Effects of Gradient Optimizer on Model Pruning

Yang Zhang\textsuperscript{1, a}, Bin Zhu\textsuperscript{1, b}, Qi Ma\textsuperscript{1, c}, Huanan Wang\textsuperscript{2, d}

\textsuperscript{1}State Key Laboratory of Pulsed Power Laser Technology, National University of Defense Technology, Hefei, Anhui, China
\textsuperscript{2}92020 Troops, Qingdao, China
\textsuperscript{a}ahjpzy@163.com, \textsuperscript{b}zhubineei@163.com, \textsuperscript{c}905303927@qq.com, \textsuperscript{d}wecholer@126.com

Abstract. Deep convolutional neural networks deploy in applications are hindered by their large parameters and high computational cost. In this paper, we test different gradient optimizer’s effect on YOLOv3 model pruning. This is achieved by enforcing channel-level sparsity in YOLOv3 network. The model trained with Adam optimizer get 5× reduction in model size only after training 60 epochs.

1. Introduction
Object detection is developing faster and faster with the improvement of computer performance and the algorithm optimization of convolutional neural networks. This computer vision task is applied in many fields and become more and more closely with life. For example, medical diagnosis \cite{1}, object tracking \cite{2}, autonomous driving \cite{3}, etc. From a practical point of view, high speed and high precision is the most important aspect.

In deep learning field, object detection is divided into two genres: one-stage detection and two-stage detection. Where the former frames the detection as a “complete in one step” process while the later frames it as to “coarse-to-fine”. In generally, the one-stage detection algorithm is more advanced in speed, but accuracy is lower than the two-stage detection algorithm. In recent years, there appears several excellent one-stage object detection algorithm, such as YOLO \cite{4, 5, 6}, SSD \cite{7}, Retina-Net \cite{8}. YOLOv1 is the originator of the end-to-end model in the area of object detection, its core idea is to use the whole picture as the input of the network, directly regress the position of the bounding box at the output layer and its associated class. Because two boxes are predicted in a grid, and belong to one class. Therefore, for objects that are closed to each other and small group, YOLOv1 can’t detect well. YOLOv2 proposed a method to joint train on object detection and classification \cite{5}, the basic idea is that training the object detectors on both detection data set and the classification data set at the same time. The model learns object’s exact position from detection data, and increases the classification quantity of the classification and improve the robustness from classification data. Besides, YOLOv2 applied some improvement measures, such as batch normalization, high resolution classifier, convolution with anchor boxes, dimension clusters. Compared with YOLOv2, YOLOv3 make some further improvements: (1) employ logistic regression to predict bounding box; (2) improve single-label prediction to multi-label prediction; (3) combine 3 different sizes of bounding boxes for prediction; (4) adopt Darknet-53 as the backbone compared to YOLOv2’s Darknet-19.

YOLOv3 is currently the most popular one-stage object detection algorithm, many researchers do lots of works to improve the performance of YOLOv3. Mohammad Mahdi Derakhshani et al. \cite{9} inspired from curriculum learning and proposed a simple and effective learning technique that feed in
localization information by exciting certain activation during training, and improve the mAP of YOLOv3 by 2.2% on MSCOCO dataset.

When large CNNs deployed in products applications, there are three limits[10]: 1) Model size: millions of trainable parameters which explain the strong representation power should be stored on disk and loaded into memory during inference time. Larger CNNs mean more parameters and larger model size, which is a big resource burden to embedded devices. 2) Run-time memory: The intermediate activations/responses of CNNs could even take more memory space than storing the model parameters, and many applications with low computational power are unaffordable. 3) Number of computing operations: The convolution operations are computationally intensive on high resolution images. It is normal for large CNN to take minutes to test one single image on mobile device which has low computing power.

In order to retain large CNN models’ high efficiency, large works have been proposed to solve these problem. These works include two directions: 1) net-work quantization[11,12] and binarization[13,14], weight pruning[11], dynamic inference[15], etc. 2) sparsity the network[16, 17]. However these methods above require special software/hardware accelerators. Zhuang Liu et al. [10] proposed a simple yet effective network training scheme called network slimming achieving CNN models with up to 20× model-size compression and 5× reduction in computing operations of the original ones.

2. Related Work

Regularization. In the supervised learning, regularization has the vital effect on the model training. Regularization coefficient $\lambda$ balanced the empirical error and the regularization. It is likely to be overfitting when the value of $\lambda$ is too small, on the contrary, it may be underfitting if $\lambda$ is too large. Therefore, parameter $\lambda$ highly decide the model’s performance in the training.

The form of common regularization is L1 norm and L2 norm. The L1 norm is the sum of the absolute values of the elements in the matrix and it is usually used for achieving parameter matrix sparse by Eq. 1.

$$\min \sum_{(x,y)} L(y, f(x; w)) + \lambda \| w \|. \quad (1)$$

In comparison with L1 norm, L2 norm is equal to the root after the sum of squares of each element in a matrix, refer to Eq. 2.

$$\min \sum_{(x,y)} L(y, f(x; w)) + \frac{\lambda}{2} \| w \|^2. \quad (2)$$

It minimize each element in parameter matrix and bring it infinitely close to 0 which in L1 norm is equal to 0. With the exception of L1 and L2, another famous method is named dropout which randomly discard some parameters to lighten or prevent overfitting. By setting an inactivation probability for each layer’s neuron of the network, dropout will randomly discard some neuron when train a network. When apply dropout to convolution neural network, we can set the layers which have more neurons or easy to become overfitting with higher inactivation probability.

Model compression. Among the existing model compression methods, model pruning is one of the most popular methods. The method judge the importance of parameters by adding a sparsity-induced penalty on the scaling factors which is multiplied to the out of channel, and prune those channels with small factors at last. [18] proposed the pruning method based on weights, judge the importance by the value of weight. As for filter, calculating the sum of its absolute kernel weights and pruning the filter with low value. [19] defined APoZ(Average Percentage of Zeros) to measure the number of activation with 0 in each filter, and then judge the filter is important or not. [20] proposed an entropy-based pruning method which used entropy to judge the importance of filters. An energy-aware pruning was put forward by Yang T J in [21], the author calculate the energy consumption in each layer, and prune the layer with high energy consumption. In spite of model pruning, the sparsity of kernel is another direction to
compress model. This method induce the updating of weights to become more sparser in the training, and it can be divide into regular method and irregular method. [22] presented a learning method named Structured Sparsity Learning which can learn a sparsity structure to reduce calculation of consumption, the learnt structure with sparsity would effectively accelerate in hardware. Guo, Yiwen proposed a dynamic model cutting method [23], this method add splicing process to restore the pruning weights which are important. Liu et al. [10] put forward a simple and effective model pruning approach called network slimming. The approach directly applied the scaling factors in batch normalization (BN) layers as channel-wise scaling factors and trained networks with L1 regularization on these scaling factors to obtain channel-wise sparsity. Channel pruning is a coarse-grained but effective approach that it is convenient to apply the pruned models without the requirements of dedicated hardware or software. In this paper, we follow in Liu’s footstep, try several different hyperparameters setting and compare the performance.

3. Network slimming
Network slimming is a simple scheme to achieve channel-level sparse. The challenges of achieving channel-level sparse require pruning all the incoming and outgoing connections associated with the same channel [10]. A scaling factory for each channel is implemented when train the network weights, then prune those channels with small factors, and fine-tune the pruned network in the last. The theory is given by Eq. 3.

\[ L = \sum_{(x,y)} l(f(x,W), y) + \lambda \sum_{y \in \mathcal{Y}} g(y). \]

Which is known as L1-norm and widely used to achieve channel sparse.

**Sparse training with BN.** Batch normalization is widely adopted in modern CNNs to achieve fast convergence and better generalization performance. BN layer normalize convolutional features using mini-batch statics, which is formulated as follow Eq. 4.

\[ y = \gamma \times \frac{x - \bar{x}}{\sqrt{\sigma^2 + \epsilon}} + \beta. \]

Where \( \bar{x} \) and \( \sigma^2 \) are mean and variance of input features in a mini-batch, \( \gamma \) and \( \beta \) denotes trainable scale factor and bias.

**Channel pruning.** After training with sparsity, we obtain a model with many scaling factors near zero. Then the channels with near zero scaling factors are prune by removing all incoming and outgoing connections and corresponding weights. The threshold we prune channels is defined as a certain percentile of all the scaling factor values.

**Fine-tuning.** After channel pruning, the accuracy of model may descend a little, but perform fine-tuning on the pruned model could compensate the temporary degradation.

4. Experiments
We have tried different gradient optimizer and learning rate decay when train YOLOv3 model on oxford hand datasets. Our experiments were operated in the environment with python3.6 and pytorch1.1, the Linux server with Intel Xeon(R) CPU E5-2650 v4 @ 2.20GHz × 48, and TITAN V/PCIe/SSE2.

**Sparse training.** We trained with sparsity on Oxford hand datasets, and chose scale sparse rate with 0.01. The number of epochs were both 60, and learning rate attenuation mode was that reduce learning rate when loss has stopped descending. The training loss and validation mAP is shown in Fig. 1. The average mAP and model size with two different optimizer is shown in Table 1.

**Pruning and Fine-tuning.** We pruned the channels of models trained with sparsity and different gradient optimizers. The pruning percentage determine the pruning threshold, the maximum pruning percentage of model trained with SGD is 45%, and the model trained with Adam is 80%. After pruning, a fine-tuning process was applied in the pruned model with 80 epochs. Fig. 2 shows the fine-tuning performance.
5. Results
We compare the detection performance of two models which both trained with sparsity on validation set of Oxford hand in Fig. 1, and the models trained with SGD optimizer and Adam optimizer have the same model size in Table 1.

Table 1. MAP and Model size trained with different gradient optimizer

| Optimizer                  | mAP | Model size |
|----------------------------|-----|------------|
| SGD(warm up + lr decay)    | 0.70| 246.4MB    |
| Adam(warm up + lr decay)   | 0.69| 246.4MB    |

Fig. 1: The training loss and mAP of validation sets with SGD optimizer (left) and Adam optimizer (right)

Table 2. Model size pruned percentile with different gradient optimizer

| Model        | Sparse train epochs | Model size pruned | Pruned percents |
|--------------|---------------------|-------------------|-----------------|
| Yolov3(SGD)  | 60                  | 99.6MB            | 45%             |
| Yolov3(Adam) | 60                  | 46.7MB            | 80%             |
Fig. 2: Fine-tuning the pruned models which trained with different gradient optimizer, the train loss and validation mAP with SGD optimizer are the left and the right are Adam optimizer.

As shown in Table 2, we can get a more compact model trained with Adam optimizer in the same training epochs, and mAP on validation set is all around 0.72 after fine-tuning.

6. Conclusion
In this paper, we try different optimizer in training YOLOv3, and find that the model trained with Adam gradient optimizer is easier to get a compact model. Network slimming technique [10] directly impose sparsity-induced regularization on the scaling factors in batch normalization layers, and automatically identify unimportant channels during training and then pruned. In comparison with SGD, Adam optimizer is a better choice for sparse data and complex convolution neural network.

References
[1] Gurovich Yaron, Hanani Yair et al. Identifying facial phenotypes of genetic disorders using deep learning
[2] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang et al., “T-cnn: Tubelets with convolutional neural networks for object detection from videos,”IEEE Transactions on Circuits and Systems for Video Technology, vol. 28, no. 10, pp. 2896–2907, 2018.
[3] Chen C, Seff A, Kornhauser A, et al. [IEEE 2015 IEEE International Conference on Computer Vision (ICCV) - Santiago, Chile (2015.12.7-2015.12.13)] 2015 IEEE International Conference on Computer Vision (ICCV) - DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving [J]. 2015:2722-2730.
[4] Redmon J, Divvala S, Girshick R, et al. You Only Look Once: Unified, Real-Time Object Detection [J]. 2015.
[5] Redmon J, Farhadi A. [IEEE 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) - Honolulu, HI (2017.7.21-2017.7.26)] 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) - YOLO9000: Better, Faster, Stronger[J]. 2017:6517-6525.
[6] Redmon J, Farhadi A. YOLOv3: An Incremental Improvement [J]. 2018.
[7] Liu W, Anguelov D, Erhan D, et al. SSD: Single Shot MultiBox Detector[C]// European Conference on Computer Vision. 2016.
[8] Lin T Y, Goyal P, Girshick R, et al. Focal Loss for Dense Object Detection [J]. IEEE Transactions
on Pattern Analysis & Machine Intelligence, 2017, PP (99):2999-3007.

[9] Derakhshani M M, Masoudnia S, Shaker A H, et al. Assisted Excitation of Activations: A Learning Technique to Improve Object Detectors[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019: 9201-9210.

[10] Liu Z, Li J, Shen Z, et al. Learning efficient convolutional networks through network slimming[C]//Proceedings of the IEEE International Conference on Computer Vision. 2017: 2736-2744.

[11] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. In ICML, 2015.

[12] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In NIPS, pages 1135–1143, 2015.

[13] M. Courbariaux and Y. Bengio. Binarynet: Training deep neural networks with weights and activations constrained to+ 1 or-1. arXiv preprint arXiv:1602.02830, 2016.

[14] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnornet: Imagenet classification using binary convolutional neural networks. In ECCV, 2016.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016.

[16] S. Changpinyo, M. Sandler, and A. Zhmoginov. The power of sparsity in convolutional neural networks. arXiv preprint arXiv:1702.06257, 2017.

[17] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky. Sparse convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 806–814, 2015.

[18] Hao L, Kadav A, et al. Pruning Filters for Efficient ConvNets [J]. 2016.

[19] Hu H, Peng R, Tai Y W, et al. Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures [J]. 2016.

[20] Luo J H, Wu J. An Entropy-based Pruning Method for CNN Compression [J]. 2017.

[21] Yang T J, Chen Y H, Sze V. Designing Energy-Efficient Convolutional Neural Networks Using Energy-Aware Pruning[C]// 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2017.

[22] Wen W, Wu C, Wang Y, et al. Learning Structured Sparsity in Deep Neural Networks [J]. 2016.

[23] Guo Y, Yao A, Chen Y. Dynamic Network Surgery for Efficient DNNs [J]. 2016.