Deep Learning

# Object detection
# Self Driving Cars

Group 6
Ayushi - Preetham - Ana

# Table of contents

# 01

# The project

# Objective

Develop a computer vision model capable of identifying different elements found on the streets so that cars are able to avoid these obstacles on the road

# About the project

We want to implement the last YOLO model for object detection, and also see how its results compare with Faster R-CNN when performing the same task

# 02

# Data

# Source

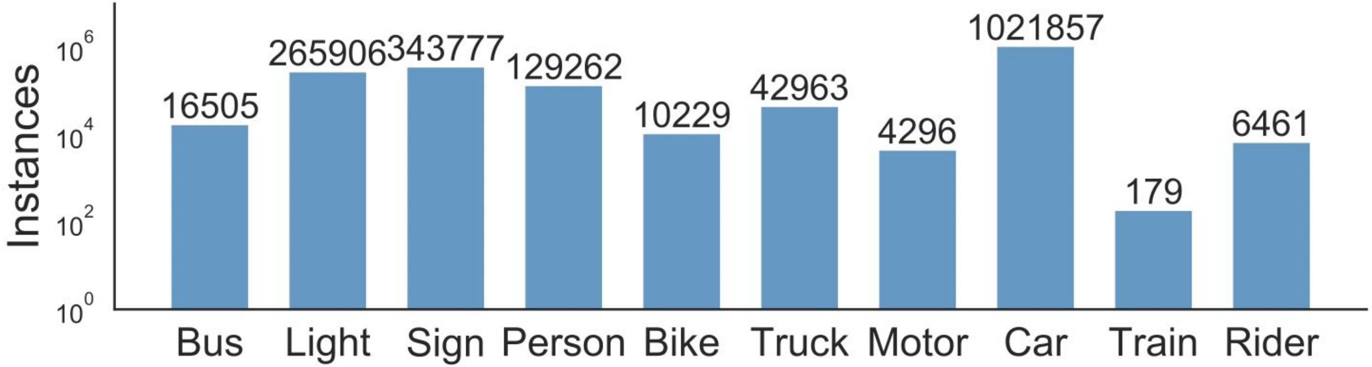
## BDD100K: A Large-scale Diverse Driving Video Database


Is the largest and most diverse open driving video dataset for computer vision research:

- 100,000 images of roads
- 10 classes for object detection
- Multiple locations in the United States
- Weather conditions: sunny, overcast and rainy
- Times of the day: daytime and nighttime

# Source

## BDD100K: A Large-scale Diverse Driving Video Database

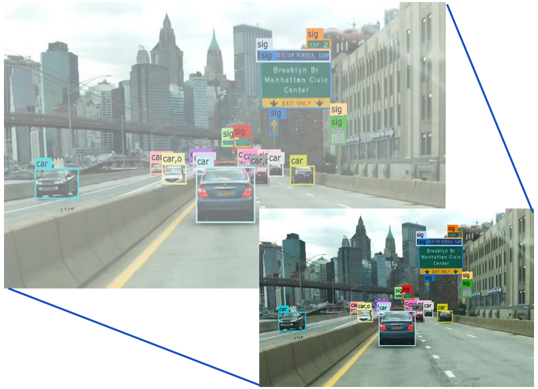



*Statistics of different types of objects.*

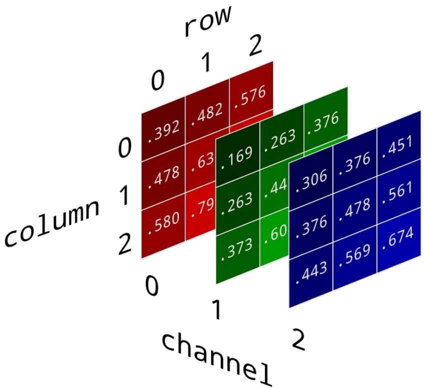
# 03

# Models

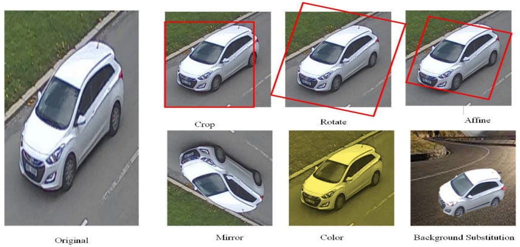
# Data Preprocessing



Resize images from 1280 × 720 to 384 × 216 pixels



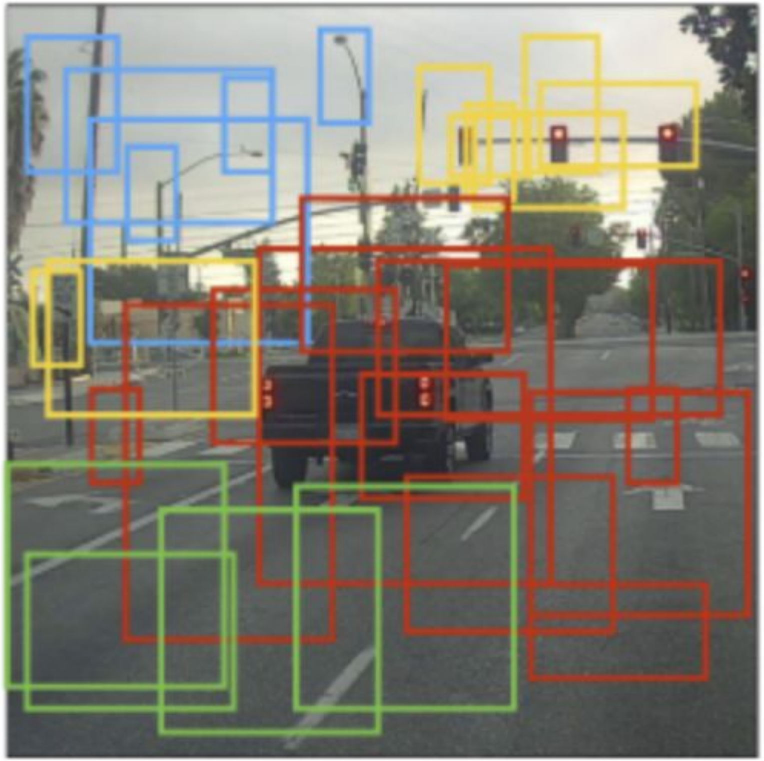

Convert images to pixel arrays




Data augmentation

# YOLO Technique





# Data Conversion

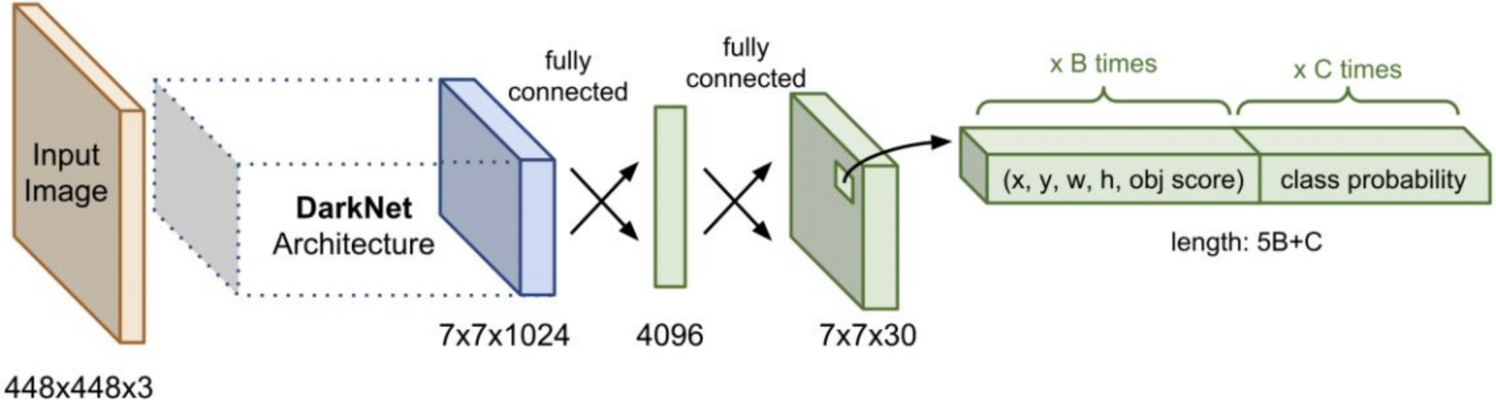

```
{
    "name": "0000f77c-6257be58.jpg",
    "attributes": {
        "weather": "clear",
        "scene": "city street",
        "timeofday": "daytime"
    },
    "timestamp": 10000,
    "labels": [
        {
            "category": "traffic light",
            "attributes": {
                "occluded": false,
                "truncated": false,
                "trafficLightColor": "green"
            },
            "manualShape": true,
            "manualAttributes": true,
            "box2d": {
                "x1": 1125.902264,
                "y1": 133.184488,
                "x2": 1156.978645,
                "y2": 210.875445
            },
            "id": 0
        },
        {
            "category": "traffic light",
            "attributes": {
                "occluded": false,
                "truncated": false,
                "trafficLightColor": "green"
            },
            "manualShape": true,
            "manualAttributes": true,
            "box2d": {
                "x1": 1156.978645,
                "y1": 136.637417,
                "x2": 1191.50796,
                "y2": 210.875443
            },
            "id": 1
```



```
11 0.174 0.154 0.018 0.014
11 0.193 0.144 0.014 0.014
10 0.181 0.102 0.014 0.052
10 0.893 0.078 0.021 0.076
10 0.952 0.075 0.02 0.08
7 0.92 0.076 0.02 0.08
7 0.692 0.313 0.007 0.019
7 0.171 0.225 0.019 0.028
10 0.193 0.227 0.011 0.024
2 0.359 0.379 0.09 0.071
2 0.496 0.377 0.053 0.068
2 0.666 0.376 0.028 0.038
2 0.683 0.374 0.013 0.028
2 0.572 0.367 0.019 0.019
2 0.587 0.363 0.008 0.019
4 0.529 0.354 0.036 0.059
4 0.636 0.362 0.028 0.052
```

# Architecture

## Model



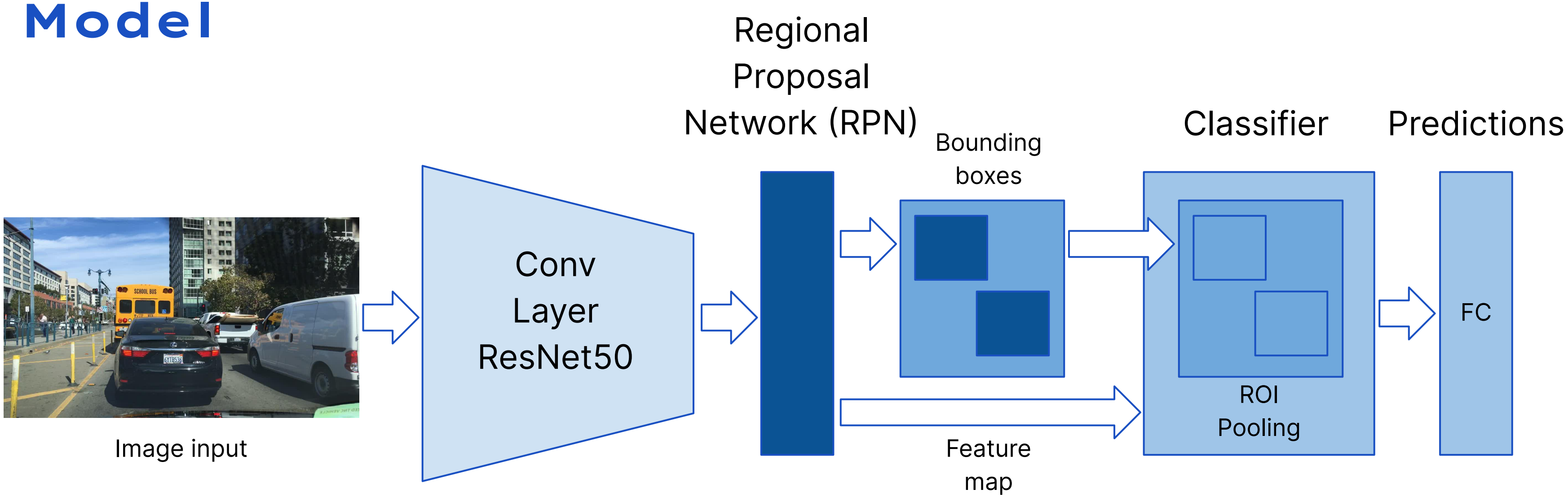
# YOLOv5s

## Implementation

- YOLO v5 pre-trained model in COCO database from Pytorch
- Trainable backbone layers = 3
- Batch size = 16
- Epochs = 50
- Image pixels = 640
- Learning rate = 0.01
- Training set = 20,000 images
- Testing set = 2,000 images

# Faster R-CNN

## Model



# Faster R-CNN

## Implementation

- Faster R-CNN with ResNet-50-FPN backbone pre-trained model in COCO database from Pytorch
- Trainable backbone layers = 3
- Batch size = 16
- Epochs = 15
- Learning rate = 0.01
- Training set = 11,000 images
- Testing set = 1,000 images


# 04

# Results

# Results

## Overall

| | maP | maR | Runtime | GPU |
|---|---|---|---|---|
| YOLO v5 | 0.44 | 0.38 | 4hrs | K80 |
| Faster R-CNN | 0.51 | 0.67 | 7 hrs | P100 |

- At the moment, Faster RCNN is giving better results than YOLO. One of the reasons may be because the YOLO model is running with low resolution images.
- There is a great difference in computation time between models. Where YOLO take almost half of the time, with double the data and worse GPU.

# Analysis

## Metrics

The maP and maR obtained are a little below when comparing with other references from the same models (~0.7). The database has complex images, with small objects or very dark light, which makes difficult to train the model and get good results

## Computation

Computation is a big limitation for computer vision models, since a high-capacity machine is needed to be able to process the images that become large tensors. For now we used a P100 GPU from Colab Pro, but we can find better options like AWS

## Improvement

The model can still be improved in many ways. Increasing number of underrepresented classes, using more images for training, bigger image size or increasing the number of conv layers


# Thanks

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, infographics & images by **Freepik**