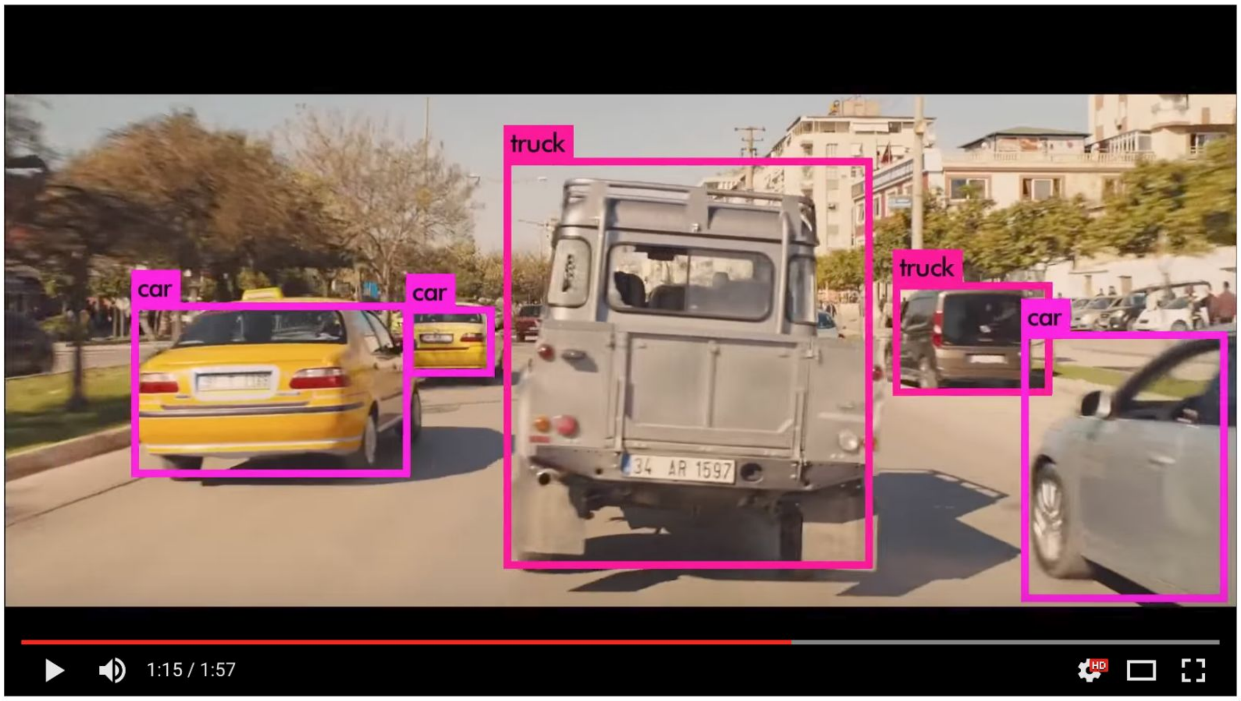
Vehicle Detection Project

Introduction

I had gone through the class lectures, and in the search for the best strategy to implement vehicle detection and tracking, stumbled upon an algorithm called You Only Look Once (YOLO). And after watching Joseph Redmon's YOLO v2 video, I had to try it out myself.

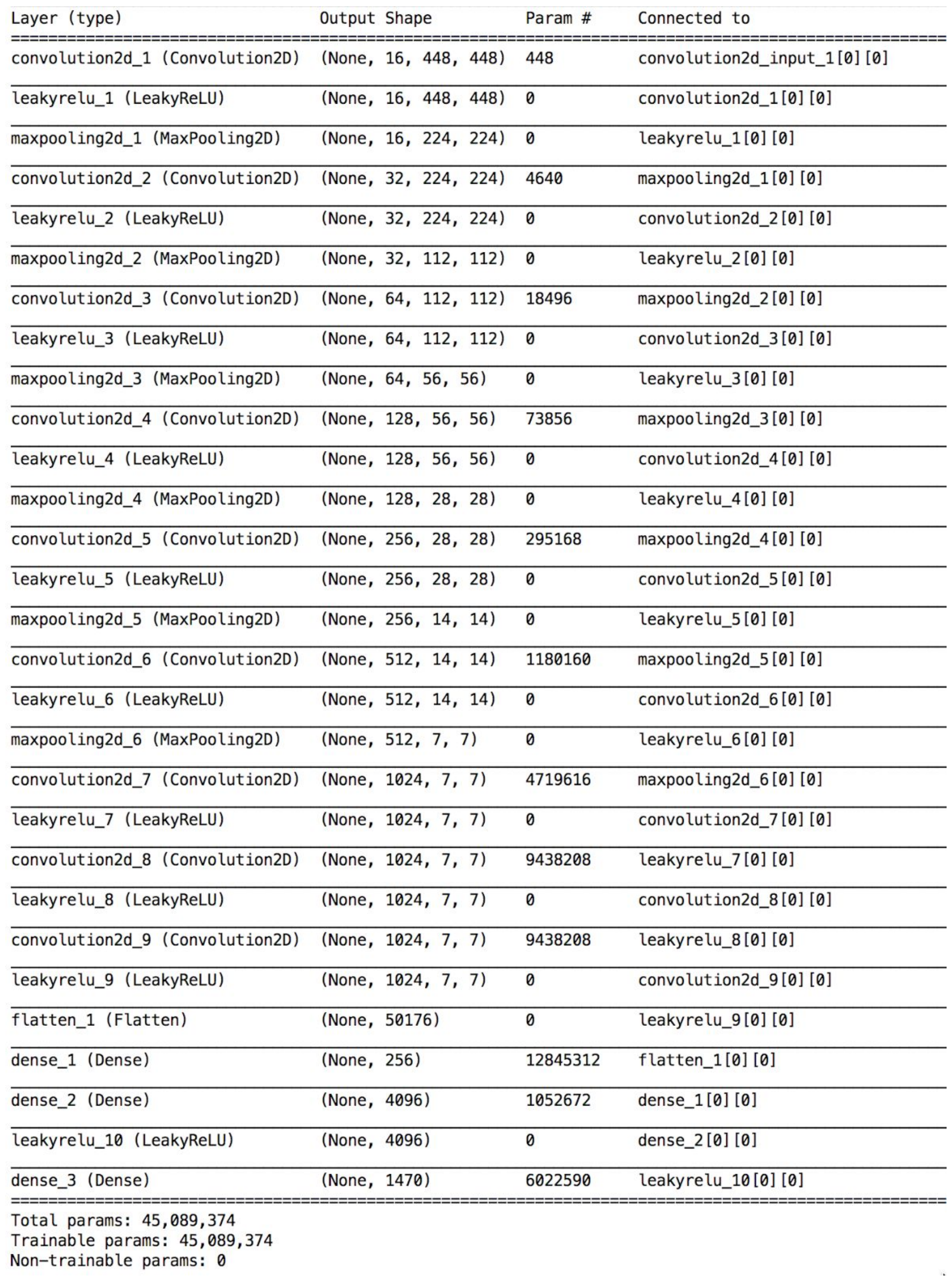


The implementation strategy shown in class uses Histogram of Oriented Gradient (HOG) with Support Vector Machine (SVM), which requires a lot of code, and would take a long time tuning parameters for gradient filters and color thresholds. This method is relatively rigid and would only work for new data that matches the trained data fairly closely.

In the real world, when we drive we scan the road for all objects that are in front of us. Be it any kind of motor vehicles, pedestrians, bicyclists, any animals, or any other objects that might interfere with our path. This calls for an object detection system that will scan the incoming image once, then identify all objects that are known, infer the size, position, and velocity of the objects that are not immediately classifiable, and extrapolate any useful information from the context of objects. The implementation i ended up using, Tiny YOLO, is a lightweight neural network that is a good start for this.

Model architecture

The model uses 6 groups of triple layers, made of convolutional 2D, leaky ReLU, and max pooling. It's then followed by 3 groups of convolutional 2D and leaky ReLU, and finished off by 3 fully connected dense layers. The model summary is as follows:



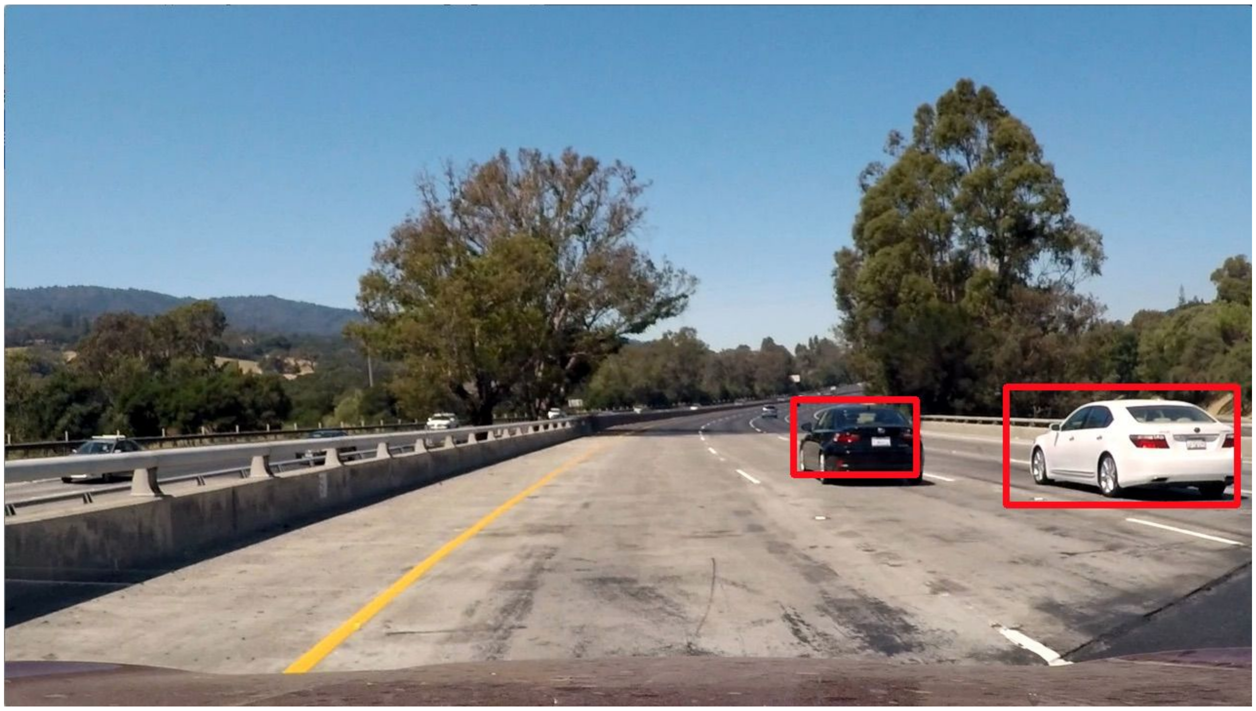
```
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
convolution2d_1 (Convolution2D)  (None, 16, 448, 448)  448         convolution2d_input_1[0][0]
____________________________________________________________________________________________________
leakyrelu_1 (LeakyReLU)          (None, 16, 448, 448)  0           convolution2d_1[0][0]
____________________________________________________________________________________________________
maxpooling2d_1 (MaxPooling2D)    (None, 16, 224, 224)  0           leakyrelu_1[0][0]
____________________________________________________________________________________________________
convolution2d_2 (Convolution2D)  (None, 32, 224, 224)  4640        maxpooling2d_1[0][0]
____________________________________________________________________________________________________
leakyrelu_2 (LeakyReLU)          (None, 32, 224, 224)  0           convolution2d_2[0][0]
____________________________________________________________________________________________________
maxpooling2d_2 (MaxPooling2D)    (None, 32, 112, 112)  0           leakyrelu_2[0][0]
____________________________________________________________________________________________________
convolution2d_3 (Convolution2D)  (None, 64, 112, 112)  18496       maxpooling2d_2[0][0]
____________________________________________________________________________________________________
leakyrelu_3 (LeakyReLU)          (None, 64, 112, 112)  0           convolution2d_3[0][0]
____________________________________________________________________________________________________
maxpooling2d_3 (MaxPooling2D)    (None, 64, 56, 56)    0           leakyrelu_3[0][0]
____________________________________________________________________________________________________
convolution2d_4 (Convolution2D)  (None, 128, 56, 56)   73856       maxpooling2d_3[0][0]
____________________________________________________________________________________________________
leakyrelu_4 (LeakyReLU)          (None, 128, 56, 56)   0           convolution2d_4[0][0]
____________________________________________________________________________________________________
maxpooling2d_4 (MaxPooling2D)    (None, 128, 28, 28)   0           leakyrelu_4[0][0]
____________________________________________________________________________________________________
convolution2d_5 (Convolution2D)  (None, 256, 28, 28)   295168      maxpooling2d_4[0][0]
____________________________________________________________________________________________________
leakyrelu_5 (LeakyReLU)          (None, 256, 28, 28)   0           convolution2d_5[0][0]
____________________________________________________________________________________________________
maxpooling2d_5 (MaxPooling2D)    (None, 256, 14, 14)   0           leakyrelu_5[0][0]
____________________________________________________________________________________________________
convolution2d_6 (Convolution2D)  (None, 512, 14, 14)   1180160     maxpooling2d_5[0][0]
____________________________________________________________________________________________________
leakyrelu_6 (LeakyReLU)          (None, 512, 14, 14)   0           convolution2d_6[0][0]
____________________________________________________________________________________________________
maxpooling2d_6 (MaxPooling2D)    (None, 512, 7, 7)     0           leakyrelu_6[0][0]
____________________________________________________________________________________________________
convolution2d_7 (Convolution2D)  (None, 1024, 7, 7)    4719616     maxpooling2d_6[0][0]
____________________________________________________________________________________________________
leakyrelu_7 (LeakyReLU)          (None, 1024, 7, 7)    0           convolution2d_7[0][0]
____________________________________________________________________________________________________
convolution2d_8 (Convolution2D)  (None, 1024, 7, 7)    9438208     leakyrelu_7[0][0]
____________________________________________________________________________________________________
leakyrelu_8 (LeakyReLU)          (None, 1024, 7, 7)    0           convolution2d_8[0][0]
____________________________________________________________________________________________________
convolution2d_9 (Convolution2D)  (None, 1024, 7, 7)    9438208     leakyrelu_8[0][0]
____________________________________________________________________________________________________
leakyrelu_9 (LeakyReLU)          (None, 1024, 7, 7)    0           convolution2d_9[0][0]
____________________________________________________________________________________________________
flatten_1 (Flatten)              (None, 50176)         0           leakyrelu_9[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 256)           12845312    flatten_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 4096)          1052672     dense_1[0][0]
____________________________________________________________________________________________________
leakyrelu_10 (LeakyReLU)         (None, 4096)          0           dense_2[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 1470)          6022590     leakyrelu_10[0][0]
====================================================================================================
Total params: 45,089,374
Trainable params: 45,089,374
Non-trainable params: 0
```

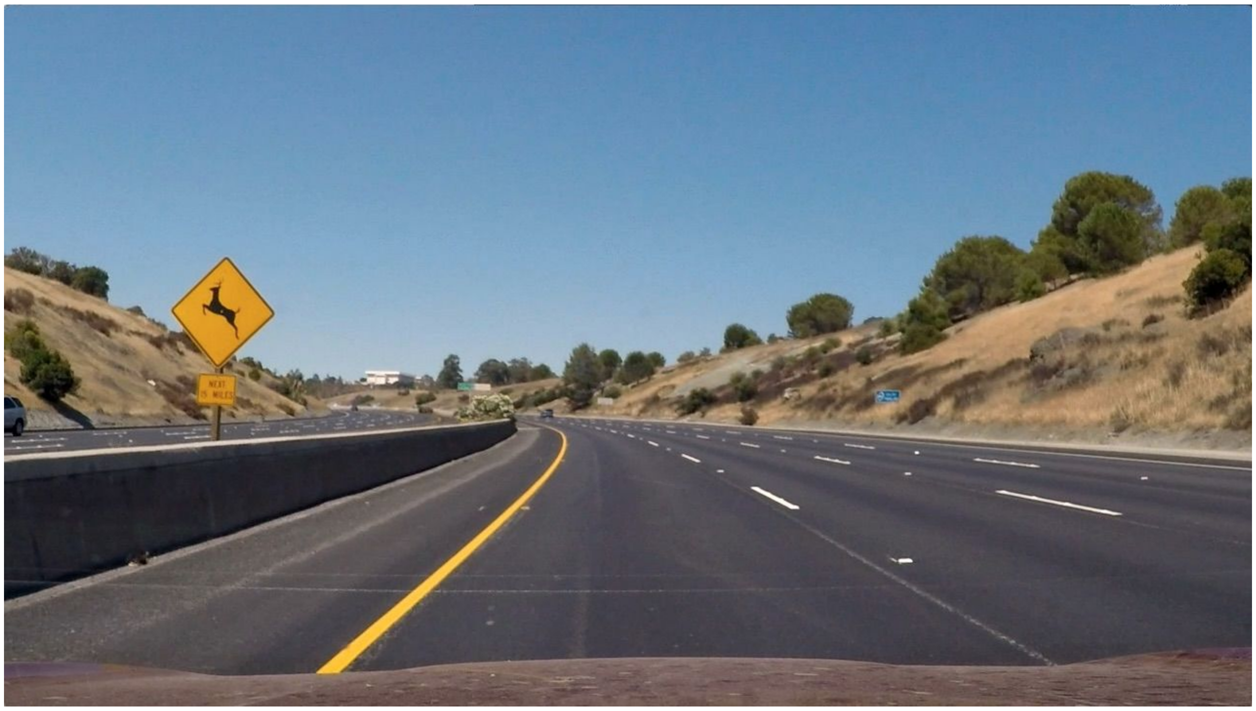
Pipeline structure

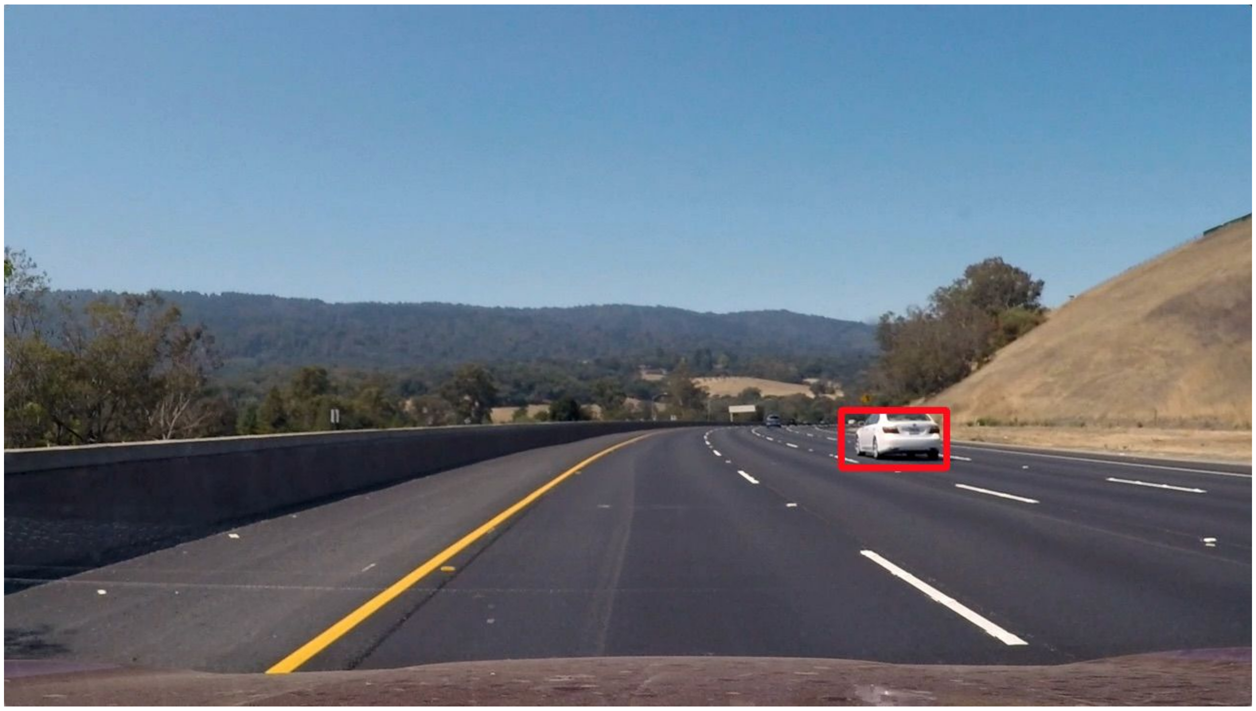
The pipelines for image and video are the same except one takes in image files and the other takes in video clips. The pipeline structure is as follows:

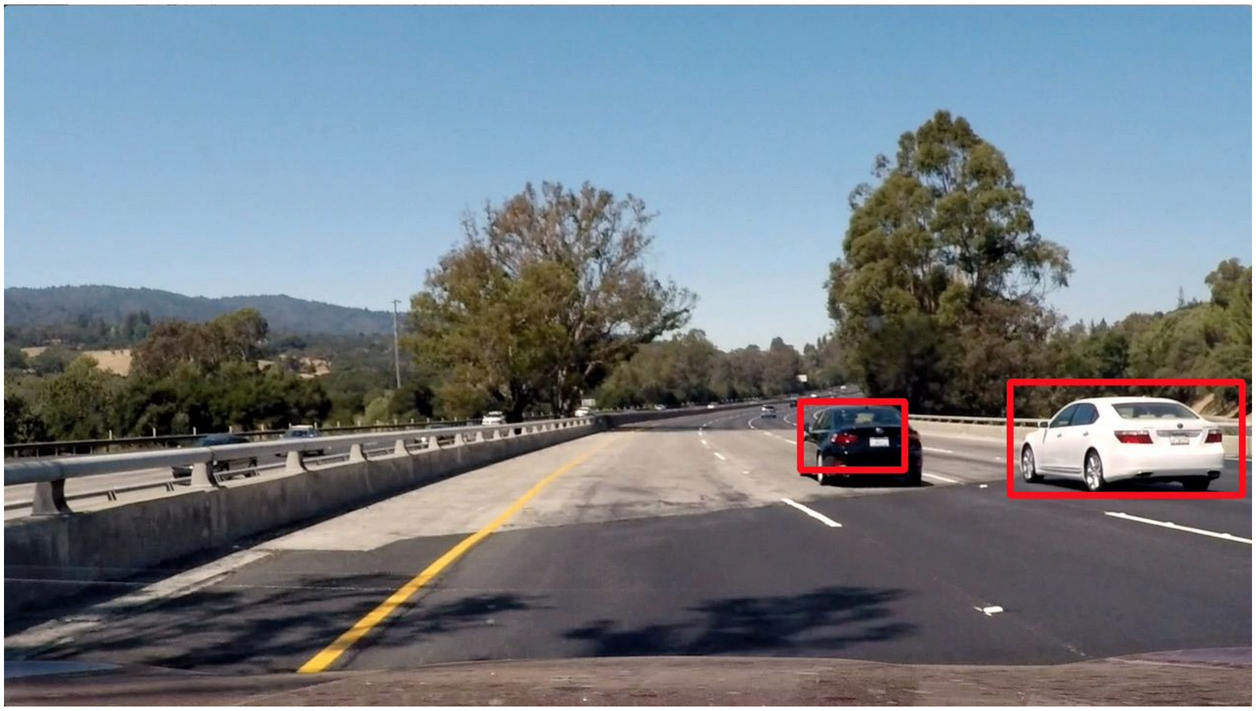
1. Crop the incoming image horizontally and vertically
2. Resize the image to match the model input layer format
3. Combine the result images into a batch and normalize the pixels
4. Use the model to predict the batch
5. Convert the prediction to the format that can be used to draw boxes
    a. Calculate probabilities for the result classes from the model
    b. Calculate prediction confidences
    c. Make a box object for the results
    d. Combine overlapped boxes using intersection over union
6. Draw boxes onto image using cv2.rectangle()

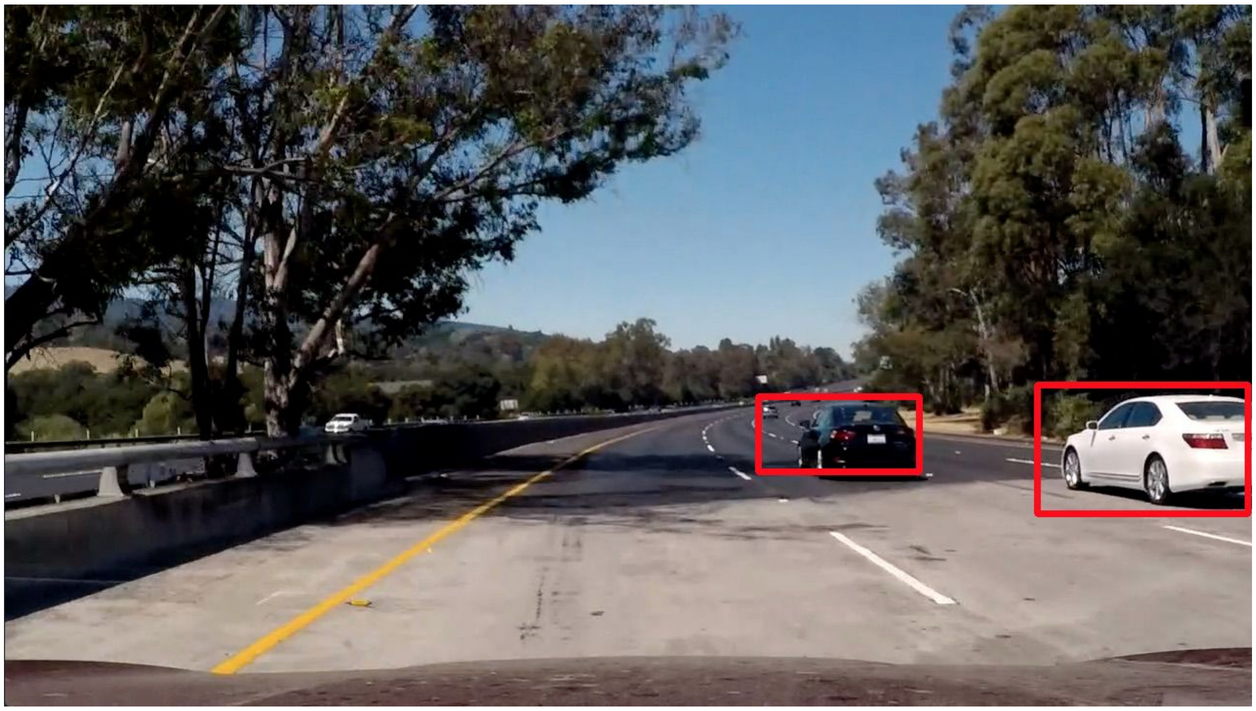
Test image results

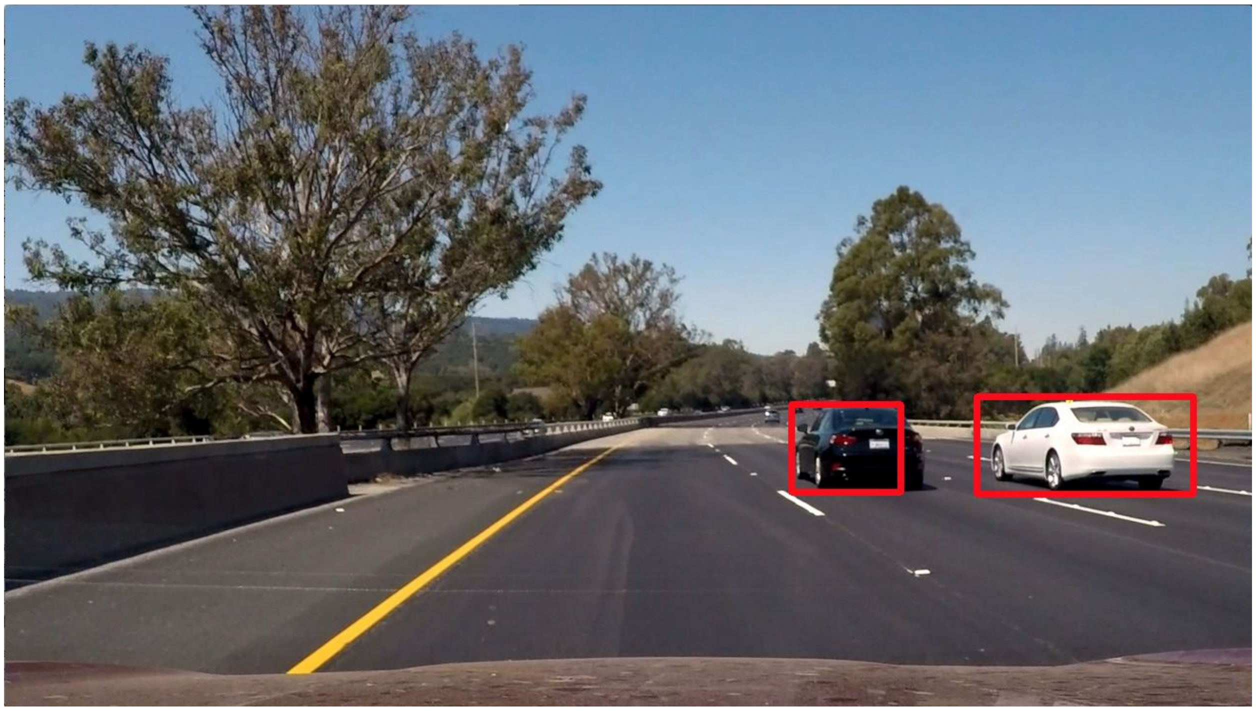






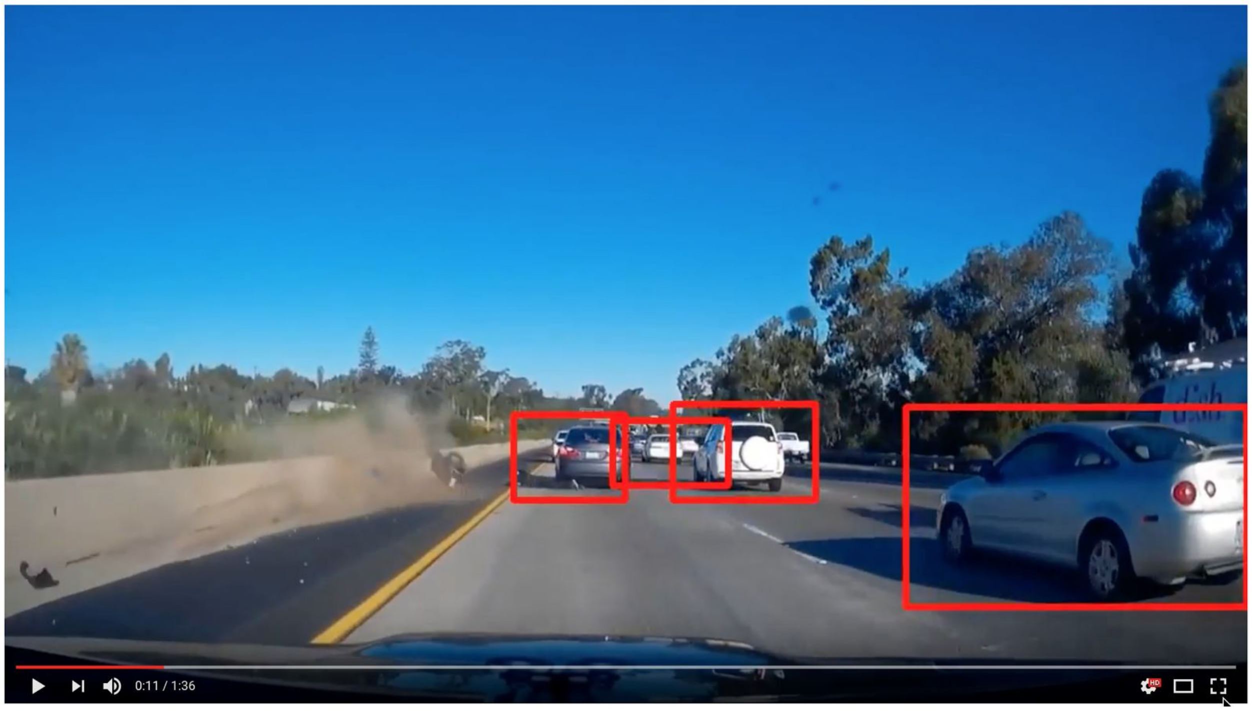






Video results

The result video is called video result.mp4 in the project root folder. I have also tried another video, with results named Motorcycle Crash Captured on Dashcam result.mp4. Youtube link



Discussion

This pipeline requires considerably less code to implement than the HOG+SVM route, and has a fairly good performance. However it's still not perfect. As we can see from the result video at its current state it has trouble inferring untrained objects like motorcycle and semi-trailers. Subsequent iterations of this pipeline could upgrade the model to use YOLO v2 which should see better results.