACKNOWLEDGEMENT

Original Author: Adam Geitgey

<https://gist.github.com/ageitgey>

#### REFERENCES

- [1] Neural Networks and Deep Learning: A Textbook Textbook by Charu C. Aggarwal
- [2] An Introduction to Neural Networks Kevin Gurney