YOLOv3-VD: A sparse network for vehicle detection using variational dropout

Dinh Viet Sang*
Hanoi University of Science and Technology
Hanoi, Vietnam
sangdv@soict.hust.edu.vn

Duong Viet Hung
Hanoi University of Science and Technology
Hanoi, Vietnam
hunglc007@gmail.com

ABSTRACT

Deep neural networks (DNNs) are currently state-of-the-art methods in many important AI tasks. However, DNNs usually contain a lot of parameters that make them prone to overfitting and slow in inference. In this paper, we apply variational dropout to sparsify YOLOv3 network for vehicle detection. We then prune redundant layers and compress the network to reduce the memory size and accelerate the inference speed of the model. Experiments show that we can eliminate up to 91% weights in the original YOLOv3 with a negligible decrease of accuracy.

KEYWORDS

Sparse Modelling, Variational Bayesian Inference, Deep Learning, Object Detection

ACM Reference Format:
Dinh Viet Sang and Duong Viet Hung. 2019. YOLOv3-VD: A sparse network for vehicle detection using variational dropout. In *The Tenth International Symposium on Information and Communication Technology (SoICT 2019), December 4–6, 2019, Hanoi - Ha Long Bay, Viet Nam.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3368926.3369691

1 INTRODUCTION

Object detection is one of the most important and challenging computer vision tasks, which forms the basic for solving many other high-level computer vision tasks such as instance segmentation, object tracking, image captioning and so on. Despite a lot of effort has been spent on solving object detection problem, this task is still considered not being solved yet. In recent years, deep neural networks have demonstrated their impressive results in many fields. Particularly, major recent improvements in object detection have been made thanks to deep learning techniques.

The purpose of generic object detection is to classify instances of a predefined set of objects in an image and locate each detected instance by a bouding box with some confidence score. In general, there are two different approaches for object detection: one-stage and two-stage detection. In two-stage detectors such as R-CNN [8], SPP-net [20], Fast R-CNN [7], Faster R-CNN [24], R-FCN [5], FPN [14] and Mask R-CNN [11], the model first defines a small set of candidate object regions by using selective search or a region proposal network. In the second stage, a classifier evaluates each candidate region to determine whereas an object exists in that region and correct the location of the object if needed. In contrast, one-stage detectors treat object detection as a simple regression problem and run the detection over a large set of candidate object locations, which are densely sampled across the image and cover multiple scales and aspect ratios. One-stage detectors is faster and simpler, but might potentially drag down the performance a bit. Along with various one-stage detectors such as MultiBox [6], SSD [16] and DSOD [26], YOLO networks [21–23] are popular choice for real-time object detection. YOLOv3 [23] is the latest version of YOLO, which has several improvements compared to the first one in [21]. However, YOLOv3 still has over 60 millions of parameters and requires a lot of computations to infer a single input sample. This makes Yolov3 hard to be deployed in resource constrained environments or in large scale IVA systems.

In this paper, we adopt sparsity techniques to address this challenge. Sparse models contain many zero valued weights that allows us to skip a lot of multiplications which dominate neural network computation. Higher sparsity leads to fewer weights, and less computational and storage requirements. There are many approaches for inducing sparsity in deep neural networks. Simple heuristics based on removing small magnitude weights have demonstrated high compression rates with minimal accuracy loss ([3, 10, 27]), and further refinement of the sparsification process for magnitude pruning techniques has increased achievable compression rates and greatly reduced computational complexity ([9, 28]). Many techniques grounded in Bayesian statistics and information theory have been proposed ([4, 17–19]). These methods have achieved high compression rates while providing deep theoretical motivation and connections to classical sparsification and regularization techniques. In particular, Molchanov et al [19] use Variational Dropout to sparsify simple convolutional neural networks for image classification task: LeNet and VGG-like networks. They show that Variational Dropout leads to extremely sparse solutions both in fully-connected and convolutional layers, reducing the the number of parameters up to 280 times on LeNet and up to 68 times on VGG-like networks with a negligible decrease of accuracy.

In this paper, we use Variational Dropout to sparsify YOLOv3, which is a much more complicated deep neural network than those in [19]. Our proposed sparse network called Yolov3-VD is trained and evaluated on our seft-collected dataset about traffic in Vietnam. Experiments show that we can eliminate upto 91% weights in the original YOLOv3 a negligible decrease of accuracy.

*Corresponding author.

SoICT 2019, December 4–6, 2019, Hanoi - Ha Long Bay, Viet Nam

ACM ISBN 978-1-4503-7245-9/19/12…$15.00
https://doi.org/10.1145/3368926.3369691

The rest of the paper is organized as follows.In section 2, we briefly introduce our proposed approach. In section 3, we show the experimental reuslts on our self-collected traffic dataset. The conclusion is in section 4 with a discussion for the future work.

2 OUR PROPOSED APPROACH

2.1 YOLOv3

YOLO [21] is the very first attempt at building an object detection system for real-time processing. YOLO divides the input image into an $S \times S$ (S=7 in YOLO) grid and predicts over the cells of this grid. This method greatly reduces the computational complexity, speeds up the target detection, and the frame rate can reach up to 45 fps. YOLOv2 [22] is an improved version of YOLO. With a lots of modifications in backbone network and anchor design, YOLOv2 is more accurate and faster. Inspired by recent advances in object detection, a bunch of design tricks has applied to YOLOv2 to create a more powerful detector called YOLOv3, which architecture is shown in the Fig. 1. First, YOLOv3 [23] predicts the confidence score for each bounding box using logistic regression instead of sum of squared errors as in its predecessors. YOLOv3 is based on a new backbone network called Darknet-53. Inspired by Resnet [12], Darknet-53 has the same architecture like the previous one, but has residual blocks added. Compared to ReseNet-101, the Darknet-53 network is 1.5 times faster. Darknet-53 yields the same accuracy level but faster twice than ReseNet-152 [23]. Inspired by FPN [14], YOLOv3 adds several additional conv layers after the backbone feature extractor Darknet-53 and makes predictions at three different scales, which help YOLOv3 capable to detect objects of various sizes overall. Besides, YOLOv3 upsamples lower resolution feature maps and merge them with the next higher resolution feature maps using element-wise addition. The combination with finer-grained information makes YOLOv3 better at dealing with small objects.

2.2 Variational Dropout

Variational Dropout [19] was originally proposed as a reinterpretation of dropout training under the Bayesian view through variational inference. Variational Dropout allows to extend the standard dropout algorithms with per-weight learnable dropout rates. After training, weights with high dropout rates can be removed to produce highly sparse neural networks.

Suppose that $\mathcal{D}$ is the training set that consists of n samples with corresponding labels $(x_i, y_i)_{i=1}^n$. Let $\mathbf{w}$ be the set of all weights of a model (*e.g.* YOLOv3) with a prior distribution $p(\mathbf{w})$. Training the model means to tune $\mathbf{w}$ to predict y, given x and $\mathbf{w}$, in term of conditional probability $p(y|x, \mathbf{w})$. The posterior distribution of the weights then can be estimated as follows:

$$p(\mathbf{w}|\mathcal{D}) = p(\mathcal{D}|\mathbf{w})p(\mathbf{w})/p(\mathcal{D}). \tag{1}$$

Since the direct calculation following Eq. 1 is intractable, the posterior distribution $p(\mathbf{w}|\mathcal{D})$ is usually approximated by a parametric distribution $q_\phi(\mathbf{w})$. This method is call variational inference. The optimal values of parameters ϕ can be found by maximizing the variational lower bound:

$$\phi^* = \underset{\phi \in \Phi}{\operatorname{argmax}} L_{\mathcal{D}}(\phi) - D_{KL}(q_\phi(\mathbf{w})||p(\mathbf{w})), \tag{2}$$

where the expected log-likelihood $L_{\mathcal{D}}(\phi)$ is calucalated as follows:

$$L_{\mathcal{D}}(\phi) = \sum_{i=1}^{n} \mathbb{E}_{q_\phi(w)}[log\ p(y_i|x_i, \mathbf{w})], \tag{3}$$

and $D_{KL}(q_\phi(\mathbf{w})||p(\mathbf{w}))$ stands for the KL-divergence of $q_\phi(\mathbf{w})$ w.r.t $p(\mathbf{w})$.

Assume that the variational distribution $q_\phi(\mathbf{w})$ over $\mathbf{w}$ has the following form:

$$q_\phi(\mathbf{w}) = \prod_{w_{ij} \in \mathbf{w}} q(w_{ij}|\theta_{ij}, \alpha_{ij}), \tag{4}$$

where $q(w_{ij}|\theta_{ij}, \alpha_{ij})$ is a Gaussian distribution with mean θ_{ij} and variance $\sigma_{ij}^2 = \alpha_{ij}\theta_{ij}^2$, *i.e.* $w_{ij} \sim \mathcal{N}(w_{ij}|\theta_{ij}, \alpha_{ij}\theta_{ij}^2)$.

Here θ_{ij} and α_{ij} are learnable parameters, where α_{ij} defines the drop rate p_{ij} of weight w_{ij} as follows:

$$p_{ij} = \frac{a_{ij}}{1 + a_{ij}}. \tag{5}$$

If $\alpha_{ij} = 0$ then $p_{ij} = 0$ and it means that no drop rate is applied to w_{ij}. Otherwise, if $\alpha_{ij} \to +\infty$ then $p_{ij} \to 1$ and w_{ij} is completely dropped and can be removed to sparsify the model.

In order to make the model trainable via gradient based methods, re-parameterization trick is then applied:

$$\begin{aligned} w_{ij} &= \theta_{ij}(1 + \sqrt{\alpha_{ij}}\,\epsilon_{ij}) = \theta_{ij} + \sigma_{ij}\epsilon_{ij}, \\ \epsilon_{ij} &\sim \mathcal{N}(0, 1). \end{aligned} \tag{6}$$

The prior distribution $p(\mathbf{w})$ is chosen as $p(|w_{ij}|) \propto \frac{1}{|w_{ij}|}$. According to [19], the KL-divergence term then can be approximated as follows:

$$\begin{aligned} D_{KL}(q_\phi(\mathbf{w})||p(\mathbf{w})) &= \sum_{ij} D_{KL}\left(\mathcal{N}(w_{ij}|\theta_{ij}, \alpha_{ij})||p(w_{ij})\right) \\ &= -\frac{0.63576}{1 + e^{-1.87320 + 1.48695\ \log\alpha_{ij}}} + 0.5\ \log(1 + \frac{1}{\alpha_{ij}}) + C \end{aligned} \tag{7}$$

The problem in Eq. (2) can be solved using Stochastic Variational Inference [19].

2.3 Generalized IoU based loss function

[25] presents a new way to optimize the bounding box GIoU (Generalized IoU). The authors found that using IoU has two drawbacks that make it less suitable to be used for loss functions. When there is no intersection between the prediction box and the real box, the IoU value is 0, even better prediction was closer to the ground truth, which results in a gradient of 0 when optimizing the loss function, meaning that optimization cannot be performed. Even if the prediction box and the real box intersection and have the same IoU value, the detection effect is quite different.

In this paper, in this paper, we use GIoU for bounding box regression loss L_{GIoU}, while keeping the confidence loss L_{conf} and the binary cross-entropy loss L_{prob} the same as in [23]:

$$L_{\mathcal{D}}(\phi) = L_{GIoU} + L_{conf} + L_{prob}. \tag{8}$$

We also introduce λ to balance the values of D_{KL} and $L_{\mathcal{D}}(\phi)$ during training. Thus, the final loss is:

$$L_{total} = L_{\mathcal{D}}(\phi) + \lambda D_{KL}(q_\phi(\mathbf{w})||p(\mathbf{w})). \tag{9}$$

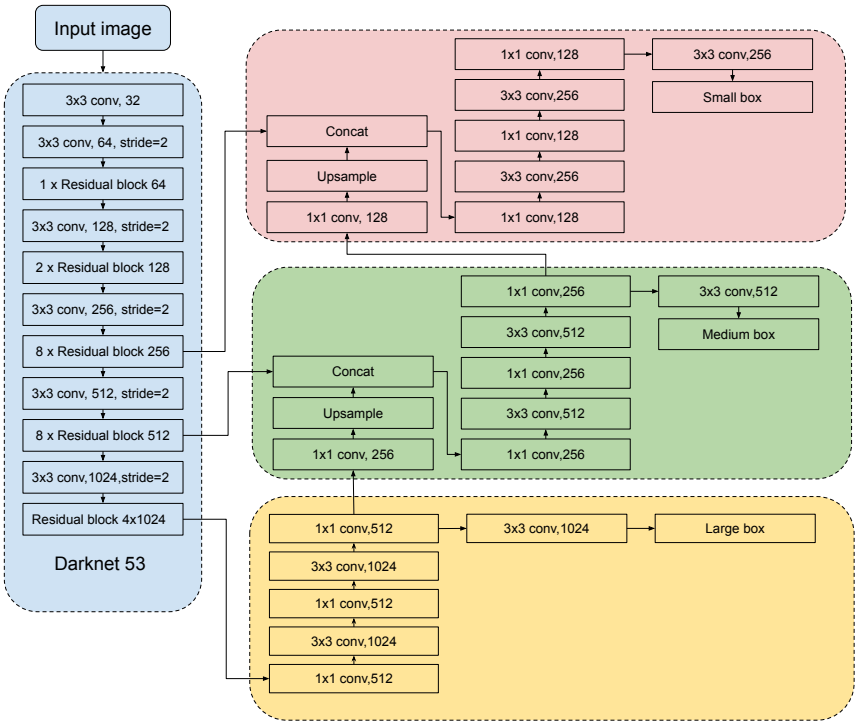

Figure 1: YOLOv3 architecture

3 EXPERIMENTS AND EVALUATION

3.1 Dataset

In this paper, we use our self-collected dataset about traffic in Vietnam. There are totally 5884 images with six categories: Car, SUV, Bus, Truck, Motorbike, Person and Bicycle. The average number of objects in an image is about 5. The dataset is divided into training set and validation set with a ratio of 5:1.

Figure 2: A sample in our dataset

3.2 Implementation details

Our proposed model is implemented in Python 3 with Tensorflow 1.13 framework [2] and evaluated on a server with the following specifications: Intel Core i5-8400 CPU 6 cores 2.80GHz Processor, Ubuntu Server 16.04.3 LTS 64-bit Operating System, 32GB of RAM and GPU NVIDIA GeForce GTX2080Ti.

3.3 Preprocessing

We used three main methods for data augmentation as follows:

- Random flip: we flip an image from left to right and upside down;
- Random crop: we takes random crops from original images;
- Random translations: original images in the datasets are also randomly translated.

The order of data augmentation is Random flip, Random crop and Random translations. Each step occurs with a probability of 0.5. Before fitting the model, all images are resized to a fixed size such as 416×416 or 608×608.

3.4 Training with bags of tricks

All weights of two models YOLOv3 and YOLOv3-VD are initialized with the pre-trained weights obtained on Coco dataset [15]. Due to the difference in class numbers between our dataset and COCO dataset, we use two stages in the training phase. At first stage, we train only last three layers, while freezing the remaining ones. At second stage, we unfreeze all layers and train them.

Tuning learning rate is a crucial point in training neural networks. In the beginning of training phase, the weights of neural networks are often randomly initialized and they are far away from

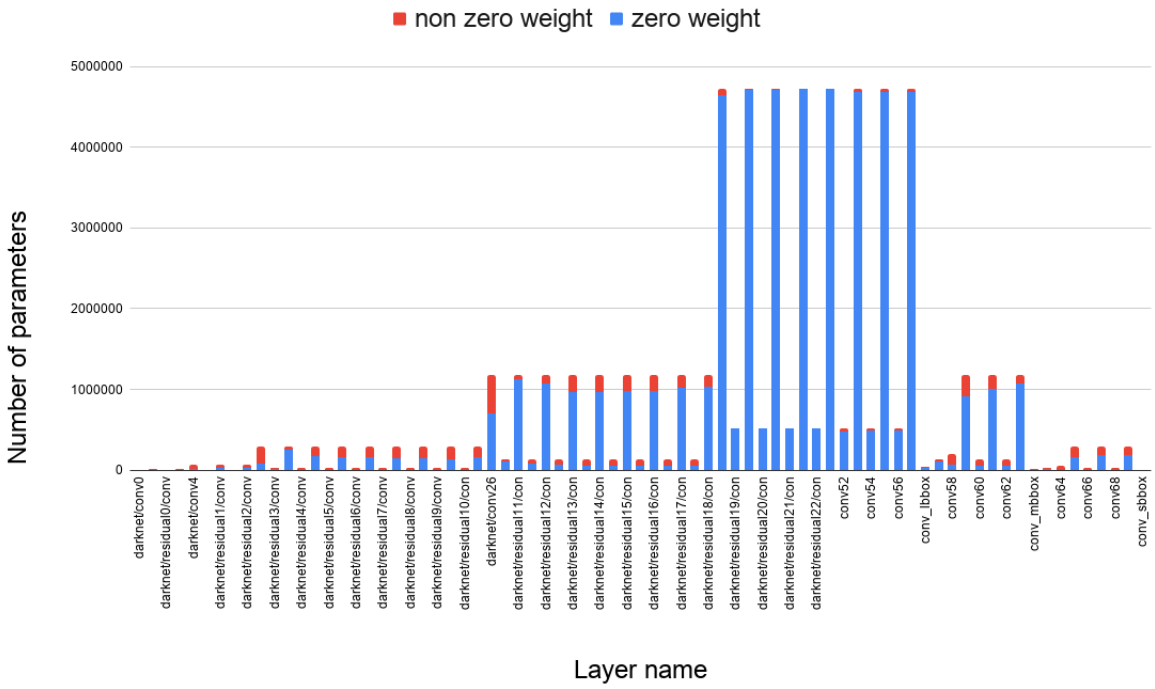

Figure 3: Sparsity ratio in YOLOv3 layers.

the converged solution. Using a too large learning rate at the beginning steps can lead to numerical instability. Here we use the warm-up phase following [13]. A small learning rate is used at the beginning and then we use the normal initial learning rate when the training process is stable. In our experiments, we use 2 epochs for the warm-up phase.

We use Adam optimization algorithm with learning rate gradually decreases from 10^{-4} to 10^{-6} during training using Cosine Learning Rate Decay [13]. All parameters in every batch normalization layer are also retrained with Exponential Moving Average and decay 0.99995. Number of anchors per scale is 3.For training phase, we set the batch size to 1 and the IOU loss threshold to 0.5. Meanwhile for testing phase, the batch size is set to 2 and the IOU loss threshold is set to 0.45.

We train both original YOLOv3 and our YOLOv3-VD in 85 epochs. As mentioned above, neural networks are often unstable in the beginning training steps. During training our YOLOv3-VD, parameter λ is used to balance object detection loss and KL-divergence. Using a large value λ means to enforce a strong level of sparsity to the networks, which causes underfitting effect. Therefore, we use small λ in beginning steps in order to make the networks more "comfortable" in learning from the data. Particularly, we set $\lambda = 0.001$ for epoch 0 to epoch 50. After 50 epochs, the model converges and the loss is not improved anymore. This is the time to enforce a strong sparsity level to the network. Thus, we increase $\lambda = 0.01$ from epoch 50 to 65, and then set $\lambda = 0.1$ from 65 to 70 and, finally, set $\lambda = 1$ from epoch 70.

3.5 Results

Table 1 shows our results. We achieve 83.78% mAP with YOLOv3 baseline model. Surprisingly, our proposed model YOLOv3-VD can achieve 82.04% mAP with sparsity level up to 90.95%, that mean only about 9% its weights have non-zero values. Fig. 4 illustrates our training process. Despite the sparsity level of YOLOv3-VD gradually increases, we just observe a slight increase of validation loss that results in a negligible decrease of accuracy.

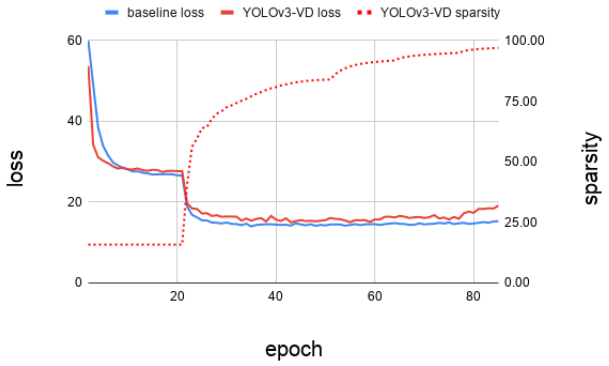

Figure 4: Sparsity-loss trade-off curves for YOLOv3

The sparsity level of different layers of our YOLOv3-VD is shown in Fig. 3. Surprisingly, while the sparsity level in the first layers

Table 1: Model comparison

Models	mAP (%)	Number of weights	Number of non-zero weights	sparsity ratio (%)
YOLOv3	83.78	61.503.325	61.503.325	0
YOLOv3-VD	82.04	61.503.325	5.564.051	90.95
YOLOv3-VD-pruned	80.70	40.531.808	5.532.556	91

seems reasonable, the sparsity ratio the last residual blocks is about 99.9%. It means that these last residual blocks are almost identity mappings that can be completely pruned from the network to reduce the computation and required memory. Table 1 shows that the pruned version YOLOv3-VD-pruned has just a slight drop in accuracy compared with YOLOv3-VD and YOLOv3. We also use NVIDIA TensorRT inference optimizer runtime [1] to boost the speed of YOLOv3-VD-prunned up to 87 fps on GPU NVIDIA GeForce GTX 2080Ti.

4 CONCLUSION

In this paper, we propose a sparse network called YOLOv3-VD, which is based on YOLOv3 network and sparsified by variational dropout. We also replace the old box regression loss in YOLOv3 with GIoU loss. Our network YOLOv3-VD achieve sparsity level up to 91 % with almost the same accuracy compared to the original network YOLOv3.

5 ACKNOWLEDGMENTS

This research was sponsored by the U.S. Army Combat Capabilities Development Command (CCDC) Pacific and CCDC Army Research Laboratory (ARL) under Contract Number W90GQZ-93290007. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the CCDC Pacific and CCDC ARL and the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

[1] Nvidia. 2018. tensorrt: https://developer.nvidia.com/tensorrt.
[2] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.
[3] M. D. Collins and P. Kohli. Memory bounded deep convolutional networks. *arXiv preprint arXiv:1412.1442*, 2014.
[4] B. Dai, C. Zhu, and D. Wipf. Compressing neural networks using the variational information bottleneck. *arXiv preprint arXiv:1802.10399*, 2018.
[5] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.
[6] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2147–2154, 2014.
[7] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.
[8] 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*, pages 580–587, 2014.
[9] Y. Guo, A. Yao, and Y. Chen. Dynamic network surgery for efficient dnns. In *Advances In Neural Information Processing Systems*, pages 1379–1387, 2016.
[10] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.
[11] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.
[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
[13] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019.
[14] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.
[15] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.
[17] C. Louizos, K. Ullrich, and M. Welling. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*, pages 3288–3298, 2017.
[18] C. Louizos, M. Welling, and D. P. Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.
[19] D. Molchanov, A. Ashukha, and D. Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2498–2507. JMLR. org, 2017.
[20] P. Purkait, C. Zhao, and C. Zach. Spp-net: Deep absolute pose regression with synthetic views. *arXiv preprint arXiv:1712.03452*, 2017.
[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.
[22] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.
[23] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.
[24] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
[25] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 658–666, 2019.
[26] Z. Shen, Z. Liu, J. Li, Y.-G. Jiang, Y. Chen, and X. Xue. Dsod: Learning deeply supervised object detectors from scratch. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1919–1927, 2017.
[27] N. Ström. Sparse connection and pruning in large dynamic artificial neural networks. In *Fifth European Conference on Speech Communication and Technology*, 1997.
[28] M. Zhu and S. Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.