Pattern Recognition Term Project: Vehicle Detection

Yen-chen Wu[1], Sheng-syun Shen[1], and Ching-ning Chen[2]

[1]Graduate Institute of Communication Engineering
[2]Graduate Institute of Information Management

2016.01.20

In this project, we proposed two approaches for car detection: You Only Look Once (YOLO) and Faster R-CNN. Both methods are end-to-end detector and achieve real-time vehicle detection: 45 fps by YOLO and 5 fps by Faster R-CNN. We trained the two models with PASCAL VOC 2007 dataset and did an experiment on imLab testing dataset, achieving accuracy of 0.07098 (YOLO) and 0.63576 (Faster R-CNN) in terms of F1 score.

1. INTRODUCTION

In this project, we aim to build a system that detects the existence of car(s) while given an image. If no car is detected, the system does not have to do anything; If yes, the system has to return a bounding-box to point out where the car is. To be exact, whenever the system is given an image (.png, .jpg, etc.) file, the system must output a corresponding comma-separated values (.csv) file, which indicates, if there exist a complete, not truncated or covered car in the image, the 4 coordinates (x1, x2, y1, y2) of the cars bounding box. If theres no (complete) car in the image, just output an empty .csv file.

We are given two image datasets for training so that we can make the system work properly and accurately: one is from KITTI, and the other is from imLab in NTU. There are 187,941 images in KITTI dataset, composed of 18,797 positive images and 169,144 negative images, and meanwhile imLab dataset has 3,360 images, which contains 336 positive examples and 3024 negative ones. In the original KITTI

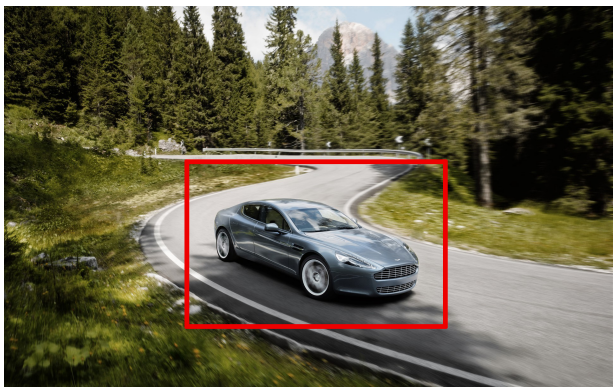

Figure 1: An example of a true positive prediction.

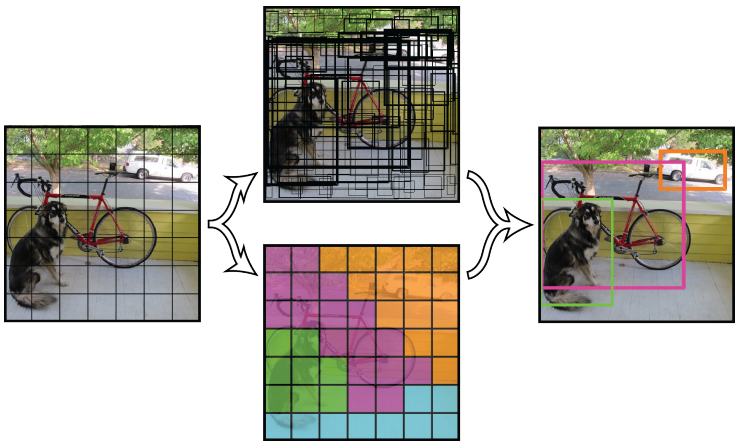

Figure 2: YOLO system flow chart.

dataset, the given training label contains cars, pedestrians, cyclists, and the corresponding coordinates where the object appears in the image; the imLab dataset is relatively simple, containing way fewer images. What distinguishes KITTI dataset and imLab dataset is that the average number of objects is way more in KITTI images than in imLab ones. We may see three or more vehicles in only one image from KITTI, while in the positive examples of imLab dataset, theres only one car for one image. We expected that the test dataset of the competition would come from imLab, so we did not do much configuration on handling multiple cars in one image except for cars appearing in the corner of images.

As for desired algorithms, we choose You Only Look Once (YOLO) and Faster R-CNN, which will be detailed in the following section. The main differences of these 2 algorithms are accuracy and fps. YOLO can handle 45 fps while Faster R-CNN can handle 5 fps, but the experiment outcome of YOLO is quite disappointing: only 0.07098 (F1 Score). On the other hand, Faster R-CNN has accuracy of 0.63576.

We will evaluate our system in three measure: precision, recall, and F measure (F1 Score). As for true positive determination, mentioned in slides given by TAs, only predictions that have intersect of union (IoU) higher than 0.7 will be considered to be true positive.

2. ALGORITHM

In this section, we are going to introduce the algorithms we applied in this project. Section 2.1. introduces YOLO, a fast and easy-to-implement method, while in section 2.2. we will show how to do all the object detection procedure through one neural network structure. Finally, we will compare these two models in section 2.3..

2.1. YOLO

YOLO, the abbreviation of "You Only Look Once", is a neural-network-based approach (Redmon et al., 2015). Unlike traditional object detection frameworks, YOLO is an end-to-end model, which means you can get the object position directly only by feed the original image into this network. We implemented the approach by

YOLO uses features from the entire image to predict each bounding box. Besides, it can detect all the bounding boxes simultaneously, which means YOLO network reasons globally about the full image and all the objects in the image. The most significant contribution of YOLO design is real-time prediction while maintaining high average precision. This is caused by its unique model design:

An image will be divided in to a 7×7 grid, and each grid cell has the potential to predict 2 bounding

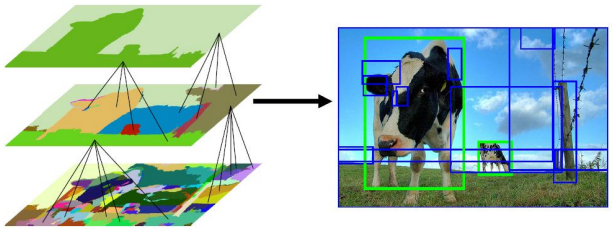

Figure 3: Selective search algorithm is generated from segmentation results.

boxes, which means the center of a bounding box lies inside the grid. Also, each bounding box is evaluated by a confidences score, and this score reflects how confident the model is that the box contains an object. For every box, we have 5 values to be predicted, 4 coordinates and one confidence score. Therefore, in general case we need to predict $7 \times 7 \times (5 + C)$ targets for each image. C is the class number, in this paper $C = 20$ while $C = 1$ in our case. The system flow chart is shown in Figure 2.

They implement this model as a convolutional neural network. The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates. With a view to extract image features, they adapted the well-known GoogLeNet model for image classification, and trained on the ImageNet 1000-class competition dataset (Russakovsky et al., 2014).

2.2. FASTER R-CNN

This section will introduce 3 sequential works conducted by Microsoft on Object Detection: `RCNN` (Girshick et al., 2014), `Fast-RCiNN` (Girshick, 2015), `Faster R-CNN` (Ren et al., 2015). The final adopted algorithm is Faster R-CNN concerning its out-standing performance.

2.2.1. RCNN

RCNN represents "Regions with CNN". It consists of three modules. The first generates category-independent region proposals by selective search (Fig 3). These proposals define the set of candidate detection available to our detector. The second module is a large convolutional neural network that extracts a fixed-length feature vector from each region. The third module is a set of class-specific linear SVMs. The three parts overview is shown in Figure 4.

2.2.2. Fast-RCNN

Fig.5 illustrates the Fast R-CNN architecture. A Fast R-CNN network takes as input an entire image and a set of object proposals. The network first processes the whole image with several convolutional (conv) and max pooling layers to produce a conv feature map. Then, for each object proposal a region of interest

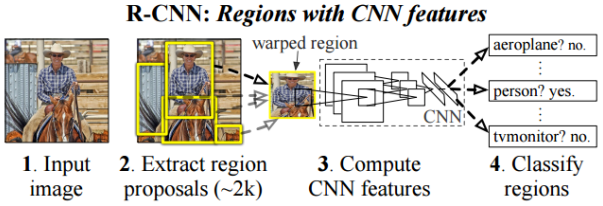

Figure 4: Object detection system overview.

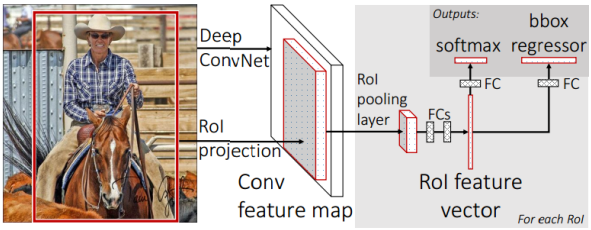

Figure 5: Object detection system overview.

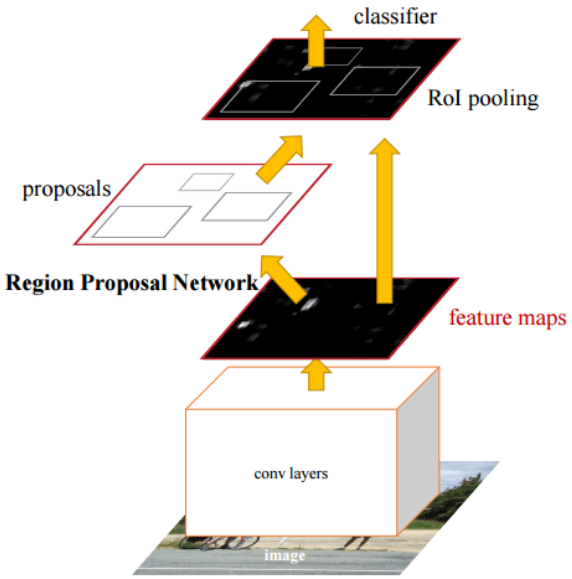

Figure 6: Object detection system overview.

(RoI) pooling layer extracts a fixed-length feature vector from the feature map. Each feature vector is fed into a sequence of fully connected (fc) layers that finally branch into two sibling output layers: one that produces softmax probability estimates over K object classes plus a catch-all background class and another layer that outputs four real-valued numbers for each of the K object classes. Each set of 4 values encodes refined bounding-box positions for one of the K classes.

2.2.3. Faster R-CNN

RPN(Region Proposal Network) replaces selective search and further merge RPN and Fast R-CNN into a single network by sharing their convolutional features using the recently popular terminology of neural networks with attention mechanisms, the RPN component tells the unified network where to look.

2.3. Model Comparison

- Both are end-to-end models.
- YOLO can been regarded as a : Image Embedding + Region Proposal Network .
- YOLO has weak overlapping rate: The loss score every bounding box returns to model are equal, which means whether the size of a box is large or small, they just caused error equally. As a result,

comparing to the ground truth, the bounding box YOLO gave might be not so accurate.

- YOLO inclines to detect background as objects: To save the computation cost, YOLO divided an image into 7×7 equally-sized grids and gave every grid an object label. As a result, the object labeling isn't that accurate.
- Speed : 45fps(YOLO) v.s. 5 fps(Faster R-CNN): YOLO dropped out some accuracy to achieve real-time classification, while Faster R-CNN still regard accuracy as its criterion. These two model are suitable for different situation, depending on user demands.

3. IMPLEMENTATION DETAILS

3.1. ENVIRONMENT

Linux, Ubuntu 14.04
GPU: Titan X (12G)
RAM: 32G, 8 kernels

3.2. FASTER-RCNN

- **Implementation details**: Modified open-source matlab and python code, using caffe (Jia et al., 2014) for Neural Network training.
- **Fine tuning**: Python coded, the following are pseudo code of our algorithm to fit the task of imLab data.

Algorithm 1 FINE TUNING

procedure DETECTOUTLIER
  $cropimg = bounding(img)$
  $center = cropimg.center$
  $mean = cropimg.gray_scale.mean$
  for i in $range(cropimg.shape[3])$ **do**
    $rgb[i] = cropimg[:, :, i].mean$
  end for
  std = rgb.std
  if $center > thres_1$ AND $mean < thres_2$ AND $std < thres_3$ **then**
    return $cropimg$
  end if
end procedure

3.3. YOLO

Modified open-source C code, and all the settings are according to original paper.

3.4. EVALUATION AND VISUALIZATION

Provided by TAs, and slightly modified IO for convenient. Besides, We designed a visualization system to compare our results and ground truth. It is a modified matlab code provided by KITTI development toolkit. The example of our visualization system can be show at Figure 7.

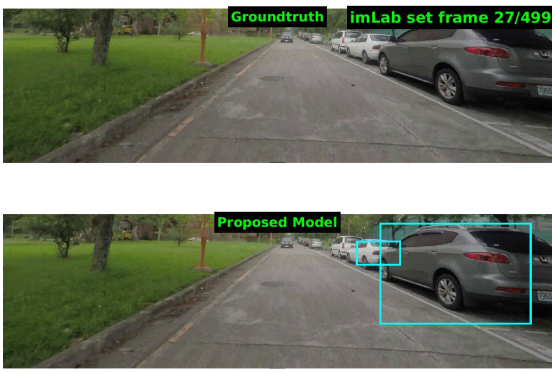

Figure 7: Visualization System.

4. EXPERIMENTS

4.1. DATA DESCRIPTION

We train our model on Pascal voc 2007 and evaluate on KITTI and imLab data set.

4.1.1. Pascal voc 2007

20 classes:

- Person: person
- Animal: bird, cat, cow, dog, horse, sheep
- Vehicle: aeroplane, bicycle, boat, bus, car, motorbike, train
- Indoor: bottle, chair, dining table, potted plant, sofa, tv/monitor

Train/validation/test: 9,963 images containing 24,640 annotated objects.

4.1.2. KITTI

The object detection and object orientation estimation benchmark consists of 7481 training images and 7518 test images, comprising a total of 80.256 labeled objects. All images are color and saved as png. For evaluation, we compute precision-recall curves for object detection.

4.1.3. ImLab

There are 500 testing images, 3024 negative examples, and 261 positive examples. It's not a general car detection task, where it only needs to specify "the black car". So, we had some additional post-processing on results.

4.2. EXPERIMENTAL RESULTS

Table 1 shows our experimental results. Row (A) is the YOLO model prediction result, and row (B) shows the accuracy of Faster R-CNN while row (C) shows the results after fine-tuning. We can discover that whether the precision or recall score of YOLO model are much lower than Faster R-CNN. However, YOLO has a faster prediction speed than Faster R-CNN. Compare Rows (B) to (C), we found that after applying the fine-tuning algorithm, we can achieve higher performance on our testing dataset.

Table 1: Experiment Results on imLab data set

Learning Model	Precision	Recall	F1-measure	Speed
(A) YOLO	0.05303	0.10728	0.07098	0.05s/image
(B) Faster RCNN	0.37248	0.85057	0.51809	0.37s/image
(C) Tuned Faster RCNN	0.55977	0.73563	**0.63576**	0.39s/image

With a view to examine our prediction results. we put the predicted bounding boxes into our visualization system. The comparison between YOLO and Faster R-CNN can be shown at the following videos:

- **YOLO**: `https://youtu.be/ZVNdVLF3vEs`
- **Faster R-CNN**: `https://youtu.be/BPQk495qC3k`

5. CONCLUSION

We implemented two algorithms, YOLO and Faster R-CNN, to build a real-time car detection system, with some fine tuning to handle cropped car problem. YOLO gives extremely high fps but its accuracy is disappointing, so obviously there's still much work to do on YOLO and at the present stage cannot be used to participate the competition. On the contrary, Faster R-CNN, the chosen algorithm for competition, gives satisfying accuracy, achieving F-measure of more than 0.93 in the competition. Still, we didn't distinguish car and motorbike/motorcycle in the training phase, so the false-positive prediction caused by images containing motorbike happened several times.

As a result, we cannot achieve accuracy of 1 by a small margin. In addition, we only focused on imLab testing set, whose images are relatively simple, so our system won't give satisfying performance under complicated traffic conditions like ones in Taipei. If we want to put the system in real-world practice, we should put more emphases on real-time multiple cars detection.

Since Dense box 2 is in first place on leader-board of KITTI competition. If we have more time, our future work is to implement it's previous published version: DenseBox (Huang et al., 2015).

REFERENCES

GIRSHICK, R., DONAHUE, J., DARRELL, T., AND MALIK, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 580–587.

GIRSHICK, R. B. 2015. Fast R-CNN. *CoRR abs/1504.08083*.

HUANG, L., YANG, Y., DENG, Y., AND YU, Y. 2015. Densebox: Unifying landmark localization with end to end object detection. *CoRR abs/1509.04874*.

JIA, Y., SHELHAMER, E., DONAHUE, J., KARAYEV, S., LONG, J., GIRSHICK, R., GUADARRAMA, S., AND DARRELL, T. 2014. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

REDMON, J., DIVVALA, S., GIRSHICK, R., AND FARHADI, A. 2015. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*.

REN, S., HE, K., GIRSHICK, R., AND SUN, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*. 91–99.

RUSSAKOVSKY, O., DENG, J., SU, H., KRAUSE, J., SATHEESH, S., MA, S., HUANG, Z., KARPATHY, A., KHOSLA, A., BERNSTEIN, M., ET AL. 2014. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 1–42.