Exploring Iterative Pruning in Deep Convolutional Neural Networks

Brian Jackson - bjack205@stanford.edu

Motivation

Modern state-of-the-art image detection networks like YOLOv2 achieve impressive results in terms of accuracy and speed [9]. These achievements, however, typically come with fairly significant computational expense and resource requirements, just as high-end GPUs [4,7-9]. Compressing neural networks has drawn more and more attention as industry pushes these networks to more computationally-limited platforms[1,5,6]. Recent work here at Stanford has shown that state-of-the-art networks like SSD can be iteratively "pruned" to about 90% of the original size, without significant drops in accuracy. This project aimed to repeat this on the YOLOv2 framework, after retraining it on the KITTI data set, a realistic autonomous driving data set.

Objectives

1. Retrain YOLOv2 on the KITTI data set
2. Provide a framework to simplify the training process and reduce the barrier to entry for using yad2k, a Keras/Tensorflow implementation of YOLOv2
3. Investigate the effect of iterative pruning on the performance of YOLOv2

Dataset

The Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago (KITTI) Vision Benchmarking Suite is an impressive data set for benchmarking algorithms for autonomous driving [2,3].

This project used the 2D object detection dataset, consisting of about 7500 images for training and 7500 images for testing. In addition to bounding boxes classification labels, the labels also include details about the occlusion, truncation, and orientation for each object. There are 9 classification categories in KITTI: Car, Van, Truck, Pedestrian, Person_sitting, Cyclist, Tram, Misc, and DontCare. For this project, 1000 images were used for training and 100 validation.

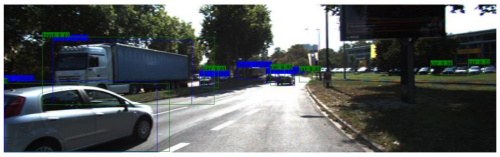
Results

Retraining the Network– 3 different approaches were taken when retraining the network. The initial approach was to re-use the classes from COCO, since the most important KITTI classes were also included in COCO. The network was first fine-tuned (Coco-FT) for 5 epochs (before the dev loss started increasing) by freezing all but the last convolutional layer. It was then fully retrained for 50 epochs (Coco-Full). The last approach was to fine-tune the last layer using the KITTI classes, so the last layer had to be randomly initialized. This model (KITTI-FT) was trained for 500 epochs. The mAP and recall values are shown in the table to the right.

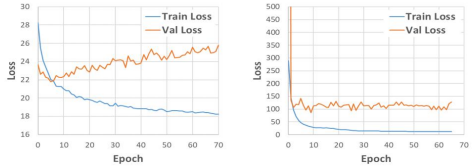
Pruning the Network – Pruning was set to remove the bottom 2% of weights (setting them to 0), and subsequently retrain the network. All attempts so far resulted in poor detections.

	Person	Cyclist	Car	Train	Truck
Original	0.37 / 0.47	0.00 / 0.00	0.44 / 0.60	0.00 / 0.00	0.21 / 1.00
Coco-FT	**0.70** / **0.53**	0.20 / 0.11	0.65 / 0.82	**1.00** / **0.11**	**1.00** / **0.67**
Coco-Full	0.47 / 0.33	**1.00** / **0.23**	0.70 / **0.83**	0.00 / 0.00	**1.00** / 0.23
KITTI-FT	0.67 / 0.18	0.00 / 0.00	**0.97** / 0.53	0.00 / 0.00	0.83 / 0.36

(Above) mAP / Recall values for 5 classes for different training approaches. No single approach showed the best results overall, and fully-training the network did not seem to help in most cases.



(Top) Sample detections on the KITTI test set. Green and blue boxes represent predicted and actual detections. (Bottom) The training and dev losses for Coco-FT and Coco-Full, shown on the left and right, respectively. The weights with the best dev-loss were used.



The Model

YOLOv2 is currently state-of-the-art in image detection, achieving 78.6% mAP on PASCAL VOC and 48.1% on COCO test-dev, with an impressive detection rate of 40-90 FPS.

This project used YAD2K [10], a popular Keras/Tensorflow implementation of YOLOv2, since YOLO was developed in Darknet, a neural network framework designed by the authors of the paper. The original weights trained on the COCO data set were used as a starting point.

Discussion

Since the YAD2K implementation proved to be difficult to work with due to poor structure, assumptions, and documentation, a very large portion of this project was allocated to developing tools to train a custom dataset within the YAD2K framework. The new framework was successfully used to retrain model using both COCO labels and KITTI labels. Attempts to prune the network by eliminating weights have been unsuccessful so far.

Future Steps

Future work includes retraining the network on the entire KITTI dataset, and including the additional label data to achieve better performance on dev and test sets. The effect of pruning the network can also be evaluated. After pruning, it should be instructive to investigate which weights remain and propose new network topologies based off these remaining weights.

References

1. V. Oseledets, "Tensor-Train Decomposition". *SIAM J. Scientific Computing*, vol. 33, no. 5, pp. 1948-1974, 2011
2. A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite", in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012
3. A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset", *International Journal of Robotics Research (IJRR)*, 2013
4. R. Girshick, "Fast C-NN", *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 Inter, pp. 1440-1448, 2015
5. S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding", pp1-14, 2015
6. A. Novikov, D. Pdoprikhin, A. Osokin, and D. Vetrov, "Tensorizing Neural Networks", pp. 1-9, 2015
7. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", 2015
8. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector", *Lecture Notes in Computer Science*, vol. 9905 LNCS, pp. 21-37, 2016
9. J. Redmon and A. Farhadi. "YOLO9000: Better, Faster, Stronger", 2016
10. Allanzelener, yad2k, https://github.com/allanzelener/YAD2K.git