PyTorch YOLOv3 Object Detection for Vehicle Identification

Tesa Ho, Mohith Ravendra
CS230, Stanford University
{tesaho, mohithr}@stanford.edu

Abstract

Detecting real world vehicle objects captured from car mounted cameras requires manual labelling of video images. Previous vehicle object detection papers such as the winners of the 2018 AI City Challenge [1] used a training set of over 4,500 hand labelled images. In this paper, I attempt to automate this task by applying transfer learning to a YOLOv3 model trained on Imagenet and then retrained on a set of stock car images and a small subset of hand labelled images taken from front-mounted dashboard camera videos. The mean Average Precision (mAP) of the validation set is used to determine the effectiveness of model vehicle classification. There is a significant variance issue between the validation and training set because the video images are taken in 1) various weather and lighting conditions and 2) the stock images have different image perspectives. The experimental results demonstrate that the YOLOv3 model can reach an overall 16.07% mAP after 60 epochs of training and can identify classes of vehicles that had few training examples in the dataset.

Keywords:

Object detection, vehicle detection, YOLOv3, deep learning, convolutional neural network.

1. Introduction

Deep learning vehicle detection can be split into two different model strategies: 1) a single shot object detector (SSD, YOLO, YOLOv2, and YOLOv3) and 2) a region-based object detector (R-CNN, Fast R-CNN, and Faster R-CNN). Recent papers such as Tang et al [1] and Sang et al [2] demonstrate the success that YOLOv2 has had on object detection in the 2018 AI City Challenge. In this paper, a PyTorch version of Redmon's [3] YOLOv3 model is applied to vehicle images from the Nexar Challenge 2 dataset, NEXET [4]. The models were pre-trained on Imagenet data and then trained on a custom dataset consisting of the Stanford car data set [5] and the Nexar Challenge 2 vehicle dataset, NEXET. The trained models were then evaluated using the mean average precision metric (mAP) on a random sample of NEXET vehicle images.

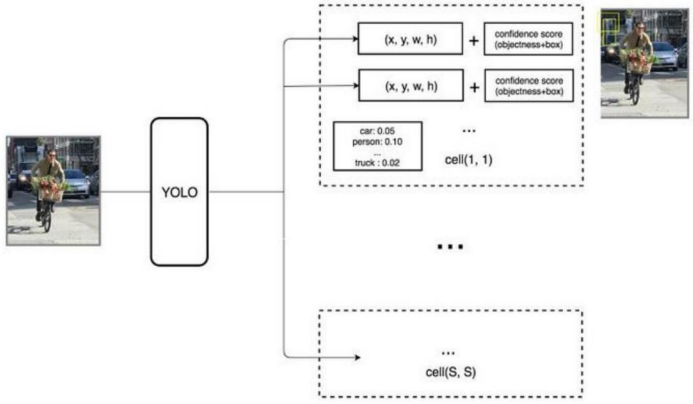

Figure 1. YOLO produces SxS predictions with B boundary boxes. [6]

2. Related Work

Region based object detection model, R-CNN[7], were the first deep learning object detection models that were successfully applied to vehicle detection. Fast R-CNN[8] improved on R-CNN by using a feature extractor (CNN) to extract features over the whole image thereby speeding up the training and inference process. Faster R-CNN[9] further improved the training and inference speed and proved to be usable for real-time vehicle detection in reference [10]. Redmon et al. introduced a single shot detector model YOLO in 2016 [11] which further greater reduced the speed of detection and improved the accuracy. YOLOv2 [12] was an improvement over the original YOLO model with additional model features such as batch normalization, multi-scale training, dimensional clustering, and a high resolution classifier.

3. Dataset

The Stanford car dataset consists of 8,144 stock car images that are well lit and clearly identify the vehicle. The original Stanford car dataset did not have vehicle classification labels so each image was manually relabeled. The dataset omitted images of buses, minibuses, trucks, and motorcycles.

The NEXET dataset consists of 1,258 car images taken from videos captured from front mounted cameras and reflect real world data. The NEXET images include night, twilight, and daytime images taken in weather conditions

that include rain and snow [Figure 2]. The original NEXET sedan images labels included suvs and hatchbacks labels and were manually re-labelled to reflect the new classes.

The training set consists of all 8,144 Stanford car images and 811 NEXET images (96% of all images). The NEXET training images were randomly selected with the same class distribution as the original NEXT distribution. The validation set consists of 377 NEXET images randomly selected with the same class distribution as the full NEXET data set. The vehicle classes included are: sedan, hatchback, bus, pickup, van, truck, and suv.

Figure 2: Stanford car image (left). NEXET car image (right).

There are nine vehicle classifications that are tested: sedan, hatchback, bus, pickup, minibus, van, truck, motorcycle, and suv. The training set does not include examples of minibus and motorcycle.

	Train				Validation	
	Stanford	Nexet	Total	%	Nexet	%
sedan	4,851	754	5,605	58.5%	247	52.9%
hatchback	554	53	607	6.3%	17	3.6%
bus	0	60	60	0.6%	19	4.1%
pickup	593	92	685	7.2%	30	6.4%
minibus	0	0	0	0.0%	0	0.0%
van	541	248	789	8.2%	81	17.3%
truck	0	102	102	1.1%	33	7.1%
motorcycle	0	0	0	0.0%	0	0.0%
suv	1,605	123	1,728	18.0%	40	8.6%
Total	**8,144**	**1,432**	**9,576**	**100.0%**	**467**	**100.0%**

Figure 3. Class image count for training and validation dataset.

3.1 Pre-processing

The following pre-processing steps were utilized:

- pad each image to a square
- resize each image to 416 x416.

The colors of each picture were augmented for saturation=1.5, exposure=1.5, and hue=0.1.

4. Methodology

The PyTorch YOLOv3 model used in this paper is based on the Darknet-53 YOLOv3 by Joseph Redmon and Ali Farhadi [3]. YOLOv3 is an object detector that splits an image into a grid and predicts one object per grid cell. Each grid cell then predicts B number of boundary boxes for an object and every boundary box is given a box confidence score. Only one object is detected per grid cell along with the conditional class probabilities. A class confidence score is then calculated by multiplying the box confidence score by the conditional class probability. YOLOv3 predicts an objectness score for each bounding box using logistic regression. An objectness score of 1 is given to the bounding box prior that has the highest overlap with the ground truth object. No loss is assigned to a bounding box prior if the prior does not overlap with a ground truth object.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 4. YOLOv3 DarkNet-53 architecture [3].

The loss function is a sum squared of error between the predictions and ground truth is composed of the classification loss, localization loss, and confidence loss. Duplicate boxes are removed through non-maximal suppression. Unlike the original YOLO model, the final softmax function is replaced with individual logistic classifiers that utilize a binary-cross entropy loss function to predict classes. This allows for multiple labels to be assigned to a bounding box

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
$$

Figure 5. YOLOv3 loss function [3].

4.1. Hyperparameters

The following hyperparameters were optimized along the following ranges using the validation set mAP as the evaluation criteria:

- Learning rate: [0.00001, 0.0001, 0.0005]
- Non-maximal suppression threshold: [0.3, 0.5, 0.8]
- Confidence threshold: [0.01, 0.05, 0.1]

The confidence threshold and non-maximal suppression threshold were selected as hyperparameters as they filter the number of bounding boxes evaluated before the IOU calculation. The learning rate was chosen to adjust the training loss speed.

Due to compute constraints, the hyperparameter models were tuned on smaller epoch sets. There is a fairly high variance between the training set and validation set because of the image differences. Images were padded and resized to a 416 x 416 shape.

Table 1. Model parameters

Batch normalization	Yes
Batch size	6
Multi-scale training	Yes
Momentum	0.9
Decay parameter	0.0005
Learning rate	0.0001
Confidence threshold	0.05
NMS threshold	0.5
IOU threshold	0.5

4.2. Set-Up

Training was done primarily on a NVIDIA GeForce GTX 1070 with 16GB memory on Ubuntu 18.04 using Intel core i7 8th generation CPU.

5. Results

The training loss moved close to the minimum within the first 20 epochs and oscillated there for the remainder of the epochs. The validation mAP peaked at 0.1607 at the 61 epoch. The sedan mAP value of 0.5040 for IOU_0.50 is on par with the YOLOv3 mAP results on the COCO dataset.

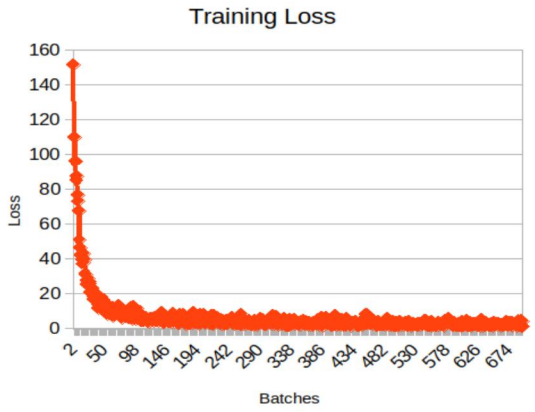

Figure 6. Training loss.

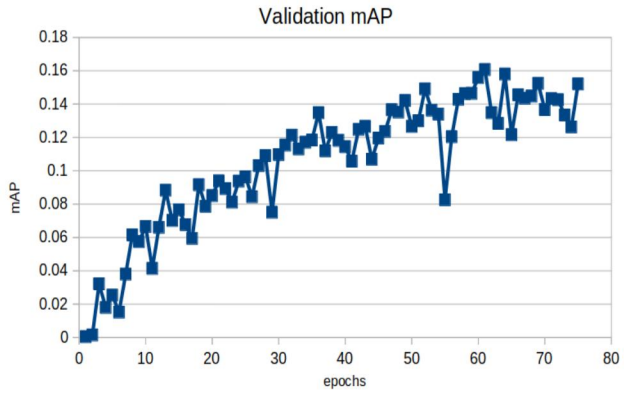

Figure 7. Validation mAP.

Using the standard IOU_0.5 benchmark, the sedan class had the highest mAP score. The model classified hatchbacks, pickups, vans, and suvs well at IOU_0.5. There were no minibus or motorcycle classification which was expected since the training set did not have any examples. There were only 60 bus training examples which may have contributed to the lack of classification of busses.

IOU_0.25 had the highest total mAP score of 0.1720. At this threshold, truck, suv, and hatchback reached their maximum mAP score.

	mAP				
	IOU_0.005	IOU_0.25	IOU_0.50	IOU_0.75	IOU_0.95
sedan	0.6125	0.5916	0.5040	0.1685	0.0000
hatchback	0.0756	0.1406	0.1406	0.0867	0.0028
bus	0.0000	0.0000	0.0000	0.0000	0.0000
pickup	0.0456	0.0542	0.0617	0.0063	0.0000
minibus	0.0000	0.0000	0.0000	0.0000	0.0000
van	0.1916	0.1855	0.1937	0.1429	0.0000
truck	0.1088	0.0971	0.0900	0.0387	0.0000
motorcycle	0.0000	0.0000	0.0000	0.0000	0.0000
suv	0.1353	0.1353	0.1353	0.0813	0.0000
total	0.1670	0.1720	0.1607	0.0749	0.0004

Figure 8. Class and total mAP with varying IOU thresholds.

5.1. Error Analysis

The images that had mAP less than 1 and misclassified objects can be attributed to:

a) Poor lighting conditions.

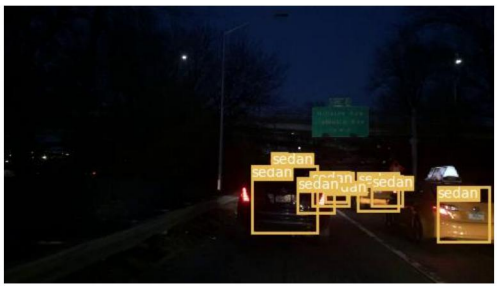

Figure 9. Night image with incorrect bounding boxes.

b) Lack of training objects.

Figure 10. No motorcyle or scooter training examples.

c) Perspective issues.

Figure 11. Suv images from the rear were mislabeled as a sedan and van.

d) Poor image visibility.

Figure 12. Rain on windows with light glare.

6. Conclusion

This paper has illustrated that transfer learning with YOLOv3 is a viable solution for vehicle detection given that the training set contains a high number of class samples with the same image quality as the validation set. The sedan IOU_0.5 mAP score of 0.5040 is in line with the COCO scores provided by Redmon [3].

In order to avoid hand labelling additional video images, data augmentation such as scaling, flipping, brightness variation, perspective transform, and image sharpening on minority classes could be used to enhance the training set.

7. Contributions

Tesa Ho was responsible for obtaining the Stanford data set, hand labelling the Nexar data, data analysis and split, YOLOv3 model build and training, data analysis, final report and presentation. Mothith Ravendra was responsible for obtaining the Nexar data.

Erik Lindernoren's git repository [14] was used as the base reference for the PyTorch implementation. Code used in this project can be found here https://gitlab.com/cs230/vehicle_tracking/.

A big mahalo to Patrick Cho, teaching assistant extraordinaire, who was an amazing resource through this project.

References

[1] Z. Tang, G. Wang, H. Xiao, A. Zheng, J.N. Hwang. "Single-camera vehicle tracking and 3D speed estimation based on fusion of visual and semantic features". 2018 AI City Challenge.

[2] J. Sang, Z. Wu, P. Guo, H. Hu, H. Xiang, Q. Zhang, B. Cai. "An Improved YOLOv2 for Vehicle Detection". Sensors December 2018.

[3] J. Redmon, A. Farhadi. "YOLOv3: An Incremental Improvement", University of Washington. 2018.

[4] Nexar NEXET dataset. https://www.getnexar.com/challenge-2/

[5] Stanford cars dataset. https://ai.stanford.edu/~jkrause/cars/car_dataset.html

[6] J. Hui. "Real-time Object Detection with Yolo, YOLOv2, and now YOLOv3". www.medium.com. 2018.

[7] R. Girshick, J. Donahue, T. Darrell, J. Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". 2014 IEEE Conference on Computer Vision and Pattern Recognition, June 2014. p 580-587.

[8] R. Girshick. "Fast R-CNN". 2015 IEEE Conference on Computer Vision and Pattern Recognition, December 2015. p 1440-1448.

[9] S. Ren, K. He, R. Girshick, J. Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015.

[10] S. Azam, A. Rafique, M. Jeon. "Vehicle pose detection using region based convolutional neural network". International Conference on Control, Automation, and Information Sciences (ICCAIS). October 2016. p 194-198.

[11] J. Redmon, S. Divvala, R. Girshick, A. Farhadi. "You only look once: Unified, real-time object detection". June 2016. p 779-788

[12] J. Redmon, A. Farhadi. "YOLO9000: Better, Faster, Stronger". IEEE Conference on Computer Vision and Pattern Recognition (CVPR). July 2017. p 6517-6525.

[13] J. Hui. "What do we learn from region based object detectors (Faster R-CNN, R-FCN, FPN)?". www.medium.com. 2018.

[14] https://github.com/eriklindernoren/PyTorch-YOLOv3