Vehicle Detection and Tracking using Deep Learning

Anirudh Mallya, Dr. Jay Urbain

Milwaukee School of Engineering

Abstract

Vehicle detection plays a crucial role in vision systems for autonomous vehicles and other tracking applications such as traffic surveillance control. In this paper, we present and elaborate on a neural network model that performs vehicle detection and tracking using Convolutional Neural Networks. A video is split into image frames, each of which is classified as containing vehicles or non-vehicle, and a bounding box is applied around each vehicle in the field of vision, to continuously track vehicles in the vicinity. Experimental results show that the proposed model is effective for real-time vehicle detection.

Conclusions and Future Direction

Our proposed method was accurate up to 99.63% using a Convolutional Neural Network model. The most common approach to this problem has been using Support Vector Machines, which detects vehicles using histogram of gradients, color histogram and spatial features. This model achieved test accuracies up to 99.89%. Although the accuracy and speed achieved using Support Vector Machines is a bit higher, the amount of data pre-processing that is required for feature extraction can be tedious. In the case of our Convolutional Neural Network detector, Vehi-ConvNet, the need for manual extraction of features is eliminated and allows for a rather simple implementation to achieve the same goal of real-time vehicle tracking with nearly the same accuracy. Since our model tracks vehicles using the sliding window approach, it is computationally expensive to carry out many convolutions on multiple frames of video. A potential work around this could be to use the YOLO (You Only Look Once) algorithm by Redmon, Joseph and Farhadi, Ali. This algorithm combined with our Convolutional Neural Network model could produce surprising results.

Introduction

Vehicle detection and real-time tracking are crucial to Intelligent Transportation Systems. They aid self-driving vehicles in understanding the behavior of other vehicles in the vicinity. Doing so can enable them to perform functions such as lane switching, intersection guidance and automated parking.

This paper focuses on CNN-based vehicle detection, segmentation and tracking based on rear view images, captured by a camera. We propose a simple approach to this using our CNN architecture – Vehi-ConvNet for this specific problem, trained on a specific large labeled dataset.

Vehi-ConvNet for vehicle & non-vehicle classification

Dataset and background information

The first step was to gather vehicle and non-vehicle datasets. These datasets were obtained from the Vehicle Image Database of Universidad Politécnica de Madrid (UPM), which consists of 64 x 64 dimensional, 8,792 color images of vehicles and 8,968 color images of non-vehicles in the RGB format. Both, vehicle and non-vehicle data was split into 70% training, 20% validation and 10% classification test data. This is a good split to ensure that the neural network validates its predictions, while training on the training set. When these data sets are imported, each pixel in an image is represented by a number ranging between 0 and 255, 0 being the brightest (white) and 255 being the darkest (black). Each of these numerical pixel values, represent color intensities in the RGB –Red, Green and Blue channels. Therefore, each image in the three-dimensional space may be represented as an image of shape 64 x 64 x 3. Below are examples of the used dataset.

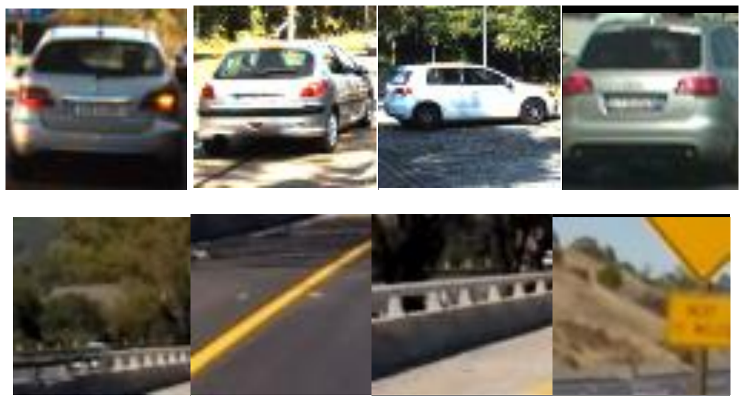

Figure 1: Examples of the dataset used

Just before an image is fed into the model, it undergoes feature scaling, where each pixels numerical value is divided by 255, such that each input pixel value lies between 0 and 1. This is a crucial stage, because the activation function in the final layer predicts probabilities between 0 and 1.

The architecture of Vehi-ConvNet

Vehi-ConvNet, our simple but powerful model consists of three layers of convolution, two of which are followed by max pooling. The idea behind this model is to have just the sufficient number of filter convolutions to learn patterns from images, to allow for precise real-time vehicle detection.

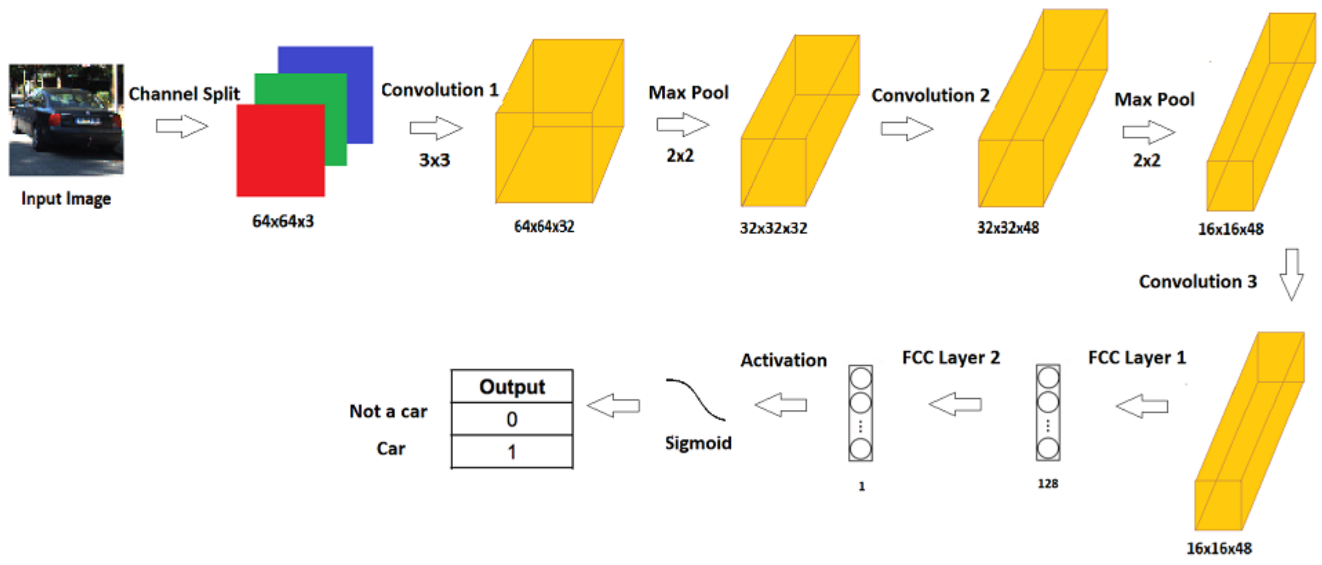

Figure 2: The architecture of Vehi-ConvNet

Training data is fed into Vehi-ConvNet as a four-dimensional array represented by (m, img_height, img_width, num_channels), where m is the number of training examples, img_height and img_width being the dimensions of the input image and num_channels corresponds to the number of channels in the input image.

This training data passes through a first convolution layer, where each 64 x 64 image is convolved with thirty two, 3x3 filters to create feature maps that analyze some patterns in the image. Vehi-ConvNet, uses a stride of 1 and same padding to maintain the height and width of the resulting image upon convolution. Max pooling is applied after this first layer, to shrink dimensionality while maintaining image features. This max pooling layer helps in downsampling data by reducing the number of parameters within the model. It also generalizes the results from convolutional filters, making the detection of features invariant to scale or orientation changes.

To further analyze images, a second convolutional layer, followed by a max pooling and third convolutional layer was added. This time, convolution was performed with forty eight, 3x3 filters. The increase in number of filters, results in calculation and analysis of more sophisticated feature maps and will allow for an in-depth learning in the following layers.

The output from the convolutional layers represents high-level features in the data. Upon completion of feature mapping during the second convolutional and max pooling layer, the features are flattened to create a single long feature vector to be used by the dense layer for the final classification. The dense layer learns non-linear combinations of these features to predict the probability that a vehicle exists in an image or not. Our model accomplishes this in two layers, one layer containing a hundred and twenty-eight units and the final layer containing one unit. This final layer outputs a zero (image is of a non-vehicle) or a one (image contains one or many vehicles).

Evaluation of Vehi-ConvNet

Below is a picture of the model predictions on downsampled test images.

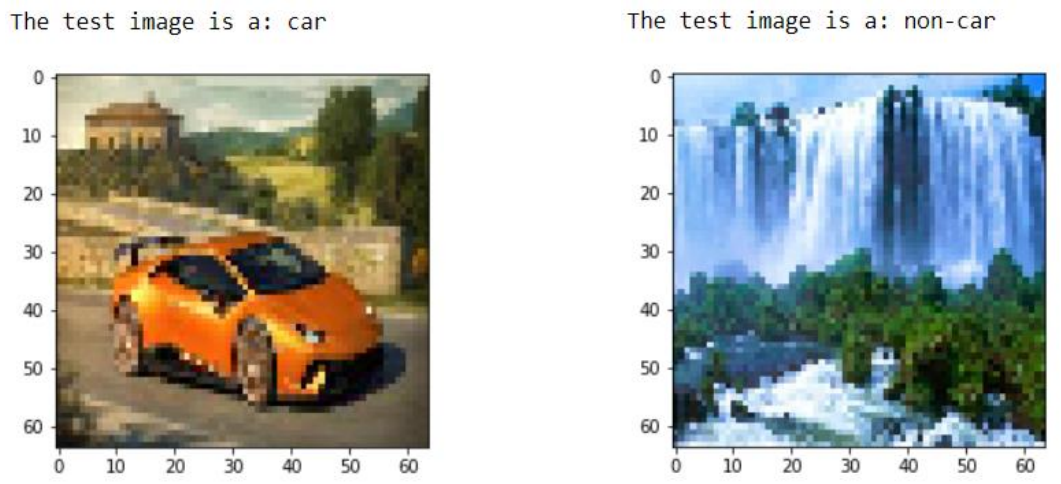

Figure 3: Test image predictions

This training, evaluation and testing process is repeated over a hundred epochs. Since our model doesn't use additional feature maps, it was necessary to train Vehi-ConvNet over larger epochs and more convolutional layers. The image below depicts the accuracy and loss of the model.

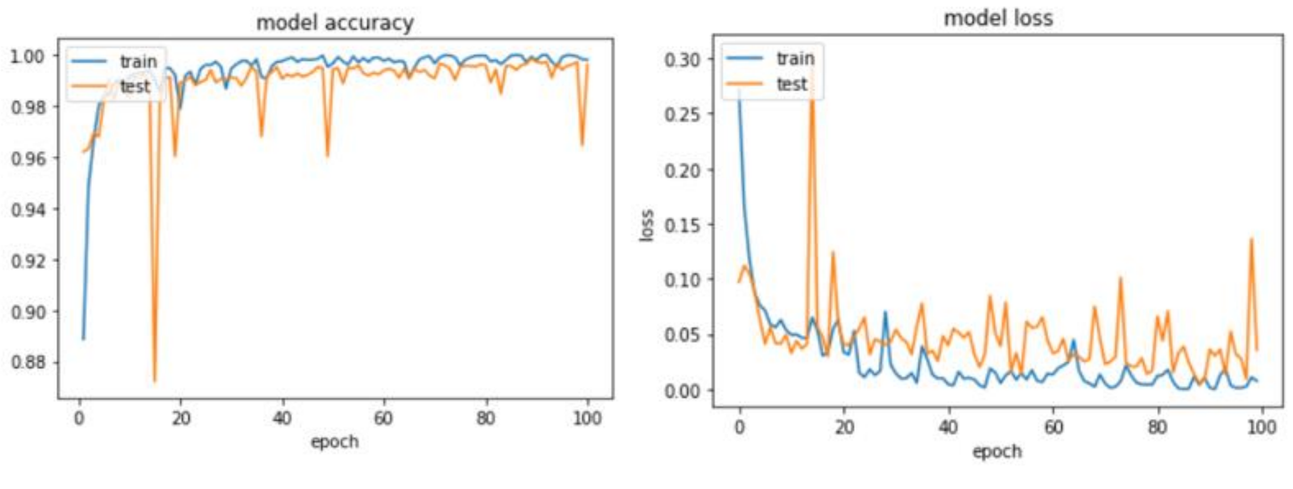

Figure 4: Model accuracy and loss over epochs

As the model trained, its accuracy on predictions was improved over time and the number of false positive predictions significantly decreased. Hundred epochs was just sufficient to obtain a robust model.

Vehi-ConvNet for Vehicle localization and tracking

Our design uses the concept of sliding windows to iterate through an image and predict the location of vehicles. Although the model was trained on 64x64 images, the image samples in the testing frame sets used for this purpose are of size 720x1280. These testing frame images were obtained from frame samples of a real-time video.

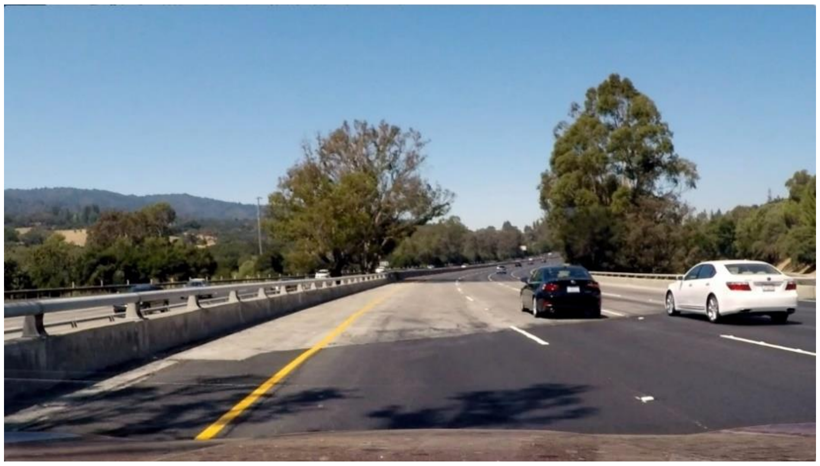

Figure 5: A test image frame sample

The region of interest for vehicle detection starts approximately at the 400th pixel from the top and spans vertically to the 720th pixel. Thus, we have a region of interest with the dimensions of 320x1280.

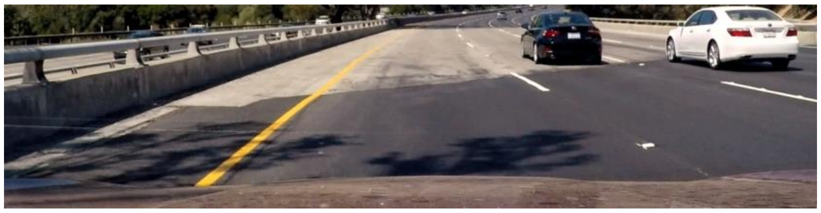

Figure 6: Region of interest in a test image frame

The region of interest for each image is split into windows containing 80x120 pixels. Each window is then resized and iteratively fed into the model as a 64x64 input image to obtain a probability of a vehicles existance in that window. Resizing is a crucial step because the neural network was trained on 64x64 images. This iterative process is called the sliding window approach.

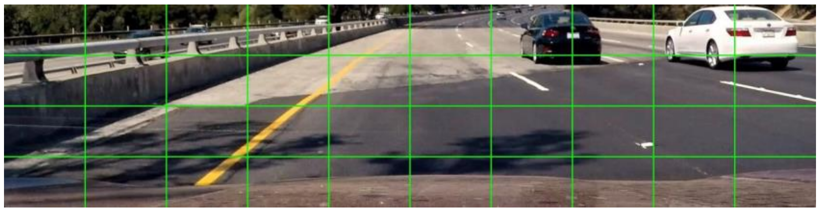

Figure 7: Windows analyzed in the region of interest (no overlapping shown)

To improve the precision of the sliding window predictions, we use an overlap of 75%. This increases the number of windows during prediction to allow for a clear cut outline of each vehicle. Each vehicle detected is stored as a label and represented in the form of a map using colors, the background being in black as shown in the picture below.

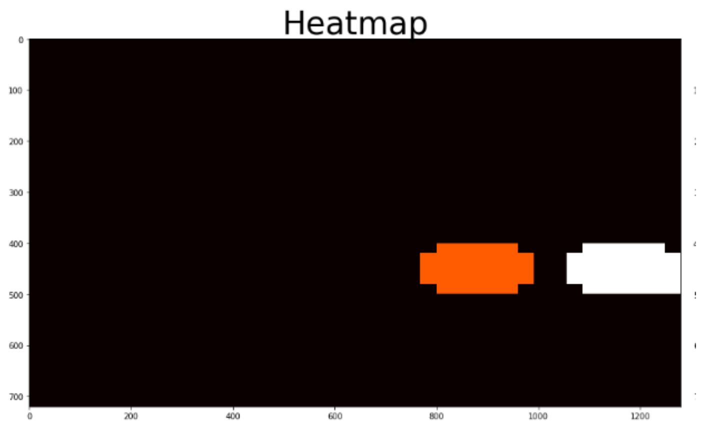

Figure 8: Heat map of vehicle locations in test image frame

Using the vehicle labels detected from the heat map, we extract the pixel coordinates for each image where the model detected the existence of a vehicle and applied bounding boxes to each set of coordinates.

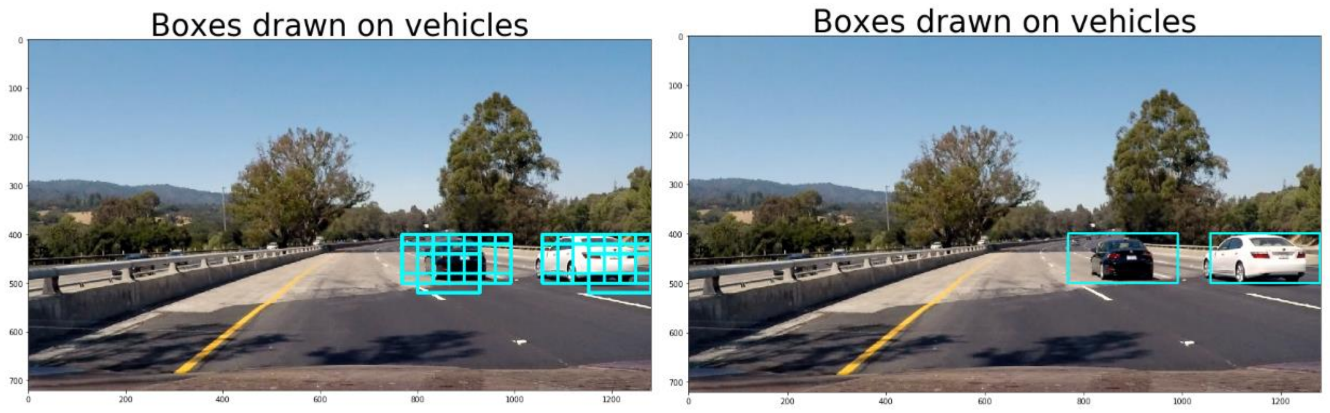

Figure 9: Bounding box predictions on test image frame

To determine the best fitting bounding box for each vehicle, we apply a confidence threshold wherein the area under intersection of all bounding boxes for a vehicle, has the most confidence and is therefore the bounding box that fits best.

References

deeplearning.ai. (2017, December). Coursera: https://www.coursera.org/learn/convolutional-neural-networks

NVIDIA Deep Learning Institute Online Lab. (2018, April 14). Qwiklab: https://nvlabs.qwiklab.com

Urbain, J. (2018, Jan 2). *CarND-Vehicle-Detection*. GitHub: https://github.com/jayurbain/CarND-Vehicle-Detection

Yiren Zhou, H. N.-T.-M. (2017). Image-based Vehicle Analysis using Deep Neural Network: A Systematic Study. *arxiv*.