# Unsupervised Learning for Parking Detection

Kushajveer Singh
https://medium.com/@kushajreal
https://www.linkedin.com/in/kushaj/
https://github.com/KushajveerSingh

Team Name:- **CG_TEAM_JPYFW**

Member:- **Kushajveer Singh**

# Jio Artificial Intelligence Hackathon

Smart Parking for Smart Cities of the future

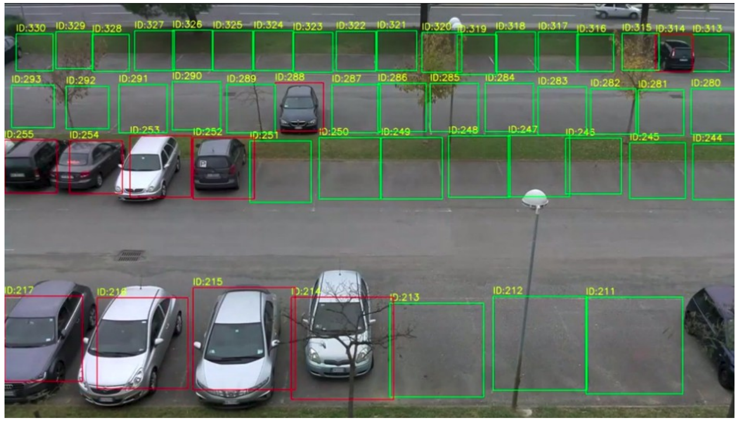
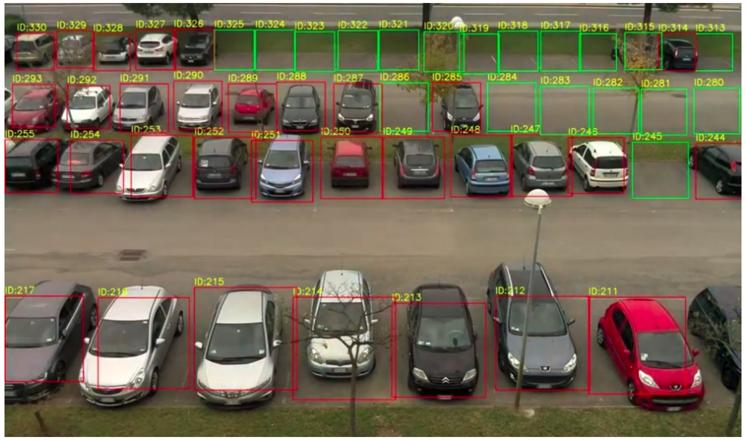
Where should I park my car?

## Parking Lot Detection

- The main aim of Parking Lot Detection is to give users information about whether there is free parking space in a parking lot or not and in case there is free space where is it.
- In order to achieve this we use the video feed from cameras which looks like the figure on right.
- **Why is it important?**
- It saves unnecessary time spent to find free parking space.
- More importantly it has the potential to save large volumes of fuel which is very important for the environment in the long run.
- This technology permits the cities to carefully manage their parking supply.

## How to solve it

- We want to make a model that can understand which parking spaces are empty and which are occupied as shown below.

- But first our model must know where the parking slots are.

- **Note:-** Directly detecting cars in an image does not work, as there may be some cars that are travelling on the road. Motion tracking can also be used to solve it but it is expensive at inference time.

## How to detect parking spaces?

There are two ways to solve it, with their own pros and cons

1) Use human annotators to store all the parking spaces
   - It will be the most accurate model as it will not miss any parking space
   - But it is expensive, even a slight change in camera would mess up the parking spaces.
   - Difficult to setup for different parking lot spaces

2) Learn parking spaces automatically
   - Can adapt to different parking spaces easily
   - But it comes at the expense of missing some parking spaces or inaccurate detections

Most of the research thus focuses on option 2, as it easy and cheap to deploy. We can easily get new prediction space in case our camera get’s rotated.

# Learning Parking Space Locations

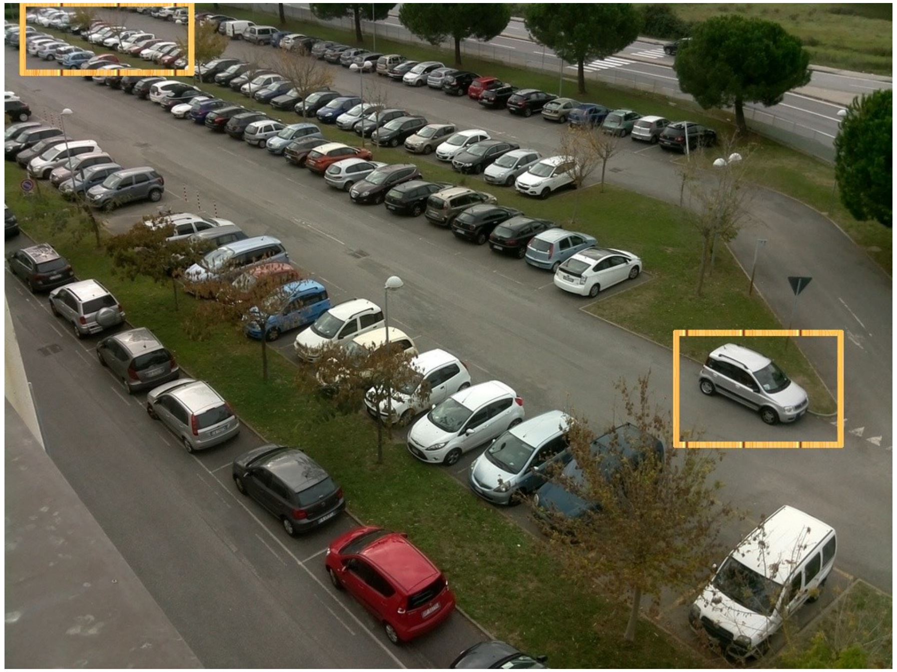
- We now reduced the complexity of the task to two simple and manageable tasks, first detecting parking spaces and then classifying if that parking space is occupied or not.

- Most of the current research has focused on using lane markings and doing some feature engineering to detect parking spaces.

- **But this approach cannot generalize well**

- For cases where we do not have any lane markings but are standard parking spaces, like the one in bottom-right of figure

- For cases where the lane markings are present but are not visible or are obstructed, like the on in top-left of figure



## General overview of my approach

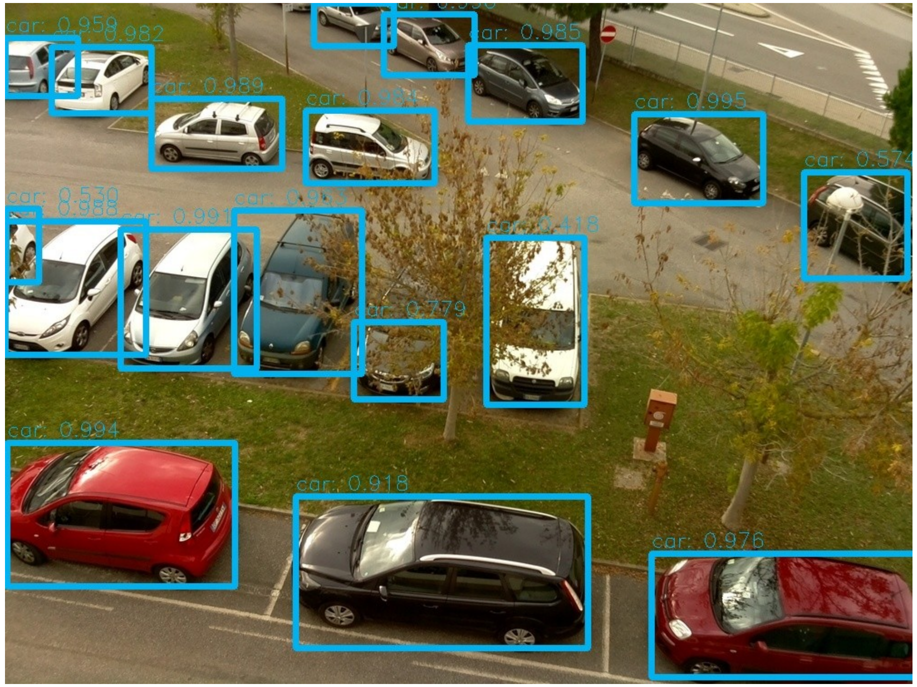
Get parking spaces.
I propose a unsupervised learning approach that does not need any feature engineering

Mask out parts of image containing parking spaces

Use an image classifier to classify the patches as occupied or not

## Unsupervised Learning

- The solution I propose is to learn the location of parking spaces from the images of the parking lot itself.
- How this is beneficial?
- You no longer have to do any feature engineering and you don't have to worry about lane markings.
- **Major benefit from this approch is we can use any pretrained object detection models without even needing to finetune them.**
- The example I show on right, is an example M2Det[1] object detection model trained on COCO[2] dataset, and I haven't done any fine-tuning yet.
- Now there are cases where some cars are not recognized in an image, for that I have made a script that combine the predictions from multiple images.



[1] M2Det: A Single-Shot Object Detector based on Multi-Level Feature Pyramid Network, *arxiv.org 1811.04533*
[2] Microsoft COCO: Common Objects in Context, *arxiv.org 1405.0312*

## My Solution Summary

- Use an unsupervised approach to detect parking space, using a pretrained object-detection model (for establishing the generaliation power of pretrained models I will refrain from fine-tuning and it also show’s that it is easy to use new models without by just plugging them in).

- When using a single camera we are limited to a limited field of view in which we can easily detect cars, but for cases where the cars are overlapping it becomes increasingly difficult to tell if the space is occupied or not due to overlapping of cars.

- A method of combining the locations of parking spaces from multiple images. In some cases detecting cars in a single image becomes difficult due to the limitations in the CNN architectures, thus using multiple images can be useful in this case.

- For inference a Resnet50 model is trained which can classify the parking spaces as filled or not. For every image, all the parking spaces are places in a single batch so that our model can give proedictions for an image in a single step.

## Modular Approach

My approach can be divided into three parts/modules

1) Object Detection Module :- It is responsible for getting the locations of the parking spaces. I use M2Det COCO 2017 pretrained model. Any object detection model can be used. Using another model is as simple as cloning the source code of the model and running the inference script.

2) Label Processing Module :- It is responsible for merging labels/bounding boxes from multiple images and also removing detections that overlap beyond a threshold. To make the implementation easier the 2D (x,y) coordinates are converted to 1D coordinates in the source code.

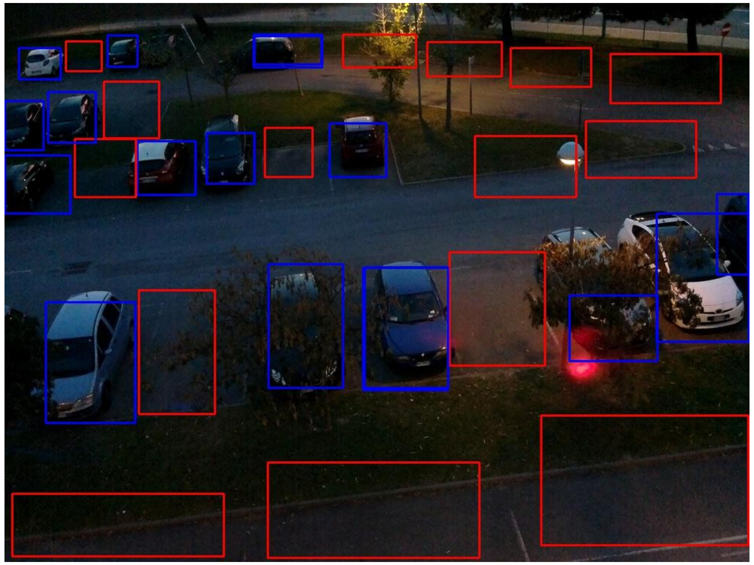
3) Classification Module :- It is responsible for classiying the patches/parking spaces in an image as occupied or not. ResNet50 pretrained on Imagenet is used and further task specific training has been done using **fastai**. This module is also extendable. Any classification model can be used. To do so only a function needs to be provided which returns the desired model as shown below.

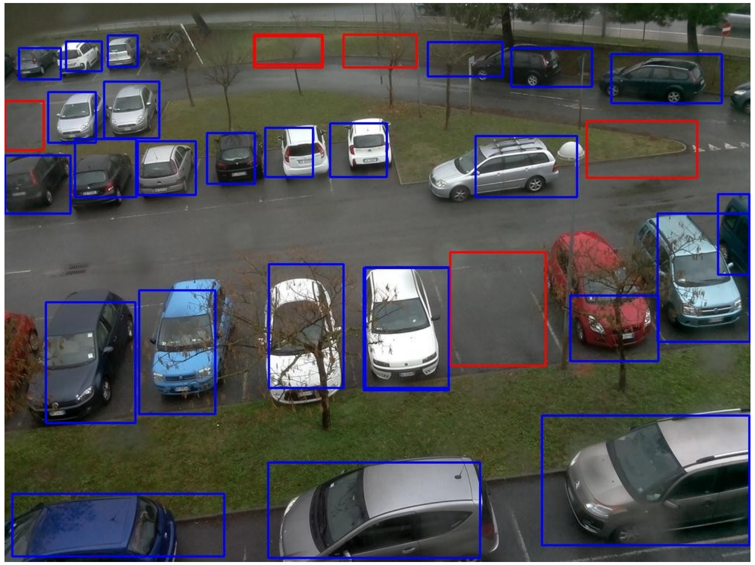
```
def load_model():
    model = resnet50()
    model.load_state_dict(torch.load('src/scripts/f_classifier.pth'))
    model.eval()
    return model
```

## Demo

- The complete instructions on how to run demo on your own images are given in DEMO.md.
- Here I show the demo for one of the parking spaces.
- Step 1. Detect parking space. To do so I use the following 2 images. You can use any number of images to get the parking spaces. After this step we would process the labels and then we are ready to test it.

- Test the model. **For more results see the `Data` folder.**





- You can see in the right image, for the parking space at top-right a car has taken sapce of 2 cars and the model shows both the spaces as occupied. These parking spaces had no marking, the only way of learning these was to infer them from the parking images.

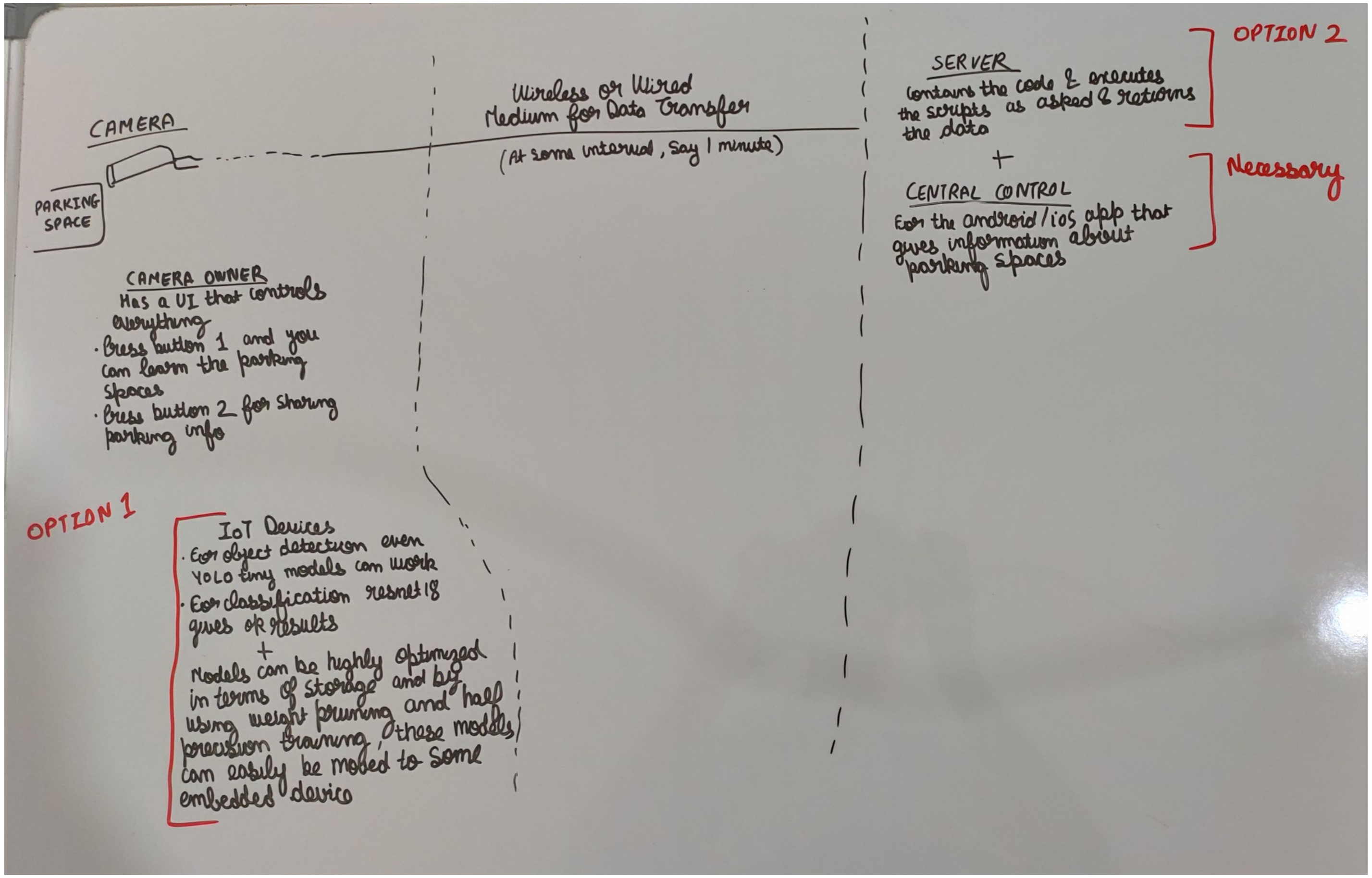
- The classifier model has been trained for night time, sunny day, rainy day, shadowed images, occluded objects.

## Technology Stack

- Python 3.6+
- PyTorch 1.0+
- OpenCV :- For various image utilities like loading, saving and drawing the bounding box predictions and for applying transforms to the images.
- Fastai 1.0+ :- Only for training the classifier model. After training, the model was converted to PyTorch format so as remove the fastai depecdency.

- For POC, I present a semantic on the next slide. It is a rough semantic showing two possible ways of using this approach.
- The approach that I present here can be treated as a general framework. In case we want to deply everything in a central server we can have big and more accurate models, but is we want to deploy this technology in an IoT environment we can use much smaller models that can do the task for us.

# Proof of Concept


CAMERA
PARKING SPACE
Wireless or Wired Medium for Data Transfer
(At some interval, say 1 minute)

CAMERA OWNER
Has a UI that controls everything
- Press button 1 and you can learn the parking spaces
- Press button 2 for sharing parking info

OPTION 1

IoT Devices
- For object detection even YOLO tiny models can work
- For classification resnet18 gives ok results

+

Models can be highly optimized in terms of storage and by using weight pruning and half precision training, these models can easily be moved to some embedded device

SERVER
Contains the code & executes the scripts as asked & returns the data

+

CENTRAL CONTROL
For the android/iOS app that gives information about parking spaces

OPTION 2

Necessary

# Why I should be selected for the finals

- I am acquainted with all the latest developments in computer vision. Making contributions to the **fastai** library and having been selected for the international fellowship for part2 of the fastai course, I have learnt all the tricks to train neural networks and get state of the art resutls. (https://docs.fast.ai/)
- I have thoroughly read Leslie N. Smith readings and practiced the way of training neural networks over the course of months such that it has become a habit. Leslie N. Smith introduced the concept of Superconvergence and cyclic learning which is the fastest and more accurate way to train neural networks.
- I am continously pushing my boundaries by expanding my research works in the field of Computer vision and at the same time practicing Natural Language Processing.
- I have  worked in the space of GANs, semantic segmentation, image translation where I implemented state of the art model SPADE by Nvidia for Image Translation. Style Transfer is another field where expadning the work of High-Resolution Networks, I implemented a Photorealistic Style Transfer model and I am currently working on combining results from multiple sources in Style Transfer so as to produce the best model for style transfer.
- I have started working on research related to the field of autonomous vehicles, thereby expanding my computer vision knowledge to 3D world and much more. I have a lot of passion about research in Computer Vision and can work on a lot of different problems in the field and also in Natural Language Processing.

- To clarify on the above you can check my medium blog https://medium.com/@kushajreal where I write about most of my projects and also can check my github for the code base.

# Associated Attachments

- **README.md** :- Gives overview of the repository and gives instructions on how to setup up the environment
- **DEMO.md** :- Contains complete instructions on how to use this source code for your images and explanation of directory structure used in the project.
- **Data** Folder :- For reference I have added results for several parking spaces in the Data folder.
- **Src** Folder :- Contains 3 subfolder
  - m2det :- source code of object detection module. Several redundant code has been removed from the official code release for the model and the model has been converted to PyTorch1.0.
  - scripts :- All the python scripts used in this project mainly *process_labels.py* and *classify_patches.py*
  - notebooks :- Contains all the development notebooks that I used when making this project

## Final Summary

- I provide a modular approach where you can just swap out modules depending on your need.
- Provide a method to split image into a grid and then combine all the results together into a single image.
- A method for combining predictions from multiple images is also provided.
- A highly accruate Resnet classifier which can classify parking spaces in the dark, rainy day, sunny day.

- The source code that does this
  - src/m2det/parking_detection_split.py
  - src/scripts/split_images.py
  - src/scripts/process_labels.py
  - src/scripts/classify_patches.py
- The source code for cases when you don’t want to split images
  - src/m2det/parking_detection.py
  - src/scripts/process_labels2.py

# Thank You

## Challenges faced

- Most of the research in this field focused around using feature engineering by using some edge detection techniques. After some time I found that these techniques do not generalize well in cases where we do not have any reference on where the parking spaces are located. So it lead to testing of various techniques one of which I have presented in this presentation.

- I could not finetune my object detection models due to hardware limitations so I came up with the strategy to combine predictions from multiple images and to split the image into smaller parts. This strategy overcame all the limitations of the pretrained object detection models and as a result of this you can easily use any pretrained object detection model without needing to do any finetuning.

- The procedure for finding on how to combine the predictions was a bit tricky and initially I thought about using IOU only, but after seeing many resutls, I came to realize that IOU alone cannot solve it. So then I added another step to clean the labels by comparing ratio of the overlapping regions.

## Possible Improvements

- Optimizations for the classification model to bring their size down using techniques like weight pruning or using half precision models to reduce the size of the models by 70%.
- The deployment side of the project is not known whether it would be a central server or a seperate IoT device for each parking space. If a central server has to used various CUDA optimizations can be done so as to increase the inference speed.
- My current implementation does inference on only one image at a time. But depending on where this project is to be deployed, we can do inference on multiple images at the same time.
- The performance of object detection models can be improved by finetuning them if needed.
- Semantic segmentation can also be used to improves the results of the parking space detection. I initially tested this approach but seeing the competition deadlines and the work that had to be done to come up my current solution I was not able to fully research it.
- My current model suffers from one drawback, it is not able to detect small objects as it was never trained to do that. In such a case it groups them all up and makes it a single car. Fine-tuning can solve it, but splitting the image as I show also works good.