Robust Compression Technique for YOLOv3 on Real-Time Vehicle Detection

Nattanon Krittayanawach and Peerapon Vateekul
Chulalongkorn University Big Data Analytics and IoT Center (CUBIC)
Department of Computer Engineering, Faculty of Engineering
Chulalongkorn University
Bangkok, Thailand
6071010621@student.chula.ac.th, peerapon.v@chula.ac.th

***Abstract*—For vehicle detection, YOLOv3 has shown promising accuracy. Since the number of parameters in this network can be more than ten million parameters, it cannot be fit into a commodity camera. In this paper, we propose a compression mechanism designed specifically for YOLOv3's network by removing unnecessary filters. Since YOLOv3 composes of two network components: backbone and pyramid networks, we propose a robust pruning mechanism to prune filters of each network separately. This can help to avoid over-pruning the network in some part of the model making our model more robust. There are two main pruning criteria investigated: Average Percentage of Zero (APoZ) and Sum Magnitude Weight. The experiment was conducted on UA-DETRAC. The results show that our compression mechanism with APoZ criterion can reduce more than 90% of the network size, while the accuracy is even higher than the full model for about 2%.**

Keywords—Vehicle detection; Deep learning; Pruning; Object detection

I. Introduction

Vehicle detection is one of the important tasks. Vehicle detection module now utilized in many applications, for example (i) autonomous car driving system, (ii) video surveillance, and (iii) car tracking system. The problems are these tasks are usually most efficient to process the data at the camera device. However, small devices such as camera devices usually have limited computation capability and power consumption while transferring real-time video data over the network is not efficient with high latency. For large-scale applications, high network bandwidth is required, with latency also playing an important role. Thus, the need to process data at a camera device becomes inevitable.

In recent years, deep learning has led to significant advancements in computer vision tasks such as image classification, object detection, and object segmentation. A deep learning-based model relies on feature extracted from deep learning network, unlike the traditional machine learning base such as DPM [1] that utilizes HOG [2] as the main feature for learning. A deep learning object detection model is usually designed as a two-stage detector to emphasize accuracy. The two-stage detector type model such as Faster R-CNN [3] has significant improvement compared to the traditional method. However, a two-stage detector still has one major problem, that is a large computational power is required to operate. Because the model is initially designed to emphasize accuracy, this problem makes two-stage detector does not suit for small devices such as camera devices with limited power consumption and computation capability. To solve this problem, Redmon, et al. [4] proposed YOLO as a one-stage deep learning-based object detection model that emphasizes speed. The key difference from previous works is that YOLO is one-stage deep learning-based object detection. This model treats the object detection problem as a regression problem make YOLO is fast by design. The recent version, YOLOv3 [5], contains many improvements over previous work. YOLO can be considered as one of milestone that makes deep learning-based object detection model get closer to real-time tasks.

Even the YOLOv3 is considered fast in terms of deep learning-based object detection models. However, the model itself still has a problem. That is the model size is very large since the model has over ten million parameters that consider too large for small devices such as camera devices. To solve this problem, many studies developed network compression techniques that aim to reduce model size and improve speed. Compression techniques can be grouped into three main categories, for instance, (i) model quantization and binarization (ii) pruning and sharing, and (iii) structural matrix.

Pruning is one of the most popular compression techniques that aim to compress the model network by removing unnecessary weight. Pruning can significantly reduce the computational cost and size of the model. The technique can be divided into two categories; (i) unstructured pruning, and (ii) structured pruning [6]. Unstructured pruning aims to remove unnecessary weight without considering the model structure; the compressed model may contain sparse weight in each layer, and the structure of the model may be damaged. The damaged model from unstructured pruning is often no longer compatible with commercial hardware and software, and the compressed model cannot gain an advantage from network compression. On the other hand, structured pruning focuses on removing weight at the convolutional filter level. Thus, the compressed model does not receive any damage to its structure that makes the compressed model remains compatible with commercial hardware and software.

In this paper, we propose a robust compression mechanism for YOLOv3. We found that weight in each part of the model has a different characteristic. Our method uses separated pruning and minimum filter constraint to prevent an over-pruning issue which occurs in a standard iterative pruning procedure [7]. As a result, we successfully prune YOLOv3 model at a higher pruning ratio compare to other baseline methods without causing a major reduction in accuracy. We evaluate our proposed method and our baseline methods on the vehicle detection dataset UA-DETRAC [8].

This paper is organized as follows. Section II reviews related works on object detection and compression techniques. Section III explains our algorithm. Section IV and V explain how the experimental setup and results. The conclusion is shown in Section VI.

II. Related Works

We discuss two fields of related work, including object detection related work and compression technique related work.

A. Object Detection

In the past, machine learning-based object detection models [1] relied on a hand-crafted feature such as HOG feature. After deep learning emerges, the hand-craft feature was replaced by feature from CNN. R-CNN [9] is the first major milestone of deep learning object detection. The model successfully brings significant improvement to the object detection domain. Later, Fast-RCNN [10] and Faster R-CNN were developed as an improved version of R-CNN, speed and accuracy significantly improved over the original model. However, the computational cost was still very high, making the model unsuitable to operate on small devices.

In 2016, Redmon, et al. [4] proposed a one-stage object detection network YOLO by redesigning the object detection problem as a regression problem. The model was designed to directly predict all bounding-box and classes in one-shot which making the model fast by design. Later YOLOv2 [11] and YOLOv3 were developed as improved versions. YOLOv3 is composed of (i) a backbone network for feature extraction, and (ii) feature pyramid network (FPN) for multi-scale detection. Even YOLOv3 is considered fast in terms of a deep learning-based model; however, the model is still overparameterized as it has over 10 million parameters and computational cost requirement is still beyond the capability of small devices. YOLOv3 with MobileNet backbone model structure [12] is shown in Fig. 1.

B. Compression Techniques

In 1990, LeCun, et al. [13] studied neural network model compression by pruning unnecessary weight. This work used second-order Taylor expansion to identify unnecessary weight. This method mainly focused on removing unnecessary weight but did not consider model structure. As a result, the model was damaged and unable to operate efficiently without requiring specialized deep learning software and hardware.

In 2017, Anwar, et al. [6] proposed structured pruning that aimed to prune model by removing unnecessary weight at the filter level from the model. As a result, model structure was not damaged, allowing the model able to operate on standard deep learning software and hardware.

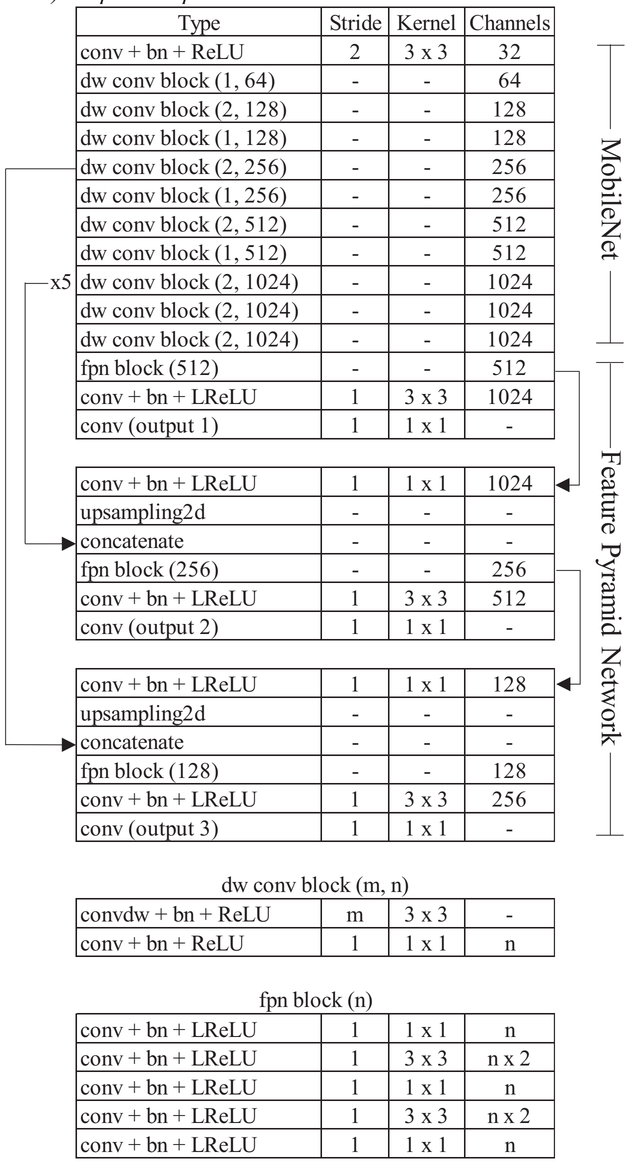
In 2015, Han, et al. [7] proposed an iterative pruning as an improvement to pruning procedure. This procedure aimed to prune model and fine-tune training model iteratively to minimize accuracy drop. The process consists of four main steps as following.

1) Filter selection: To correctly select a group of unnecessary convolutional filters and weights, the importantness value of each filter needs to be determined by a ranking method. Then, the group of unnecessary convolutional filters and weights are selected.

2) Remove the least important filter: Remove the group of unnecessary filters from the first step from the model.

3) Fine-tuning model: After removing unnecessary convolutional filters and weights from the model. The model weight values may shift from the optimal stage, which makes model accuracy decrease. To recover the model accuracy, fine-tune training is required to retrain the weight after the structure of the model is changed as a result of pruning.

4) Repeat step 1 – 3



Type	Stride	Kernel	Channels
conv + bn + ReLU	2	3 x 3	32
dw conv block (1, 64)	-	-	64
dw conv block (2, 128)	-	-	128
dw conv block (1, 128)	-	-	128
dw conv block (2, 256)	-	-	256
dw conv block (1, 256)	-	-	256
dw conv block (2, 512)	-	-	512
dw conv block (1, 512)	-	-	512
dw conv block (2, 1024)	-	-	1024
dw conv block (2, 1024)	-	-	1024
dw conv block (2, 1024)	-	-	1024
fpn block (512)	-	-	512
conv + bn + LReLU	1	3 x 3	1024
conv (output 1)	1	1 x 1	-

conv + bn + LReLU	1	1 x 1	1024
upsampling2d	-	-	-
concatenate	-	-	-
fpn block (256)	-	-	256
conv + bn + LReLU	1	3 x 3	512
conv (output 2)	1	1 x 1	-

conv + bn + LReLU	1	1 x 1	128
upsampling2d	-	-	-
concatenate	-	-	-
fpn block (128)	-	-	128
conv + bn + LReLU	1	3 x 3	256
conv (output 3)	1	1 x 1	-

dw conv block (m, n)

convdw + bn + ReLU	m	3 x 3	-
conv + bn + ReLU	1	1 x 1	n

fpn block (n)

conv + bn + LReLU	1	1 x 1	n
conv + bn + LReLU	1	3 x 3	n x 2
conv + bn + LReLU	1	1 x 1	n
conv + bn + LReLU	1	3 x 3	n x 2
conv + bn + LReLU	1	1 x 1	n

Fig. 1. YOLOv3 model structure

There are many works related to pruning criteria. Sum magnitude weight [14], this method determines unnecessary weight by the sum of magnitude of weight. This criterion is simple and has good enough performance. Average Percentage of Zero (APoZ) [15] is a data-driven ranking criterion that relies on the activation output value. This method determines unnecessary weight from the sparsity of activation output.

- **Magnitude**: Importantness value of the filter is calculated by the sum of the absolute value of filter weight. If the value is low means filter is unnecessary. Where i^{th} denotes the index of convolutional filter, $W()$ denotes an array of weight values of a convolutional layer, and N denotes the size of convolutional filter i^{th}.

$$\sum |W(i, :, :, :)| \tag{1}$$

- **APoZ (Average Percentage of Zeros)**: Importantness value of the filter is calculated from the activation layer output. A filter with high sparsity value means that the filter is unnecessary, where i^{th} denotes the index of convolutional filter, $I()$ denotes output value from activation function, *Sparsity*() denotes the percentage of zero outputs from the activation layer, and N denotes validation dataset size.

$$\frac{1}{N}\sum|Sparsity(I(i,:,:))| \qquad (2)$$

In 2017, [16] experiment pruning on VGG16 [17] and ResNet50 [18] models using various ranking criteria on object classification task, and successfully reduce the computational cost by a large amount. In 2018, [19] experiment pruning on object detection task and successfully reduce the computational cost of YOLOv2 more than four times while retaining almost the same accuracy. However, these works focus on a network that composes of one type of network.

In this paper, we use [14] [15] as the main ranking method. For the pruning procedure, we apply [7] as the main pruning technique since this method is designed to minimize accuracy drop. We use method [6] for pruning because this type of pruning has a major advantage over the unstructured pruning method as the pruned model remains compatible with commodity deep learning software and hardware. For object detection model, we select YOLOv3 as our baseline model as the model is more suitable for small devices as the computational cost is lower than a two-stage model such as R-CNN while also showing promising accuracy. However, YOLOv3 model is composed of two different parts, including, backbone and feature pyramid network. The standard pruning method often over-prunes some parts of the model which results in a large decrease in accuracy. To solve this problem, we propose a separated pruning to prevent pruning process from over-pruning some parts of the network.

III. Proposed Compression Method

From our investigation on YOLOv3, we found a range of weight values in backbone network part and feature pyramid network part is different. If we just prune the whole network, this can lead to over-pruning on one part of the network and cause a significant reduction in model accuracy.

Thus, we propose a robust pruning mechanism to solve the over-pruning problem. First, we explain the details of each component in our proposed method, and then we describe how our proposed method works.

A. Separated Pruning

In the pruning process, if a model is composed of multiple types of network, then there is a probability that the characteristic of weight in each part of the model is different. Therefore, if we prune the whole network altogether, this will result in over-pruning on some part of the network and cause a significant reduction in model accuracy. The over-pruning state is shown as the left network in Fig. 2 as weight in the backbone network is heavily pruned, whereas weight in the feature pyramid network part is rarely pruned.

To solve this problem, we propose a separated pruning method to prune each part of the network separately. YOLOv3 is composed of backbone part and feature pyramid part. First, this method starts pruning on the feature pyramid network part and then switches to prune the backbone network part until reaching its pruning limit.

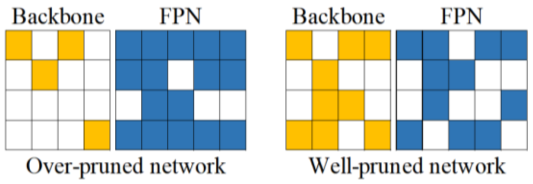


Fig. 2. Block filled with color represents the weight that is not already pruned. The yellow color represents the weight of the backbone network, while the blue color represents the weight of the feature pyramid network (FPN). Block with no color represents the weight that is already pruned. This image shows how over-pruning occurs in the network. An over-pruned network can be seen as the left network while a well-pruned network is shown in the right network.

To identify and measure layer differences, we use statistical measurement to capture different of weight characteristics in each layer of the network, including (i) mean, (ii) standard deviation, (iii) skewness, (iv) kurtosis, and (v) min-max range. From these measurements, we can group layers in YOLOv3 into two groups consisting of (i) backbone network group, and (ii) feature pyramid network group.

B. Minimum Filter Constraint

In the pruning process, when a model is pruned until reaching a high pruning ratio, there is a probability that some layers in the network are over-pruned. This causes the model does not have enough convolutional filters in some layers and significantly decreases model accuracy. The reason that we need to use this constraint when reaching a high pruning ratio because if we apply this constraint at a too early stage, this will make pruning process reserve too many filters than it should. That means some unnecessary filters might not be pruned.

The minimum filter constraint is the constraint that sets the reserve filter at a certain amount for each layer to prevent the layers from having insufficient weights after pruning.

C. Stopping Criteria

In our pruning mechanism, we use separated pruning to prune each part of the model separately. So, we need to define stopping criteria so that our pruning mechanism can decide when to stop the pruning process and select the best state of the model for each part of the model. Stopping criteria enable the pruning process to select the best model automatically.

We design our stopping criteria by adapting the patience concept. Our stopping criteria module is designed to monitor the validation accuracy of the pruned model in each state of pruning. Then, our criteria determine the best state to stop the process and select the best pruned model from the process. Our module will stop the pruning process if model accuracy starts to decrease. After the pruning process is stopped by the stopping criteria threshold, the module selects the highest pruning ratio model that validation accuracy does not start to decrease from the recent state.

D. Robust Pruning Mechanism

We propose a robust pruning mechanism. This mechanism composes of three main components, including (i) separated pruning (ii) minimum filter constraint, and (iii) stopping criteria. We use APoZ as the main ranking criterion in mechanism as APoZ outperforms other baseline ranking methods. Our pruning mechanism is designed to prevent an over-pruning problem. Pseudocode of the robust pruning mechanism is shown in Fig. 3.

Algorithm 1 Robust Pruning Mechanism

```
1: procedure RobustPruningMechanism(model)
2:     $groups \leftarrow getLayerGroup(model)$
3:     $modelHistory.append(model)$
4:     for each $group \in groups$ do
5:         while not isAccuracyDrop(model) do
6:             $filters \leftarrow getUnnecessaryFilters(group)$
7:             $model \leftarrow prune(model, filters)$
8:             $model \leftarrow finetune(model)$
9:             $modelHistory.append(model)$
10:        $model \leftarrow modelHistory.getBestModel()$
11:        $modelHistory.clearHistory()$
12:        $modelHistory.append(model)$
13:    while not isAccuracyDrop(model) do
14:        $filters \leftarrow getUnnecessaryFilterWithLimit(group)$
15:        $model \leftarrow prune(model, filters)$
16:        $model \leftarrow finetune(model)$
17:        $modelHistory.append(model)$
18:    $model \leftarrow modelHistory.getBestModel()$
19:    return $model$
```

Fig. 3. Pseudocode of the proposed robust compression mechanism.

The overall process of our robust pruning mechanism is visualized in Fig. 4. The process of YOLOv3 is grouped into three main stages. First, we perform iterative pruning on the feature pyramid network part of YOLOv3. After validation accuracy decreases to more than the stopping criteria threshold, the pruning process is stopped. Then, stopping criteria choose the pruning state with the best accuracy from the recent states as an initial model for the next pruning stage. Second, we switch to prune the backbone network part. We perform iterative pruning on the backbone part until model validation accuracy decreases to more than our stopping criteria threshold. The pruning process is stopped. Then, we choose the pruning state with the best accuracy from four recent states as the start model for the next pruning stage. Finally, before we start the final pruning stage, we set a minimum convolutional filter constraint on the process to prevent over-pruning in some layers; then we prune the backbone and feature pyramid network together until model accuracy decreases to more than the threshold of the stopping criteria. After that, we choose the pruning state with the best accuracy from four recent states as a final result.

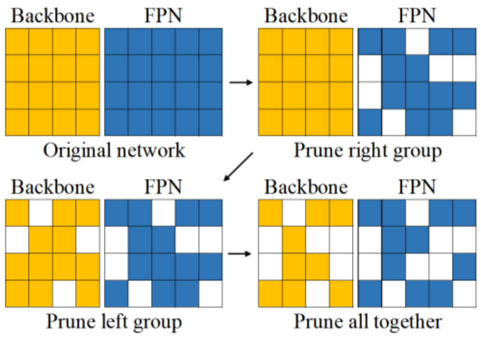


Fig. 4. Four stages of robust compression mechanism on YOLOv3, including (i) original network, (ii) prune right group (feature pyramid network), (iii) prune left group (backbone network), and (iv) prune all together (backbone network and feature pyramid network). The yellow color represents a weight of the backbone network, the blue color represents a weight of the feature pyramid network, and the blocks with no color represent pruned weights.

E. *APoZ Criterion Adjustment*

We adjust the APoZ criterion by changing the definition of sparsity from equal to zero to less than or equal to zero to make APoZ compatible with LeakyReLU activation layer in the feature pyramid network part.

IV. Experimental Setup

In this section, we describe how to setup our experiment: including dataset, baseline model, and ranking methods.

A. *UA-DETRAC*

This dataset consists of 10 hours of video recording at a road in a different location for vehicle detection and tracking task. The dataset provides ground truth annotation and ignored area for each image in the video. Besides, the dataset also provides the category of the vehicle as additional information for detection, including (i) car, (ii) van, (iii) bus, and (iv) others.

For preprocessing, we fill each ignored area of each image with black color. We also remove each ground truth annotation that intersects with the ignored area. These two preprocessing processes are done to prevent the model from training on incomplete image data and annotations that have a negative effect on the model.

B. *Baseline Model*

We use YOLOv3 with MobileNet backbone as our base model. Our backbone network is pre-trained on ImageNet [20] dataset. Then, we freeze all weights in the backbone network and only fine-tune training feature pyramid network on the UA-DETRAC dataset for 100 epochs. Finally, we train all layers together for another 100 epochs. Our baseline model achieves 63.68% accuracy on the UA-DETRAC dataset.

C. *Baseline ranking method*

- **Random**: Filters are selected randomly.
- **Magnitude**: Filters are selected from weight value.
- **APoZ**: Filters are selected from the percentage of zeros from activation output value.

V. Experimental Results

There are two main discussions in this section. First, we analyze and compare weight values in the backbone network and the feature pyramid network of YOLOv3. Second, we experiment pruning YOLOv3 using each method to compare our mechanism and baseline methods. We also investigate the pruning results in-detail to investigate the root cause of an over-pruning problem that occurs in standard pruning procedure.

A. *Define a Group of Layer for Group Pruning*

We group each convolutional layer in YOLOv3 network by statistical measurement, including (i) means, (ii) standard deviation, (iii) skewness, (iv) kurtosis, and (v) min-max range. As a result, we found that weight characteristics in the backbone network and feature pyramid network of YOLOv3 are different. From the statistical measurement as shown in Table 1, we found that the distribution of convolutional layer output value in the backbone network is more similar to a normal distribution as (i) skewness value is near zero and, (ii) kurtosis value is near three. On the other hand, the characteristic of the feature pyramid network weight is more different from normal distribution as kurtosis values are much higher than three. Moreover, the range of min-max value of backbone and feature pyramid network is different. The results from statistical measurements indicate that the backbone network weight and feature pyramid weight are different and should be treated separately in the pruning process. Distribution of convolutional layer output value is shown in Fig. 5 and Fig. 6. The statistical values are shown in Table 1.

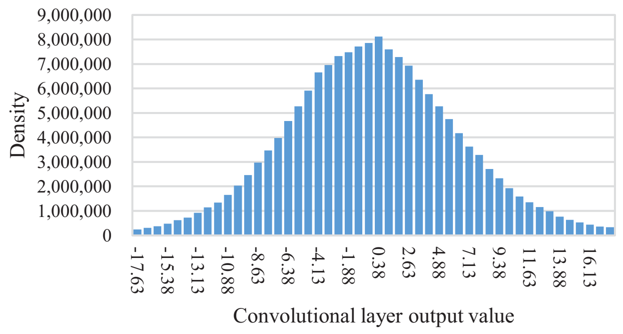


Fig. 5. Distribution of convolutional layer output values of the backbone network showing that network range is similar to a normal distribution

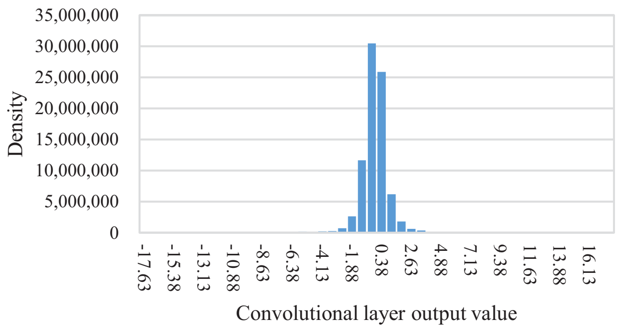


Fig. 6. Distribution of convolutional layer output values of the feature pyramid network showing that network range is smaller than network range in the backbone network and different from a normal distribution.

TABLE I. THE OVERALL STATISTICAL VALUE OF THE BACKBONE NETWORK AND FEATURE PYRAMID NETWORK CONVOLUTIONAL LAYER OUTPUT VALUES.

Name	Avg	Std	Skewness	Kurtosis	Min	Max
backbone	-0.0626	6.7125	0.1374	3.7700	-51.0073	51.8578
FPN	-0.1439	1.0944	-1.0982	26.1652	-17.2669	16.6356

B. *Comparison of Different Compression Techniques*

We compare our proposed mechanism to baseline criteria. We prune YOLOv3 using baseline criteria and our proposed mechanism on YOLOv3 on the UA-DETRAC dataset to evaluate the performance of our proposed mechanism.

In our experiment, we pruned YOLOv3 with difference ranking criteria, including (i) random, (ii) APoZ, (iii) Magnitude, and (iv) our proposed method.

The results on UA-DETRAC are shown in Fig. 7 and Table 2. From the results, we found a very interesting characteristic that occurs in the pruning process. That is there is a chance that model accuracy will increase during the pruning process.

From the results, our proposed method outperforms other baselines. Our method can retain model accuracy even after the model weight is removed more than 90% outperform other baseline methods.

The one interesting point is that when model accuracy starts to decrease, it does not slowly decrease when pruning the model from 0% to 80% pruning ratio. However, the result shows that when we prune the model iteratively until we reach a certain pruning ratio, model accuracy will decrease very quickly. This problem occurs when some part of the model is pruned until it reaches an over-pruned state where some layers of the model do not have sufficient weight parameters.

The random criterion shows a very interesting result. The model using the random ranking criterion to select filter performs better than our expectation. However, random criterion also has some disadvantages. First, the result of the random criterion suffers from larger fluctuations in model accuracy. Second, the random criterion is a non-deterministic method as we cannot predict the outcome which makes this criterion unreliable. Finally, random criterion max accuracy is lower than other baseline methods like APoZ and Magnitude because the random method does not consider the current optimal value of weight. Thus, the random method tends to cause significant damage to the knowledge that weights from the previous state have learned.

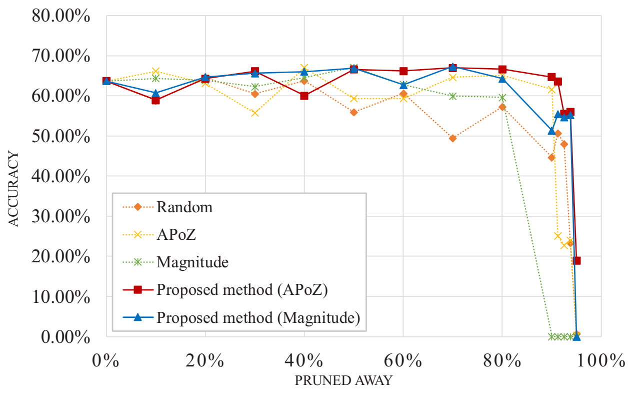


Fig. 7. Result of each ranking method

TABLE II. RESULT OF EACH RANKING METHOD.

Pruned away	Random	APoZ	Magnitude	**Proposed method (APoZ)**	Proposed method (Magnitude)
0.00%	63.68%	63.68%	63.68%	**63.68%**	63.68%
10.00%	58.82%	66.13%	64.36%	**58.92%**	60.73%
20.00%	64.59%	63.13%	63.84%	**64.28%**	64.70%
30.00%	60.47%	55.73%	62.29%	**66.15%**	65.66%
40.00%	63.73%	67.01%	64.56%	**60.04%**	66.04%
50.00%	55.92%	59.34%	67.06%	**66.57%**	66.84%
60.00%	60.60%	59.30%	62.76%	**66.22%**	62.71%
70.00%	49.45%	64.60%	59.94%	**66.97%**	67.30%
80.00%	57.25%	65.06%	59.63%	**66.62%**	64.31%
90.00%	44.62%	61.63%	0.00%	**64.70%**	51.33%
91.25%	50.67%	25.07%	0.00%	**63.57%**	55.41%
92.50%	47.98%	22.74%	0.00%	**55.55%**	54.70%
93.75%	23.34%	23.98%	0.00%	**56.00%**	55.27%
95.00%	0.63%	0.29%	0.00%	**18.94%**	0.00%

We also investigate how each part of the network is pruned in each state and when over-pruning occurs in detail. The detail about how much each part of the network is pruned in-detail is shown in Table 3.

From the results of our proposed method (APoZ) in Table 3, we found that the feature pyramid network part can be pruned up to about 92.32% before model accuracy starts to decrease significantly. On the other hand, the backbone network part can be pruned up to about 84.32% before model accuracy significantly decreases. From these results, we can determine the maximum pruning ratio of the backbone network part and feature pyramid network part. In the APoZ case, the model accuracy starts to significantly decrease after weights are removed by 91.25%. At this pruning ratio, we found that the feature pyramid network part is removed by 93.64%, higher than the upper limit we found at the proposed method (APoZ) result by 1.32%. That means the feature pyramid network part has already reached an over-pruned state. Thus, we can summarize that the cause of APoZ accuracy reduction at 91.25% pruning ratio is because the model is over-pruned on the feature pyramid network part.

TABLE III. DETAILED PRUNING RESULTS. BACKBONE NETWORK CONTAINS 13.34% OF OVERALL PARAMETERS, AND FPN CONTAINS 86.66% OF OVERALL PARAMETERS.

Pruned away	**APoZ**	Backbone (13.34%)	FPN (86.66%)	**Proposed method (APoZ)**	Backbone (13.34%)	FPN (86.66%)
0.00%	**63.68%**	0.00%	0.00%	**63.68%**	0.00%	0.00%
10.00%	**66.13%**	26.29%	7.49%	**58.92%**	0.00%	11.54%
20.00%	**63.13%**	26.45%	19.01%	**64.28%**	0.00%	23.08%
30.00%	**55.73%**	26.45%	30.55%	**66.15%**	0.00%	34.62%
40.00%	**67.01%**	27.23%	41.97%	**60.04%**	0.00%	46.16%
50.00%	**59.34%**	29.28%	53.19%	**66.57%**	0.00%	57.70%
60.00%	**59.30%**	29.93%	64.63%	**66.22%**	0.00%	69.24%
70.00%	**64.60%**	30.42%	76.09%	**66.97%**	0.00%	80.78%
80.00%	**65.06%**	31.59%	87.45%	**66.62%**	0.00%	92.32%
90.00%	**61.63%**	68.90%	93.25%	**64.70%**	74.95%	92.32%
91.25%	**25.07%**	75.70%	93.64%	**63.57%**	84.32%	92.32%
92.50%	**22.74%**	81.77%	94.15%	**55.55%**	93.69%	92.32%
93.75%	**23.98%**	86.27%	94.90%	**56.00%**	93.69%	93.76%
95.00%	**0.29%**	92.75%	95.35%	**18.94%**	95.04%	94.99%

VI. CONCLUSION

In this paper, we propose the robust compression mechanism for YOLOv3 that is designed to prevent a model from over-pruning on some part of the model network to maximize pruning ratio and model accuracy. Our proposed method composes of three main components, including (i) separated pruning, (ii) minimum filter constraint, and (iii) stopping criteria. We evaluate our proposed method on a UA-DETRAC dataset. Our proposed method can retain almost the same accuracy as the original model at 64.70% after 90.00% of weight is removed from the model that outperforms other baselines by 2%.

REFERENCES

[1] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence,* vol. 32, no. 9, pp. 1627-1645, 2009.

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 2005, vol. 1, pp. 886-893: IEEE.

[3] 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*, 2015, pp. 91-99.

[4] 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*, 2016, pp. 779-788.

[5] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767,* 2018.

[6] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *ACM Journal on Emerging Technologies in Computing Systems (JETC),* vol. 13, no. 3, p. 32, 2017.

[7] 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*, 2015, pp. 1135-1143.

[8] L. Wen *et al.*, "UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking," *arXiv preprint arXiv:1511.04136,* 2015.

[9] 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*, 2014, pp. 580-587.

[10] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440-1448.

[11] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *arXiv preprint,* 2017.

[12] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861,* 2017.

[13] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in neural information processing systems*, 1990, pp. 598-605.

[14] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710,* 2016.

[15] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," *arXiv preprint arXiv:1607.03250,* 2016.

[16] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5058-5066.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556,* 2014.

[18] 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*, 2016, pp. 770-778.

[19] S. O'Keeffe and R. Villing, "Evaluating pruned object detection networks for real-time robot vision," in *2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2018, pp. 91-96: IEEE.

[20] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision,* vol. 115, no. 3, pp. 211-252, 2015.