# Compressed Objects Detection


## Abstract

Deep learning approaches have achieved unprecedented performance in visual recognition tasks such as object detection and pose estimation. However, state-of-the-art models have millions of parameters represented as floats which make them computational expensive and constrain their deployment on hardware such as mobile phones and IoT nodes. Most commonly, activations of deep neural networks tend to be sparse thus proving that models are over parametrized with redundant neurons. Model compression techniques, such as pruning and quantization, have recently shown promising results by improving model complexity with little loss in performance. In this work, we extended pruning, a compression technique which discards unnecessary model connections, and weight sharing techniques for the task of object detection. With our approach we are able to compress a state-of-the-art object detection model by 30.0% without a loss in performance. We also show that our compressed model can be easily initialized with existing pre-trained weights, and thus is able to fully utilize published state-of-the-art model zoos.


## 1 Introduction

Deep neural networks are computationally expensive. Their memory and compute complexity limit their deployment on edge devices and make them unsuitable for applications with strict latency requirements [9]. The progress in VR, AR, IoT and smart wearable devices create opportunities for researchers to tackle the challenges of deploying deep neural networks on devices with constrained memory, CPU, bandwidth and energy [3, 5]. To this end, compression techniques have shown promising results in reducing the memory and time requirements of deep neural networks. Pruning reduces the model's complexity by removing unnecessary elements in their structures at different levels [7]. Quantization converts a model to use reduced precision integer representation for the weights or activations. [6, 2]. In this work, we apply pruning and quantization techniques for visual recognition tasks. We apply these compression techniques on the convolutional neural networks which commonly comprise the backbone architectures of state-of-the-art recognition models such as in Faster R-CNN [12]. Model compression on CNNs is well suited as recent studies [7] have shown that CNN structures have redundant parameters which can be removed without loss in performance. In addition to pruning, we further improve the efficiency of object recognition models by quanitizing their parameters from 32-bit float to 8-bit integer precision. Finally, we show that our compression techniques also support pre-traiend model weight initialization thus, making it possible to take advantage of published model zoos.

## 2 Methods and Experiments

Our approach compresses the model parameters by introducing sparsity into the weights and using few bits to represent each parameter with marginal performance sacrifice.

**Pruning.** We remove the parts of the model that contribute less or nothing to the final performance. *Weight pruning* ranks the individual parameters in the parameter matrix $w$ according to their magnitude (absolute value), and then set to zero the smallest k% of the weights. *Unit/Neuron pruning* sets entire columns in the parameter matrix to zero. This corresponds to completely removing a neuron. We use $L_2$-norm to rank the columns of the parameters matrix.

We cast pruning as the following optimization problem.

$$\min_{w} L(w; D) = \min_{w} \frac{1}{n} \sum_{i=1}^{n} l(w; x_i y_i) \quad \text{s.t.} \quad \|w\|_0 \leq k \tag{1}$$

where $w$ are our model parameters, $D = \{x_i, y_i\}_{i=1}^{n}$ is our dataset, $l(w; x, y)$ is our loss function and $k$ represents the desired sparsity level in the parameters.

**Experimental Setup.** Before applying any compression techniques, using our custom dataset of 1309 instances that we collected from East African parks, we consider the training of a faster RCNN with a Resnet50 backbone and with FPN [4]. Following the common experimental setting in related work on network pruning in [7], we extended their approach for image classification to object detection with a Faster RCNN architecture. Our approach can be easily extended to other object detection models such as YOLO [11, 10] among others. Our model has three main blocks; backbone, a proposal generator and ROI heads. The whole model has a total of 41.4 Millions of trainable parameters. The backbone has more than 60% of the total parameters, which makes it the most targeted block in our compression experiments. We take advantage of the publicly available state-of-the-art object detection models in Detectron2 and its model zoos [13, 1]. We use the Pytorch pruning [8] and quantization libraries implemented in Pytorch. We implement global pruning of k% parameters and neurons and then we quantize our parameters to 8 bits.

**Experimental Results.** Regardless of the task, pruning imposes a trade-off between model efficiency and quality, with pruning increasing the former while (typically) decreasing the latter.

Table 1 shows a comparison of different methods of pruning. For low sparsity our approaches outperforms even the dense baseline, which is in line with regularization properties of network pruning. On large models, pruning shows reasonable performance even with extremely high sparsity level.

Pruned percentage in the backbone

| | 0% | 10% | 20% | 30% | 40% | 50% | 70% | 80% | 90% |
|---|---|---|---|---|---|---|---|---|---|
| AP50 | 87.92 | 87.97 | 88.40 | 88.15 | 86.95 | 83.48 | 77.42 | 60.24 | 0.17 |
| Memory(MBs) | 165.6 | 154.88 | 144.16 | 133.44 | 122.72 | 112.0 | 90.56 | 79.84 | 69.12 |

Pruned percentage in the ROI head

| | 0% | 10% | 20% | 30% | 40% | 50% | 70% | 80% | 90% |
|---|---|---|---|---|---|---|---|---|---|
| AP50 | 87.92 | 87.97 | 88.40 | 88.15 | 86.95 | 83.48 | 77.42 | 60.24 | 0.17 |
| Memory(MBs) | 165.6 | 160.0 | 154.4 | 148.8 | 143.2 | 137.6 | 126.4 | 120.8 | 115.2 |

Pruned percentage in the both backbone and the ROI head

| | 0% | 10% | 20% | 30% | 40% | 50% | 70% | 80% | 90% |
|---|---|---|---|---|---|---|---|---|---|
| AP50 | 87.92 | 87.72 | 87.04 | 83.60 | 73.64 | 61.18 | 54.69 | 33.12 | 0.14 |
| Memory(MBs) | 165.6 | 149.2 | 132.9 | 116.6 | 100.3 | 84.0 | 51.3 | 35.0 | 18.7 |

## 3 Conclusion

We show that pruning and quantization techniques can efficiently compress object recognition models with little loss in performance. We can prune 40% of the model with loss of a few points in average precision. The reduction in memory allows for efficient storage and enables deployment of object detectors on devices of lower computational capacity.

## References

[1] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. `https://github.com/facebookresearch/detectron`, 2018.

[2] Sambhav R. Jain, Albert Gural, Michael Wu, and Chris H. Dick. Trained quantization thresholds for accurate and efficient fixed-point inference of deep neural networks, 2019.

[3] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A survey of deep learning-based object detection. *IEEE Access*, 7:128837–128868, 2019.

[4] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2016.

[5] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning, 2018.

[6] Prateeth Nayak, David Zhang, and Sek Chai. Bit efficient quantization for deep neural networks, 2019.

[7] Morteza Mousa Pasandi, Mohsen Hajabdollahi, Nader Karimi, and Shadrokh Samavi. Modeling of pruning techniques for deep neural networks simplification, 2020.

[8] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[9] Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. Binary neural networks: A survey. *Pattern Recognition*, 105:107281, Sep 2020.

[10] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2015.

[11] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018.

[12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2015.

[13] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.