Evaluating YOLOv3 Training Methods for Pedestrian Detection

Anonymous CVPR 2021 submission

Paper ID ****

Abstract

Pedestrian detectors need to be accurate and available for edge computing applications such as autonomous driving. We developed a pedestrian detector using the YOLOv3-tiny (You Only Look Once) open-source real-time object detection algorithm. We selected YOLOv3-tiny because the smaller model is a good fit for edge computing applications. We investigated how to improve image detection using pedestrians. Detectors have been empirically researched and the best results are found to use a diverse combination of approaches. We took the YOLO object detection model, applied it to different pedestrian detection datasets, and tuned hyperparameters. We ran individual experiments adding batch normalization, using the Adam learning rate optimization algorithm, improving anchor selection, fine tuning, and then combining these approaches in one diverse run.

1. Introduction

Pedestrian detection is a subset of object detection. Object detection is a computer vision task of recognizing and classifying objects in still images. Object detection can be accomplished using neural networks. A detector is not the same as a tracker. A tracker is given a video as an input and A) tracks a given object or B) looks at the video and detects what's moving. In keeping with the Pareto principle, a tracker or detector can be developed to track a myriad of objects or development efforts can be focused to track just pedestrians well. Our semester goal was to develop a pedestrian detector on still images. Our educational objective was to learn the core technology and theory of pedestrian detection. We investigated how to improve image detection using pedestrians. We took the YOLO object detection model, applied it to different pedestrian detection datasets, and changed hyperparameters.

Pedestrian detection has many real world applications, from autonomous vehicles to urban planning. There has been extensive research into solutions for this problem, with recent breakthroughs coming from the application of convolutional neural networks. In this project, we examine pedestrian detection using images. Our approach was to use a popular object detection model, YOLOv3-tiny, and experiment with methods to improve its performance. We used YOLOv3-tiny for its balance of performance and training/inference speed. Our goal was to investigate methods to improve performance using popular methods such as fine tuning, using the Adam optimizer, adding BatchNorm layers, and calculating anchors specific to the dataset.

We used the CityPersons dataset for training. This dataset is around 10 GB. We used the Penn-Fudan dataset, and measured the F1 score, the precision and recall curves.

2. Motivation

Pedestrian detection is an exciting and growing research area. In 2013, there was a renewed interest in this problem. In 2014, the automobile manufacturer Tesla introduced their autopilot hardware suite. Also in 2014, half of research submissions that use the KITTI dataset were submitted. The KITTI dataset is connected to Toyota Motor Corporation. Daimler, the parent company of Mercedes-Benz, is connected to two other popular pedestrian detection datasets. This area is actively growing both in academia and industry.

Classification inputs an image with one or more objects and outputs one or more bounding boxes. These bounding boxes are defined by a point, width, and height. Object detection is classification plus localization. Object detection is to input an image with one or more objects and output one or more bounding boxes and a class label for each bounding box. For our project, these objects are pedestrians. Pedestrian detection is on still images and pedestrian tracking is on video.

There are many exciting research problems in this topic area including: general detection, general tracking, pedestrian detection, pedestrian tracking, and pedestrian re-identification. Person re-identification in deep learning is when images or videos taken from multiple angles are classified as the same person [1].

Our goal was to produce a model that can be accurate, but also fast and light enough to be used for edge computing and real time detection. To that end, we examined methods to improve existing small networks without sacrificing speed.

3. Related Work

Pedestrian detection has various applications including car safety, self-driving cars, surveillance, and robotics. The advantages include this being a well-defined problem and having established benchmarks and datasets. This area benefits from empirically tested approaches, a pattern analogous to the broader field of neural networks. For example, the reasons why batch normalization works are under discussion and challenged, but the benefits of batch normalization are agreed on. We overall know that the empirically tested approaches work.

3.1. Public Pedestrian Datasets

There are a variety of public pedestrian datasets. As one of the oldest datasets, the INRIA Person dataset has relatively few images and does not contain video. One advantage of this dataset is it has a diversity of geographical settings like the city, beach, and mountains while also having high quality annotations. The ETH Pedestrian dataset is a mid-sized video dataset from ETH Zurich that provides stereo information. The TUD-Brussels dataset is a mid-sized video dataset. Daimler AG, the parent company of Mercedes-Benz, has both the Daimler and Daimler Stereo datasets. The Daimler dataset lacks color channels while the Daimler Stereo dataset provides stereo information. The KITTI dataset, from the Karlsruhe Institute of Technology and Toyota Technological Institute, is a large dataset that provides stereo information. The KITTI dataset has become a benchmark in pedestrian detection. A distinguishing feature of this dataset is that the test set is more diverse, but also has yet to be commonly used. The Caltech-USA dataset is one of the larger and more intensive datasets [2]. However, this dataset has also set itself apart as a benchmark in pedestrian detection. Furthermore, this dataset has been evaluated numerous times by researchers using the concept of model ensemble. The concept of model ensemble is combining a diverse set of approaches to achieve the best overall performance. In practice, this diverse approach is what many cutting edge researchers have used.

3.2. Families of Pedestrian Detectors

There are three families of pedestrian detectors. They are deformable parts model (DPM), deep networks (DN), and decision forests (DF) [3]. DPM detectors were built to solve pedestrian detection problems. They are advantageous at overcoming occlusion problems. Recent research has questioned the need for parts, and instead investigated using deep networks. A convolutional neural network (CNN) is a deep architecture example. Deep architectures benefit from the availability of large datasets and growing computational power. Deep architectures can be applied to a wide range of problems, including semantic labelling, classification, and detection. Researchers were successful in training a CNN on the INRIA dataset, and applying that model to the INRIA, ETH, and TUD-Brussels datasets. Unfortunately, when they tried testing on the Caltech dataset, they were unsuccessful. Researchers did not report evidence that deep networks are necessarily better at learning features for pedestrian detection [3]. Future areas of improvement include optimizing the core algorithm and expanding the variety of techniques used inside the model.

3.3. YOLO Neural Networks

Current research on object detection models have been mostly focused around deep convolutional neural networks. Early networks focused on proposing regions of interest, then using a CNN to detect objects in these regions [4]. But these networks were slow and had multiple disjoint parts that we could not perform gradient descent on. Improvements to these networks centered around creating an end-to-end training pipeline with gradient descent throughout [5, 6]. However, these networks are still too slow for real time detection. The popular method for real-time object detection currently is the YOLO series of networks. These networks look at the entire image, and use a CNN to predict the bounding boxes for objects. Each new version of the YOLO architecture brings further improvements, including BatchNorm [7], residual networks [8], and anchor boxes.

Our work is based on the most widely used version of YOLO, called YOLOv3 [9, 10, 11]. Specifically, we chose the YOLOv3-tiny architecture for fast training and inference time [12].

4. Approach

Our approach is to use the YOLO object detection model as a reference point [13], train it on the CityPersons dataset [14], and tune hyperparameters. We created a baseline using pretrained weights. We also trained from scratch to compare the impact the pretrained weights had on training. We ran individual experiments adding batch normalization, using the Adam optimization algorithm, improving anchor selection, and adding fine tuning. Finally, we combined these approaches in one run. We trained on the CityPersons dataset. We tested on the Penn-Fudan dataset [15].

The YOLOv3-tiny model's shortcomings are lower overall performance compared to the much bigger YOLOv3 model. The former is roughly 30% of the latter's performance on the mAP metric. However, it makes up for this by being lighter and faster during both training and inference.

4.1. Baseline

For our baseline, we chose to use a pretrained YOLOv3-tiny model, trained on the MS COCO dataset for 300 epochs. Our assumption was the tasks were similar enough (object detection to pedestrian detection) that the checkpoint would provide a good starting point. We also didn't have much computing resources for a task of this type, so we decided to leverage pre-existing computing by using this checkpoint. To standardize our results, we ran everything for 30 more epochs. To train from scratch, we trained on the CityPersons dataset with initial weights uniformly sampled from a random distribution.

4.2. Architecture Modifications

In one experiment we modified the YOLOv3-tiny architecture to add batch normalization layers. Batch normalization is a technique discovered in 2015 by Sergey Ioffe and Christian Szegedy that helps solve the problem of vanishing and exploding gradients [7]. The innovation behind batch normalization is that the technique re-centers and re-scales the inputs at each layer.

4.3. Experiments with Hyperparameters

In one of our experiments, we trained our YOLOv3-tiny model using Adam. The Adam optimizer should allow us to converge to the minima more quickly. In another experiment, we chose to reduce the learning rate whilst keeping the pretrained weights, under the hypothesis that this is a fine-tuning problem, which usually performs better with a lower learning rate. We also did an experiment with increased learning rate, with the hypothesis that if the pretrained weights are in a local minima, this allows it to get past it. To investigate the impact of anchors on performance, we decided to find the best anchors for our current dataset. We did this by computing all the bounding boxes, then using K-means clustering to calculate the cluster centroids, and used those centroids as the anchors. To verify without training the model, we calculated the average IoU of the default anchors versus the new ones. This gave us an improvement from 52% to 72%. We then trained the model using these new anchors.

4.4. Combined Run

After running individual experiments, we combined our methods into one model. We added batch normalization, used the Adam optimization algorithm, and improved anchor selection. This allows us to measure the effects of using all of these methods at once.

4.5. Pedestrian Detection Metrics

For pedestrian detection metrics, we chose to measure the F1 scores, precision/recall curves, and mean Average Precision (mAP) scores. We chose these because they are technically sufficient and well-established metrics in this field. Recall is a measure of detecting positive matches.

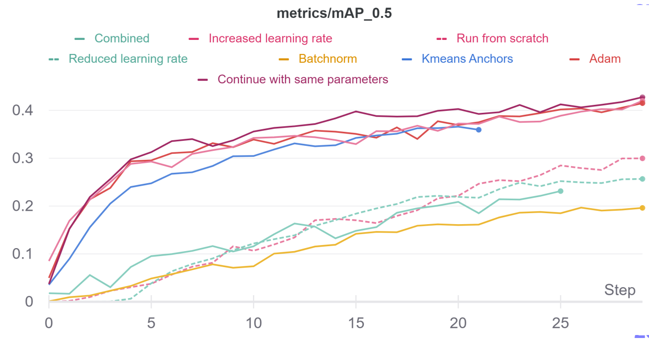


Figure 1: Mean Average Precision over 30 Epochs.

$$P = \frac{TP}{TP + FP} \quad (1)$$

Precision is calculated using Equation 1 where P stands for Precision, TP stands for true positive, and FP stands for false positive.

$$R = \frac{TP}{TP + FN} \quad (2)$$

Recall is calculated using Equation 2 where R stands for Recall, TP stands for true positive, and FN stands for false negative.

$$\text{F1 score} = 2 \cdot \frac{P \cdot R}{P + R} \quad (3)$$

Equation 3 shows how to calculate F1 score. Equation 4 shows how to calculate Intersection over Union (IoU) where Intersection stands for the area of the overlapping regions, and Union stands for the total area covered by either the ground-truth or the predicted bounding box.

$$IoU = \frac{Intersection}{Union} \quad (4)$$

An image has both a ground-truth and a predicted bounding box. A ground-truth is preset by a human based on the human's assessment and a predicted bounding box is what the network outputs. Mean Average Precision is a metric used in object detection. To determine a mAP score, we compare the ground-truths to the predicted bounding boxes. A higher mAP score means you have a more accurate pedestrian detection model.

5. Evaluation

For the Mean Average Precision (Figure 1), we observed that continuing training on the new dataset with the same parameters produced the best result, while adding

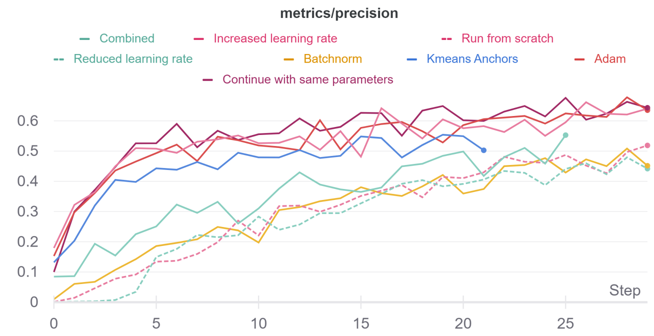


Figure 2: Precision over 30 Epochs.

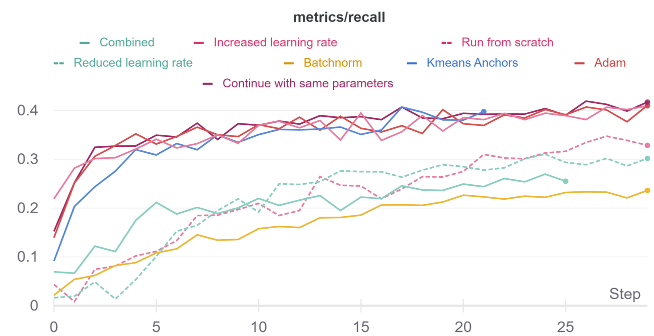


Figure 3: Recall over 30 Epochs.

BatchNorm produced the worst average precision. We also observed a clear separation between groups of methods, with Adam optimizer, continued training, increased learning rate, and K-means anchors in one group, significantly above the other group. One surprising result was that training from randomly initialized weights, though it performed worse than some, was outperforming other methods. It was interesting to observe that having a reduced learning rate with pretrained weights performed was worse than training from scratch. The same results were observed for the Precision metrics, although the gap between groups were much smaller (Figure 2).

We also had similar performance in Recall (Figure 3), although a difference was the K-means anchors appeared to begin to outperform every other methods, but the run was closed by Google Colab due to exceeding resource usage limits before it could finish.

For the F1 score, we observed that starting from pretrained weights resulted in a significantly better F1 curve than running from scratch (Figure 4). A similar result was observed for the Precision/Recall curves, where training from pretrained weights produced a significantly better curve (Figure 5).

The most unexpected result for us was the poor performance of the reduced learning rate. While the performance of the training from baseline weights being significantly above random weights suggested that the tasks are related enough that knowledge from one task can be transferred to the other, the poor performance of the reduced learning rate suggests that pedestrian detection is perhaps not a fine-tuning of object detection.

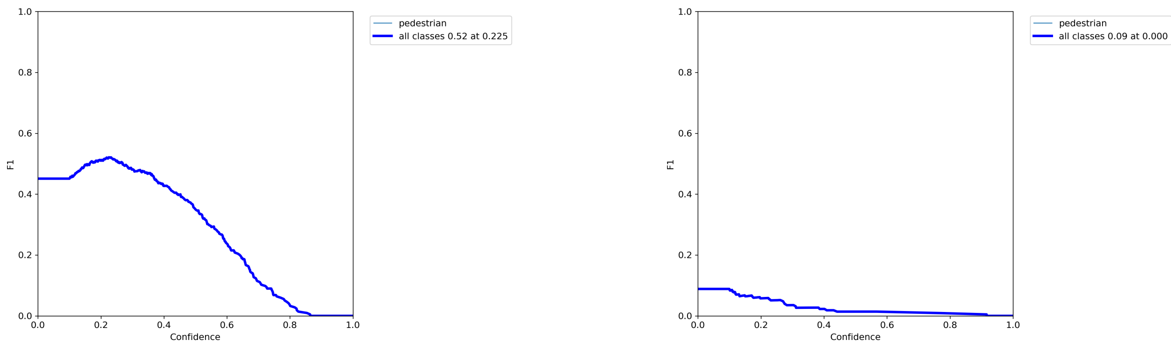
Another observation we made was comparing the performance before and after training. We observed that before training, the network gave much more confident predictions, but had difficulties identifying multiple pedestrians close together. After training, the confidence became much lower, but the performance on groups of close pedestrians improved (Figure 6). Finally, we attribute the poor performance of BatchNorm to the pretrained weights not being trained on BatchNorm before, which means their range of expected values could be very different from the value BatchNorm outputs, leading to poor performance.

6. Conclusion

We found that pedestrian detectors can be engineered using tiny neural networks models while still maintaining accuracy. We achieved a precision of 0.64 and recall of 0.4. The detection time was 20ms per image. The training time for each methods was similar, at around 5 hours. We've experimented with various popular methods of changing hyperparameters for better results, such as the Adam optimizer, K-means anchors, and changing learning rates for fine-tuning. We observed that the single most influential method was to initialize training on pretrained weights, as opposed to random weights. We also observed that it is not better to reduce the learning rate, which is one method of fine-tuning, but to keep the same learning rate. A final observation is that the weights for MS COCO were very good for pedestrian detection, but struggle to correctly identify multiple pedestrians in one area, which is perhaps a limitation of the dataset not having that type of data. For future work, we hope to further study other methods of adapting YOLOv3-tiny for pedestrian detection, and study whether these results hold true for longer runs. Frankly, we did not expect just training from pretrained weights with unchanged hyperparameters to be the best performing method. However, that gives us insight into whether changing hyperparameters mid-run will actually give worse or similar results.

7. Acknowledgements

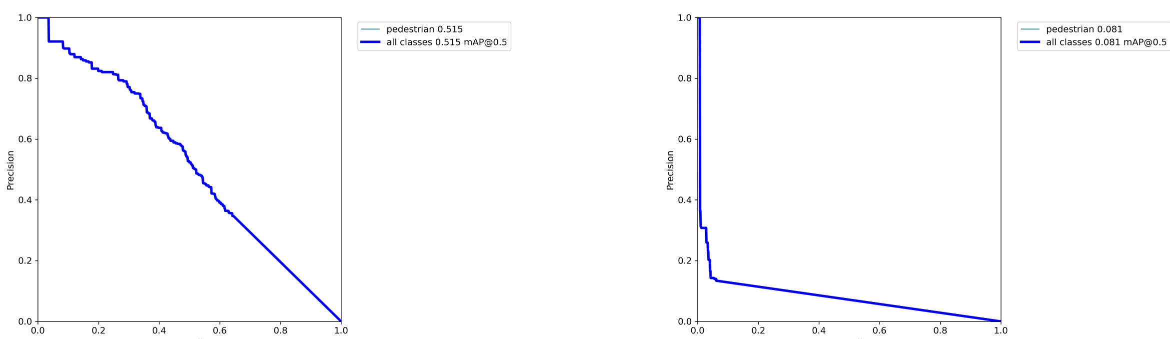
We would like to thank Professor Erik Learned-Miller, Ankita Rajaram Naik, Eddie Cunningham, and Yunda Liu of the University of Massachusetts Amherst for helping us learn and grow our understanding of neural networks. We would like to thank Shanshan Zhang, Rodrigo Benenson, and Bernt Schiele of the Max Planck Institute for Informatics for the CityPersons dataset.



(a) Pretrained weights

(b) Random weights

Figure 4: F1 scores between pretrained weights and random weights.



(a) Pretrained weights

(b) Random weights

Figure 5: Precision/Recall curves between pretrained weights and random weights.

Figure 6: Detection after training (above) vs Detection before training (below). Both had confidence threshold set to 0.01. After training, the model could better detect groups of people.

References

[1] L. Wu, C. Shen, and A.V.D. Hengel, ”PersonNet: Person Re-identification with Deep Convolutional Neural Networks,” arXiv preprint arXiv:1601.07255, 2016. 1

[2] P. Dollar, C. Wojek, B. Schiele, and P. Perona, “Pedestrian detection: A benchmark,” 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009. 2

[3] R. Benenson, M. Omran, J. Hosang, and B. Schiele, “Ten years of pedestrian detection, what have we learned?,” Computer Vision - ECCV 2014 Workshops, pp. 613–627, 2015. 2

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, ”Rich feature hierarchies for accurate object detection and semantic segmentation,” In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 580-587. 2

[5] R. Girshick, ”Fast R-CNN,” In Proceedings of the IEEE international conference on Computer Vision, 2015, pp. 1440-1448. 2

[6] S. Ren, K. He, R. Girshick. and J. Sun, ”Faster R-CNN: Towards real-time object detection with region proposal networks,” Advances in neural information processing systems 28, 2015, pp. 91-99. 2

[7] S. Ioffe, C. Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” ICML’15: Proceedings of the 32nd International Conference on International Conference on Machine Learning, Volume 37, 2015, pp. 448–456. 2, 3

[8] K. He, X. Zhang, S. Ren and J. Sun, ”Deep Residual Learning for Image Recognition,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778. 2

[9] P. Adarsh, P. Rathi, and M. Kumar, “YOLO v3-Tiny: Object Detection and recognition using one stage improved model,” 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), 2020. 2

[10] W. Lan, J. Dang, Y. Wang, and S. Wang, “Pedestrian detection based on Yolo Network Model,” 2018 IEEE International Conference on Mechatronics and Automation (ICMA), 2018. 2

[11] Long, Xiang et al., “PP-YOLO: An Effective and Efficient Implementation of Object Detector,” 2020. 2

[12] Z. Yi, S. Yongliang, and Z. Jun, “An improved tiny-yolov3 pedestrian detection algorithm,” Optik, vol. 183, pp. 17–23, 2019. 2

[13] YOLOv3-tiny. (2021), Ultralytics. Accessed: December 1, 2021. [Online]. Available: https://github.com/ultralytics/yolov3 2

[14] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The Cityscapes Dataset for Semantic Urban Scene Understanding,” in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. 2

[15] Penn-Fudan Database for Pedestrian Detection and Segmentation. (2007), University of Pennsylvania. Accessed: December 1, 2021. [Online]. Available: https://www.cis.upenn.edu/ jshi/ 2