CentraleSupélec

Computer Vision
Assignement 2 - Cars Detection

RIO Timothée

February 2022

Contents

1 Introduction

The objective of this challenge is to identify cars on a dash cam video. The main difficulties of this challenge is that the camera is moving which makes it difficult to apply standard pre-processing steps such as optical flow computation to identify moving objects or background subtraction. For this challenge we are provided with a training video made of around 2500 images on which cars are identified with bounding boxes. We are also provided with a testing video made of 203 images on which we are meant to identify cars with bounding boxes. Another difficulty of the task comes from the high context difference between training and testing videos (the former one has been taken in a city while the latter has been recorded on a motorway).

2 Pipeline and Processing

Our approach for this problem was to retrieve "vehicles" and "non vehicles" images to train a XGBoost classifier. We then reduced the false positive rate of the classifier using hard negative sampling. To draw bounding boxes on images, we firstly create areas of interest using window sliding with a big step size. Then we used the same classifier but with a much smaller step size on those areas of interest to find the locations on vehicles. After this step we still had many false positive so we implemented a pseudo-tracking method to remove the boxes that are not persistent throughout the video.

2.1 Data Sampling

The first step of our processing pipeline is to retrieve a training and validation set from the training video to train our classifier. To do so we retrieved on each training images the content of all bounding boxes to create a "vehicles" data set. We resized each sample to a $64 \times 64 \times 3$ size. We then slid a 64×64 window across all training images and each time the window did not overlap with a bounding box we retrieved the image to create a "non vehicle" data set. After this step we retrieved around 25000 vehicles images and 400000 non vehicles images. We randomly sampled 10000 images and 20000 non vehicles images to create our labeled images set.(Figure 1)

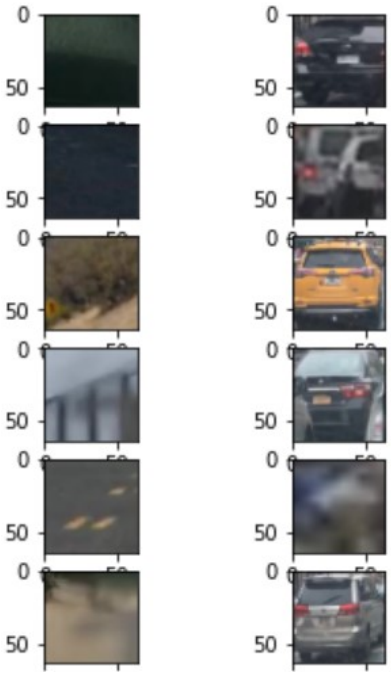

Figure 1: Sample from the "non vehicle" data set (left) and sample from the "vehicle" data set (right)

2.2 Data Augmentation

We then added a data augmentation pipeline on the vehicles images. We randomly picked 20% of the vehicles images and we make them pass through our home made augmentation pipeline. The pipeline behaves as follows: a random number of modifications is picked (between 1 and 4), the modification can be :

- salt and pepper addition
- Gaussian blurring

- random crop and resizing to $64 \times 64 \times 3$ size
- Brightness increase
- Random Flip

2.3 Features computation and model training

Concerning the features we tried two approaches (Bag of Sift and HoG features). HoG features basically learns a "template" of the objects while Bag of Sift is more robust to object modification, masking etc. (because it can recognize only parts of the object). However after several tries, we obtained better performances using HoG features. Concerning the hog features we also tried several parameters combination and we obtained the best performances using $orientations = 9$, $pixels\_per\_cells = (16, 16)$ and $cells\_per\_block = (3, 3)$. We tried to use less pixels per cells but this resulted in overfitting and in far too many features per image which made the model incredibly long to train.

We computed for each images 3 HoGs (one for each image channel). We also added three 32 bins histograms of the pixels intensities to the HoGs.

As for the model we tried to use a SVM (with probabilities enabled) and XgBoost. We obtained very similar performances with both models, however the XgBoost was much faster to train (5 minutes versus 15). We thus decided to use the XgBoost model. On the validation set we obtained an accuracy of 95%.

2.4 Hard negative sampling

Right after the training we implemented a very simple sliding window function to visualize the prediction of the model on the training image. We noticed that we had a lot of false positive (clouds, street poles, road markings, buildings were often considered as cars by the model). To solve this issue we decided to implement hard negative sampling. To do so, we took each image in the training video and slid a window through each of them. Each time the model confidently predicted (with a probability higher than 0.7) a vehicle on the image, if no parts of vehicles was present on the image we saved the sample in a "hard negative samples set" (i.e. model biggest mistakes set) (Figure 2)

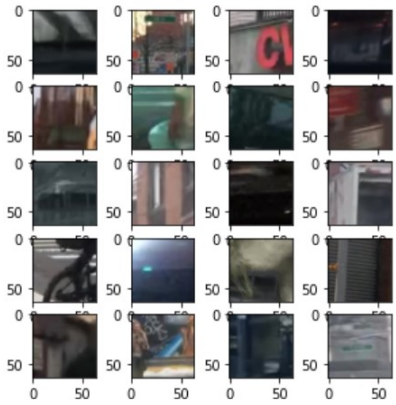

Figure 2: Sample from the hard negatives set

We added this set to our initial training data and retrained the XgBoost model. This new training slightly reduced the model accuracy (93% vs 95%) but it considerably reduced the number of false positives. Which is exactly what we wanted.

2.5 Window Sliding Cascades

To predict the presence of a vehicle in a region of the image, we wanted to slid windows of various sizes across the image and for each of them if the vehicle presence probability if above a threshold (to fine tune) we either draw a box or add 1 to each corresponding pixels in a heatmap. However this involves to try far too many windows and this is a very slow approach.

Instead we sub-sampled interest areas by sliding at first large and medium windows (256×256, 192×1932, 128×128, 64×64) with a high step size (64). If the confidence of the model exceeded 0.1 we retained the zone as an interest area (we used here a low probability because we wanted a high recall and did not care about the precision at this stage). We then implemented the same window approach but this time only on the interest areas that we had sub-sampled. This time we used a wider range of window sizes (from 32×32 to 192×932) as

well as a much smaller step size (10). This time we wanted accurate predictions so we increased the confidence level to be considered as positive to 0.7.

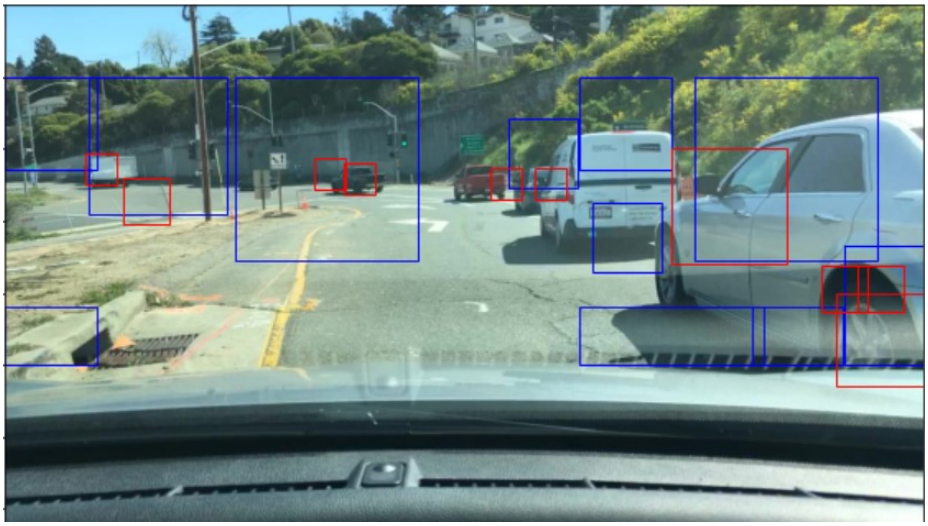

Figure 3: Visualisation of interest areas (in blue) and predicted bounding boxes (in red)

The result of this small "sliding windows cascade" can be observed on 3. As we can see, although the recall of the red boxes is not perfect, the false positive ratio is very low. We can also notice that the blue boxes (interest areas) have indeed capture all cars (high recall).

2.6 Pseudo Tracking

As we can see on 3 some bounding boxes still correspond to false positives. To solve this issue we took advantage of the fact that those images come from a video and are thus related to each other. After a rapid exploration through the predictions on the different images, we noticed that most of the false positives box are not stable on time (they appear and disappear from a frame to the next ones or they remain for only a few frames). The idea was thus to check the stability of the bounding boxes in a few "look-back frames"(parameter to fine tune). We decided to look back and not look ahead to be closer to a real usage of such camera. Concretely for each bounding box in a given frame, we count in the previous 10 frames the number of time another frame is present in an area around the frame we are studying (for the first frame back the area we explore is small, but for the tenth frame we explore a wider area). If in more than 60% of the past frames we found a box, we consider the box as a true positive and we keep it otherwise we discard it. A rapid visual check confirmed that this idea helped us to drastically improve the performance of our model.

3 Conclusion

The approach described in this short report helped us achieve around 34% mean Sørensen–Dice coefficient. To improve this score we could try to implement better tracking models such as models using Kalman filters. We could also try to implement more advanced region of interest identification methods (we could for instance use region growth)[1]. Another possibility could be to use advanced techniques for background identification (or moving objects identification) with a freely moving camera (some methods rely on sift detection and homography transformations). Of course the state of the art today for such task implies to use deep learning based models such as YOLO[2].

References

[1] Yang Li, Guangcan Liu, and Shengyong Chen. Detection of moving object in dynamic background using gaussian max-pooling and segmentation constrained RPCA. *CoRR*, abs/1709.00657, 2017.

[2] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.