# Final Project Report

**Kexin Yan**
Department of Computer Science
University of Toronto
Toronto, ON
kexin.yan@mail.utoronto.ca

**Jiawei Yu**
Department of Computer Science
University of Toronto
Toronto, ON
jiawei.yu@mail.utoronto.ca

**Kaijian zhong**
Department of Computer Science
University of Toronto
Toronto, ON
zhongk@cs.toronto.edu

## Abstract

Autonomous driving requires fast and accurate object detection. We chose YOLOv3 and SSD, two single neural network detection models, and compared their performance on a large-scale, diverse driving dataset, BDD100K. Although both models are far from being good enough in accuracy, we found YOLOv3 to perform better on common objects, while SSD is more robust to class imbalance. YOLOv3 is also more robust to environment change. Both models achieved good inference speed (>70FPS), for images of size around $300^2$.

## 1 Introduction

Object detection is widely used in different computer vision tasks including autonomous driving. In order to make real-time applications like self-driving vehicles possible, fast and accurate object detection algorithms are required. Object detection involves detecting the objects(generating potential bounding boxes) and recognizing them(running a classifier on these proposed boxes). Many approaches like R-CNN [7] use separate algorithms to finish the whole task. However, approaches like SSD [11] and YOLOv3 [3] use a single neural network instead. These models are therefore faster than traditional models (achieving 40+ FPS in test time).

In this project, we conduct experiments with SSD and YOLOv3 on a new self-driving dataset(BDD100K) [2]. Compared to YOLOv2, YOLOv3 is a more recent version that performs better, and also its existing implementations are more compatible with modern deep learning platforms. Our goal is to find out which one is the better model when it comes to self-driving-related tasks.

In order to do that, we used pretrained models for both algorithms and trained them on the BDD100K dataset. We evaluated these two algorithms in terms of testing speed and accuracies on different objects and settings.

## 2 Related Work

The usage of neural networks for tasks related to object detection date back to the 1990s [4], but it wasn't until the last decade that deep convolutional network achieved great results in complex computer vision tasks [5], including object detection [6]. The R-CNN architecture proposed by Girshick et al. uses a selective search algorithm to produce region proposals, feed them through a

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

CNN to compute features, and use linear SVMs to predict classifications (2014). The model defeated the then SOTA in terms of accuracy, but is slow and hard to train because it needs to feed each region proposal through the convolution layers. This was partially addressed by Girshick himself [8] with Fast R-CNN, where an entire image is fed through the convolution layers, and each region of proposal then goes through a RoI pooling layer and fully connected layers for classification and localization. In 2015, Ren et al. proposed Faster R-CNN [9], in which a Region Proposal Network is used in combination with sliding windows to directly propose regions of interest on computed embeddings, achieving improvement in both speed and precision.

While Faster R-CNN produced impressive results, its testing rate still could not achieve real-time speed. YOLO [10], YOLOv2, YOLOv3 and SSD are examples of efforts to speed up computation without much sacrifice on accuracy. With computer vision models being more popular and useful for autonomous technologies [13], a real-time detector is likely more useful than a more accurate but slower one. In addition to autonomous driving, object classification and localization can also be used to gather useful data and build smarter traffic systems [14].

There are various datasets with the purpose of fueling automated driving research [1], the more popular ones being KITTI [15], Caltech Pedestrian [16], nuScenes [17], and others. The Berkeley Diverse Driving (BDD100K) dataset [2] is among some of the most popular ones, which we describe in the nest section.

## 3 Dataset

BDD100K [2] is one of the largest and most diverse driving video dataset. The videos are taken from inside the car, facing either the forward or the sideway direction, in real driving scenarios in the U.S. Figure 5 in the appendix include some sample images. We are using the image version of the dataset, where the images are taken sampled at 1 frame per 10 seconds for each video. Each image has been annotated with bounding boxes around objects like cars, pedestrians, traffic signs and more. There are 79,853 frames annotated in total. A small subset of these images have also been annotated for fine segmentation, but in this experiment we use the bounding boxes only.

The dataset covers various objects including cars (55% of all objects), traffic signs/lights(33%) and pedestrians(7%). There are also other vehicles classes, but most of them are under-represented. For example, buses account for less than 1% of all objects, while bicycles and motorcycles are even fewer. In terms of object sizes, traffic signs and traffic lights are relatively small( 0.1% area of the total image). One car on average takes up 1% of an image's area, and buses and trucks are larger (> 3%). For a detailed breakdown of each class's statistics, please see table 1 in appendix. Our experiments will aim to analyze the models' performance for each of these classes.

The dataset is quite diverse in terms of time of day, road condition and weather conditions. In terms of time of day, the dataset is relatively balanced between daytime (52% of all images) and night time (40%). The rest are taken during dawn/dusk. In terms of road conditions, the dataset is 60% city roads, 25% highway, 12% residential, and some other conditions such as tunnel and parking lot.

One major challenge with regard to our dataset is its data imbalance. There are 13 object classes in total, but the top three categories (traffic lights, traffic signs and cars) make up over 88% of all objects. Other classes such as trailers or trains have every few counts (less than 0.01%). This makes prediction on rare classes extremely difficult. Another major challenge is the size of objects. Traffic signs and traffic lights are both extremely important items when it comes to driving, but they have the smallest area out of all classes. In a 300 by 300 image (size of image input to our SSD model), a traffic light takes up only 45 pixels on average. In addition objects often overlap with each other.

## 4 Algorithms/Methods

In this project, we will implement two popular object detection algorithms, YOLOv3 and SSD. Both YOLOv3 and SSD are improved algorithms based on YOLO. The principle of YOLO is to divide every image to a grid of S x S and then produce N bounding boxes and confidence for each grid. The confidence indicates the precision of the bounding box and whether the bounding box actually contains objects that define the class. The architecture of YOLOv2 is similar to YOLO, but compared

to YOLO, YOLOv2 is better and faster. YOLOv3, as the upgraded version of YOLOv2, has an incremental improvement from YOLOv2.

Similar to YOLOv2, instead of using dropout, YOLOv3 applies batch normalization to eliminate the overfitting problem. Moreover, a higher resolution classifier is used to train the model on the ImageNet dataset, which increases the accuracy of the model. However, compared to YOLOv2 who uses Darknet-19 which contains 19 convolutional layers, 5 max pooling layers, and 1 average pooling layer as the base network. YOLOv3 uses Darknet-53 which removes max pooling layers and contains more convolutional layers. Borrowing ideas from Faster R-CNN, YOLOv2 uses anchor boxes as well. However, in Faster R-CNN the anchor boxes are designed manually, while in YOLOv2 they are generated by K-means algorithm. This feature is kept in YOLOv3. Unlike YOLOv2, YOLOv3 uses Resnet in the residual net, which can improve the performance of extracting features from objects. In addition, YOLOv3 performs detection on 3 scales, detecting large, medium and small objects in sequence, which improves the accuracy of object detection.

The framework of SSD is similar to YOLOv3. SSD also has anchor boxes and replaces the fully connected layers by convolutional layers. However, different from YOLOv3 who uses Darknet-53 as the backbone, SSD is built based on VGG-16 (although variants exist). Additionally, SSD takes different sizes of feature maps to do the detection. The bigger feature maps which are extracted from the front convolutional layers can be used to detect small objects, and the small feature maps which are extracted from the later convolutional layers can be used to detect large objects.

In this project, the goal is to compare the performance of YOLOv3 and SSD, so they use the same data pipeline to preprocess the dataset. The original resolution is 1280x720 and we need to apply resizing. We chose 300x300 for the SSD model as it is one of the most popular resolutions. The structure of YOLOv3 requires the size to be a multiple of 32, so we selected 320x320 for a fair comparison with SSD. After resizing, the same augmentations techniques are applied to these two models. They include random size cropping, hue saturation, gaussian blur, and random brightness contrast. Each action has 0.5 probability of being performed, independent of other actions.

The dataset contains 79,853 images with labels. We choose 10,000 of them as the test set to evaluate the final performance of the two models. Therefore, the training set contains 69,853 images. To monitor the performance of models in each epoch, we set aside 10% of training images as the validation set, to indicate the performance change of models during training.

## 5 Results/Comparisons

### 5.1 Inference Speed

For testing, we used a batch size of 1, because in actual driving, images need to be processed in real time. The CPU used was Intel(R) Xeon(R) Silver 4208 CPU @ 2.10GHz, and the GPU was GeForce_RTX_2080 (CUDA version 11.2). To assess a model's inference speed, we add up the time used to feed images through the model, and the time to extract prediction boxes and classes.

In test time, YOLOv3 is able to process 77 images per second, and SSD processes 86 images per second. Although SSD has a higher FPS, it is important to note that 1) image size for YOLOv3 is slightly larger, and 2) YOLOv3 predicted more objects on average (4.3026 v.s. 3.9348). Therefore it is safe to say that the two models have very similar inference speed. Note that although these speed are good enough for real-time inference, we are using images of reduced resolution. Whether we can still get reasonable speed at finer resolution will require further experiments.

### 5.2 Convergence in Training

Both models applied a stepwise learning rate schedule, in which learning rate is decreased as a step function of the number of batches. We adopt the stepwise schedule instead of a continuous decay strategy, because the former allows more flexible control over the learning rate (and we can make adjustments as we monitor the training progress).

In terms of validation mAP, we found the SSD model to converge faster than YOLOv3. For SSD, validation mAP stabilizes after 40 epochs, and the learning rate decay appears to have little effect. Unlike SSD, YOLOv3 needs the learning rate change to continue improving after 20 or so epochs. Validation mAP stabilizes after 70-80 epochs. Figure 3 and 4 in the appendix contain more details.

### 5.3 Inference Results

Average Precision is calculated at IOU = 0.45. While this deviates from the more standard metrics of $AP^{0.5}$ and $AP^{0.50:0.05:0.95}$, for the purpose of comparison such deviation is fine.

SSD is more consistent across the classes, achieving AP of at least 0.15 for most classes (except a few severely under-represented ones). On the other hand, YOLOv3 achieves much better performance on classes that are more common (cars, traffic lights, traffic signs and pedestrians), but are much worse on less common ones (0.28 lower on buses, 0.08 lower on trucks and bikes, and almost completely unable to make any correct prediction on motorcycles or riders). Therefore, while YOLOv3 is able to get better results for more common objects, SSDs are more robust to data imbalance. See appendix 3 table 1 for more detail.

We also calculated a "weighted" mAP for each model, where "weighted" means we multiply the AP of each class by its share of percentage in the dataset. The weighted mAP for SSD is 0.3619, while the weighted mAP for YOLOv3 is 0.4614.

While past YOLOs struggle with small objects, Redmon and Farhadi claim that YOLOv3 have improved significantly for small objects [3]. The standard definition of "small object" is size < $32^2$, which is roughly 1% of our image areas. The category with the smallest objects in this dataset is "traffic light", each one occupying 0.05% of an image's area, on average. YOLOv3 indeed did reasonably well with them, obtaining an AP higher than that of SSD by 0.116.

We also segmented the test corpus into different scenarios (e.g. rainy vs sunny, daytime vs night) and tested our models on them. For details please see appendix 3 table 2. We found SSD to be more variable under different scenarios but YOLO is more consistent. For example, on frames with at most 8 objects, SSD's mAP is 0.07 higher than frames with at least 25 objects. In the YOLOv3 case, the difference is less than 0.02. Surprisingly, YOLOv3 is better at working with nighttime frames than daytime frames.

## 6 Future Work

Class imbalance resulted in catastrophic performance for several under-represented classes, and class balancing techniques such as resampling and/or reweighting cost function might help. In a recent paper, (Ghiasi et al) suggest that simply copying and pasting segmented objects in the data augmentation phase also helps a lot. BDD100K provides a sample of 10K fine instance segmentation frames, and we can use the under-represented objects in these frames to apply copy pasting.

Another cause for our relatively low mAP is the small size of the images (300 for SSD and 320 for YOLOv3), as a result of time constraint. In the domain of object detection for autonomous driving, we would encounter many small objects (signs, traffic lights, fastly-moving cars, etc) and accuracy on these objects is key. Training with larger image sizes will help mitigate the problem.

## 7 Conclusion

This project reimplements two popular object detection algorithms, YOLOv3 and SSD, and applies these two algorithms to a customized dataset, BDD100K, to compare their contributions to autonomous driving. In order to make a fair comparison, the same data preprocessing and evaluation pipeline are applied. By comparing their performance on the test set, we discovered SSD to be more consistent and more robust to data imbalance, while YOLOv3 is better at detecting more common objects, such as cars, traffic lights, traffic signs and pedestrians. We also found YOLOv3 to be more consistent under different environment scenarios. Unlike previous version of YOLO, YOLOv3 no longer suffers with small objects. Inference speed of the two models are comparable, with SSD processing 86 FPS, and YOLOv3 processing 77 FPS at a slightly higher resolution.

Although the accuracy of YOLOv3 and SSD on this dataset falls short of the requirement for autonomous driving, our results demonstrates the advantages and disadvantages of these two models on object detection. It also provides a reference for which object detection algorithms to choose from in further research.

## References


[1] H. Yin and C. Berger. (2017) When to use what data set for your self-driving car algorithm: An overview of publicly available driving datasets. *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Yokohama, pp. 1-8, doi: 10.1109/ITSC.2017.8317828.

[2] Yu, Fisher et al. (2020) BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2633-2642.

[3] Joseph, Redmon & Ali, Farhadi. (2018) YOLOv3: An Incremental Improvement. *CoRR*.

[4] Rowley, Henry & Baluja, Shumeet & Kanade, Takeo. (1996). Neural Network-Based Face Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**:203-208.

[5] Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet classification with deep convolutional neural networks. *In NIPS*, pp. 1097–1105.

[6] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *In CVPR*, pp. 580–587.

[7] J. Uijlings, K. van de Sande, T. Gevers, & A. Smeulders. (2013) Selective search for object recognition. *IJCV*.

[8] Girshick, R. (2015). Fast R-CNN. *In ICCV*, pp. 1440–1448.

[9] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real time object detection with region proposal networks. *In NIPS*, pp. 91–99.

[10] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real time object detection. *In CVPR*, pp. 779–788.

[11] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C., & Berg, A. (2016). SSD: Single shot multibox detector. *In ECCV*, pp. 21–37.

[12] Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. *In CVPR*.

[13] Joel Janai, Fatma Güney, Aseem Behl, & Andreas Geiger (2020), Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art, *Foundations and Trends® in Computer Graphics and Vision*: **12**(1-3): pp. 1-308.

[14] Heechul Jung, Min-Kook Choi, Jihun Jung, Jin-Hee Lee, Soon Kwon, & Woo Young Jung. (2017) *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 61-67.

[15] A. Geiger, P. Lenz, & R. Urtasun. (2012) Are we ready for autonomous driving? The KITTI vision benchmark suite. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354-3361.

[16] P. Dollár, C. Wojek, B. Schiele, & P. Perona. (2009) Pedestrian Detection: A Benchmark, *CVPR*

[17] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, & Oscar Beijbom. (2020) *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11621-11631.

[18] Sagar Vinodababu, a-PyTorch-Tutorial-to-Object-Detection, GitHub repository, `https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Object-Detection`

[19] Erik Linder-Noren et al., PyTorch-YOLOv3, GitHub repository, `https://github.com/eriklindernoren/PyTorch-YOLOv3`

## Appendix

Table 1: Categories of objects present in BDD100K dataset

| Class | Count | Percent | Average area of object (as a % of image area) |
|---|---|---|---|
| traffic light | 214755 | 14.7009 | 0.0518 |
| traffic sign | 272994 | 18.6877 | 0.1261 |
| car | 803540 | 55.0059 | 1.0426 |
| pedestrian | 105584 | 7.2277 | 0.31389 |
| bus | 13637 | 0.9335 | 3.7667 |
| truck | 32135 | 2.1998 | 2.8539 |
| rider | 5218 | 0.3572 | 0.6756 |
| bicycle | 8163 | 0.5588 | 0.6433 |
| motorcycle | 3483 | 0.2384 | 0.8426 |
| train | 143 | 0.0098 | 4.2050 |
| other vehicle | 889 | 0.0609 | 3.2516 |
| other person | 211 | 0.0144 | 0.2525 |
| trailer | 73 | 0.0050 | 3.7944 |
| all | 1460825 | 100.0000 | 0.7359 |

Table 2: mAP of SSD v.s. YOLOv3 on test, under different scenarios (each set contain 500 frames)

| | SSD | YOLOv3 |
|---|---|---|
| Clear Test | 0.169 | 0.158 |
| Rainy Test | 0.193 | 0.152 |
| Daytime Test | 0.197 | 0.142 |
| Night Test | 0.161 | 0.159 |
| Dawn/Dusk Test | 0.166 | 0.147 |
| Few Objects Test | 0.237 | 0.171 |
| Many Objects Test | 0.166 | 0.155 |
| Overall Test | 0.182 | 0.154 |

Table 3: AP of each class on test; SSD v.s. YOLOv3

| | SSD | YOLOv3 |
|---|---|---|
| Bicycle | 0.19613 | 0.11574 |
| Bus | 0.38402 | 0.10423 |
| Car | 0.49737 | 0.60625 |
| Motorcycle | 0.16528 | 0.00033 |
| Other person | 0.0 | 0.0 |
| Other vehicle | 0.02373 | 0.0 |
| Pedestrian | 0.18906 | 0.27137 |
| Rider | 0.15630 | 0.00147 |
| Traffic Light | 0.16273 | 0.27863 |
| Traffic Sign | 0.19491 | 0.31502 |
| Trailer | 0.0 | 0.0 |
| Train | 0.0 | 0.0 |
| Truck | 0.39383 | 0.31335 |

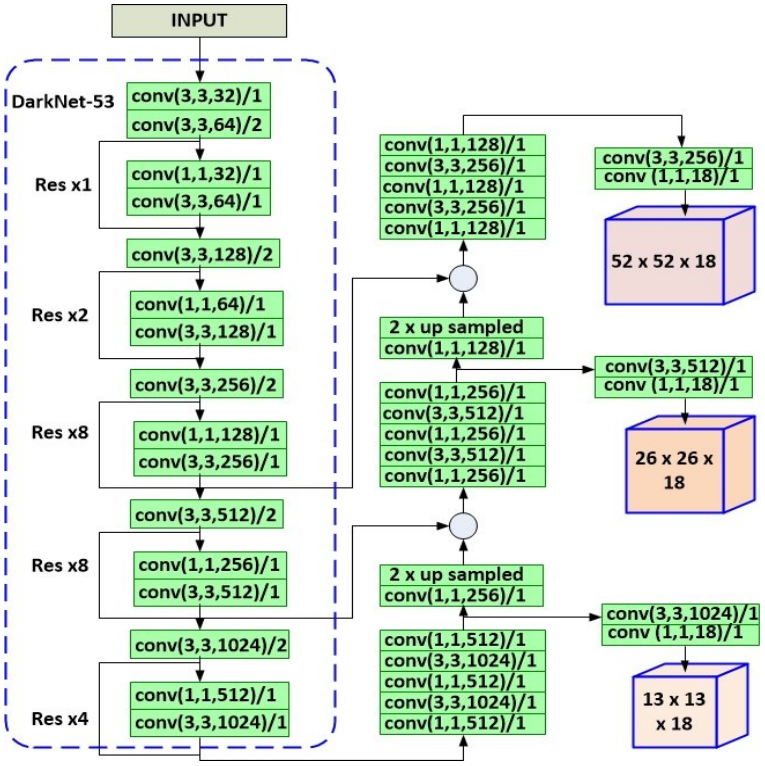


Figure 1: Architecture of YOLOv3

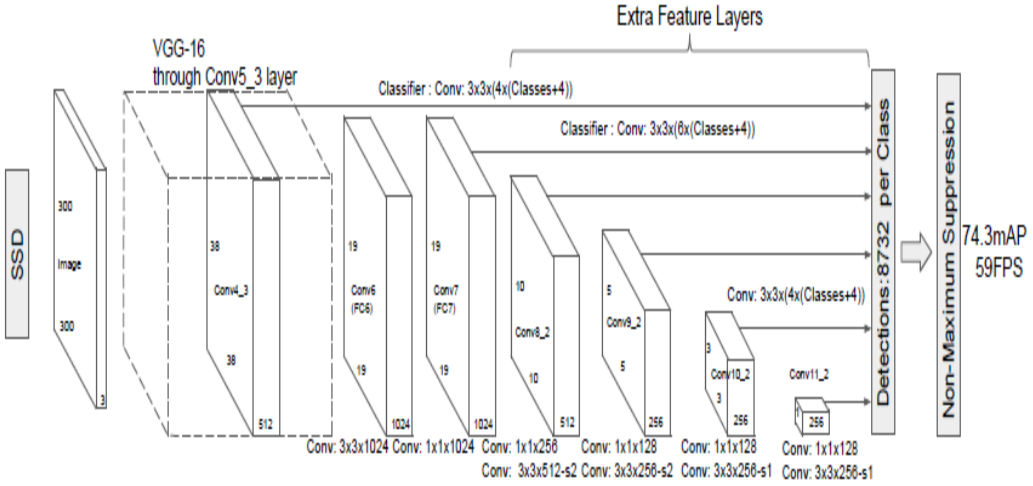


Figure 2: Architecture of SSD

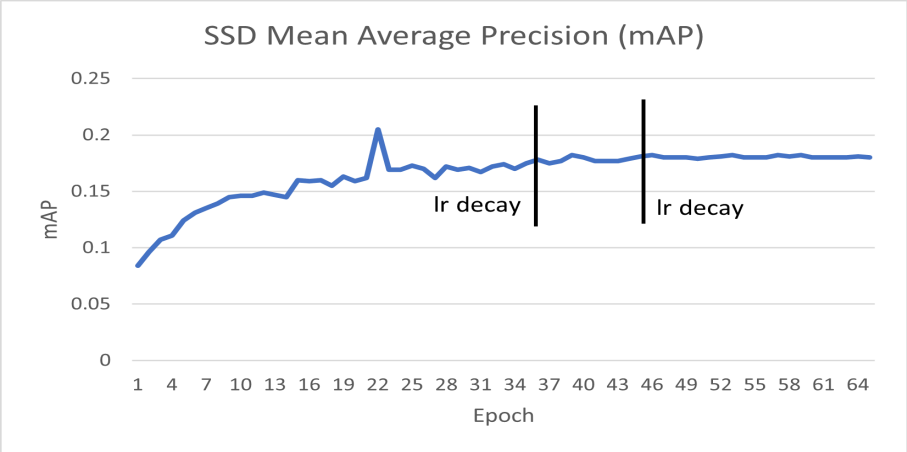


Figure 3: SSD validation mAP

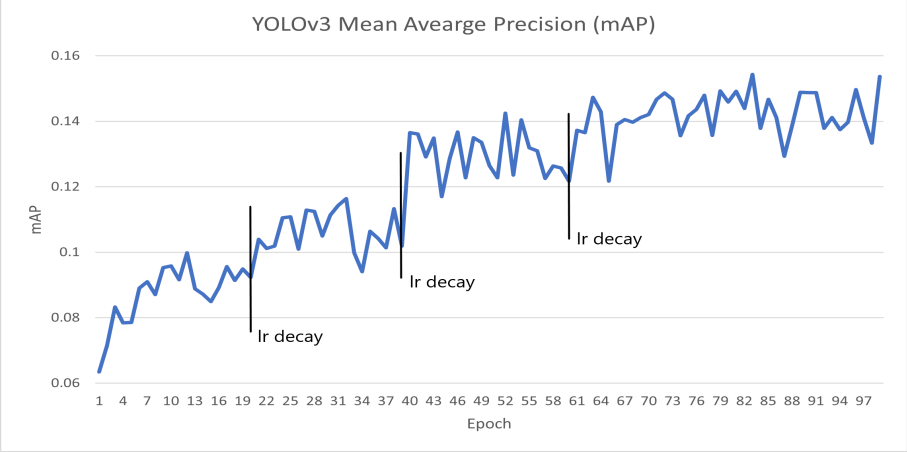


Figure 4: YOLOv3 validation mAP


Figure 5: Ground truth vs. SSD vs. YOLOv3

Our code repo is hosted on github (`https://github.com/jiawei-yu-97/YOLO_and_SSD_on_BDD100K`)