Optimization Model Performance through Pruning Techniques

Hanzhang Tang

[1] American International School, HongKong, 999077, China

221115@ais.edu.hk

Abstract. Due to their ability to automatically extract features, Deep Neural Networks (DNNS) have demonstrated performance never before seen. Due to this high degree of performance during the past ten years, a considerable number of DNN models have been combined with various Internet of Things (IoT) applications. However, deploying DNN models on resource-constrained IoT devices is impractical because of the high computing, energy, and storage needs of these models. Because of this, several pruning approaches have been put out recently to lessen the storage and processing needs of DNN models. These DNN pruning methods take a new approach to condense the DNN while lowering accuracy. It motivates us to present a thorough analysis of deep neural network compression methods. In order to decrease storage and computing requirements, A thorough analysis of the current literature pruning techniques will be given. The currently used strategies are into three groups are categorized as layer, channel, filter, and connection pruning. The difficulties that come with each class of DNN pruning strategies are also covered in the study. Finally, a brief summary of the ongoing work in each classification is provided, along with a projection of network pruning's future evolution.

Keywords: Pruning, Machine Learning, Optimization Method.

1. Introduction

Machine learning is widely used in today's society consisting of data prediction, image recognition, machine translation, speech recognition, recommendation systems, etc.. One of the sustainable models is the Deep Neural Network (DNN). This model is used in different parts of people's lives, such as home automation, agriculture, and motion pattern recognition. This is because DNN contains a high ability to extract data from huge amounts of parameters. DNN is a combination of Convolutional Neural Networks (CNN) extracted spatial features and Recurrent Neural Networks (RNN) identified temporal features from the datasets [1]. Even with all the advantages of DNN, it needs resources, including energy, processing capacity, and storage, which will cause the time usage and workload to increase to a huge number.

B. H. Ahmad (ed.), *Proceedings of the 2023 International Conference on Data Science, Advanced Algorithm and Intelligent Computing (DAI 2023)*, Advances in Intelligent Systems Research 180,
https://doi.org/10.2991/978-94-6463-370-2_6

Since access to the small computational device was easy and convenient, the workload of machine learning increased. In the past decades, the use of the internet has become easy, the meantime, the data increased at a huge rate. Just in the time of reading one sentence, 10^18 bytes of data have just been added to the word's store [2], and modern neural networks have grown to reach billions of parameters. In such a high parameter quantity, the memory, hardware, and inference time are getting increasingly high to be able to compress. At this stage, optimization plays an inconvenient role in solving the problem. Network Pruning is one of the methods widely used. There are two main types of pruning, structured pruning and unstructured pruning. Just like the name, structural pruning is eliminating a group of parameters or a whole structure for example neurons, channels, or filters [3]. Structure pruning is suitable for structural networks such as convolutional neural networks because it helps to retain the concealed structure of the network. This is usually easier to do because the structure of the model is perverse. Unstructured pruning is on the opposite side of structured pruning, in which unstructured pruning only cuts single parameters and doesn't need to consider the structure of the network. The benefit of using unstructured pruning is that it is usually more effective because it allows more fine-grained pruning, on the other hand, it is more complex.

Optimization methods include thousands of different techniques, this essay aims to research pruning techniques' usage to optimize machine learning models' performance through pruning techniques. In the following parts, there are several different categories that this essay will reach. In the next part, there will be literature reviews to understand background information and previous research done on the criteria. The third part of the essay will be methodology, which is what is the algorithm of different pruning methods, visualization of the method, and important techniques. The fourth part will include some practical applications and cases.

2. Literature Review

Currently, in the area of optimization methods, the main type includes sparse representation, knowledge distillation, and network pruning.

2.1. Spare Representation

Sparse representation takes advantage of the sparsity present in the weight matrices of a deep neural network (DNN) model [4]. This approach involves removing the weights that are zero or close to zero from the weight matrix, reducing the storage and computational requirements of the DNN model. In other words, it combines the network's links with similar weights into multiplexed links, where a single weight replaces multiple weights on a single link [1]. Sparse representation encompasses techniques such as low-rank estimations, quantization, and multiplexing. The primary objective of sparse representation is to compact the weight matrix while maintaining the performance of the DNN model [5].

2.2. Knowledge Distillation

Knowledge distillation is the term used to describe the procedure of transferring the ability to generalize from a complex model (referred to as the "teacher") to a simpler model (known as the "student") within the framework of a deep neural network (DNN) model [6]. This technique provides a solution to mitigate the loss of accuracy that occurs during DNN compression. By employing knowledge distillation during the training of the student model, it becomes possible to mimic the behavior exhibited by the teacher model in predicting the probabilities of class labels [5].

2.3. Network Pruning

Deep network pruning is a widely adopted technique for reducing the size of a deep learning model by eliminating ineffective channels, filters, neurons, or layers, resulting in a lightweight model [7]. The resulting lightweight model has reduced memory requirements, lower power consumption, and enables faster inference while minimizing the loss of accuracy. The extent of the accuracy loss when a component is removed determines its suitability. Pruning is often seen as a binary criterion for deciding whether to keep or discard a component in a deep neural network (DNN). The pruning process involves iteratively removing underperforming components from a pre-trained model. The network pruning methods can be categorized into four groups: layer pruning, connection pruning, filter pruning, and channel pruning. These methods contribute to reducing the storage and computational demands of the DNN model [5].

3. Methodology

The network pruning strategies for DNN compression are covered in this section. The main concept is to strip the original DNN model of unnecessary elements like layers, filters, channels, etc. A compressed DNN model is created from the remaining elements. In comparison to the original DNN model, the compressed model uses less processing power and less storage. Additionally, the compressed model is tuned by training it on the existing dataset for an enormous number of epochs. Reducing the time required for DNN model fine-tuning and limiting the accuracy compromise caused by network compression are the two main research problems related to network pruning.

3.1. Channel Pruning

The basis of channel pruning is to reduce the number of channels in the input supplied to DNN's intermediate layer [8]. The data provided to the DNN model are first channelized to create the proper input, for example, a picture has three channels (RGB). The channels improve the performance of DNN, and every layer contains different channels. However, these channels will increase calculation and memory, whereas reducing

channels could help to decrease the calculation and memory needed [5]. Fig 1 represents the process of Channel Pruning, by deleting channel data will go through, the model could be optimized.

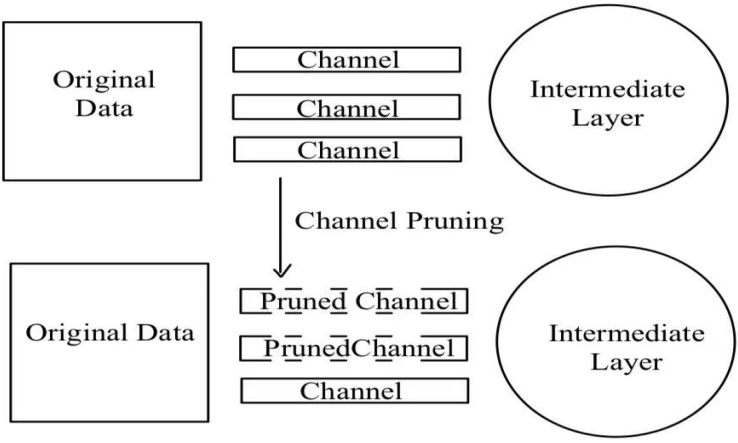

Fig 1. This is a graph of how the channel pruning works (Picture: Original).

3.2. Filter Pruning

A large number of filters are involved in the convolutional operation of CNN in order to improve the performance of the model in different processes (regression, classification, and prediction). The filter could help the model divide data into different groups. According to the presumption, increasing the number of filters enhances the distinguishing qualities of the spatial information produced by the CNN model [9]. However, this increase in convolutional filters causes the DNN model to perform a significantly greater amount of floating-point calculations. Therefore, removing the unnecessary filters is crucial to lowering the processing demands of the DNN model [5]. As shown in Fig 2, sets of data will be divided into different groups by going through different filters. By decreasing the number of filters, the model could be optimized.

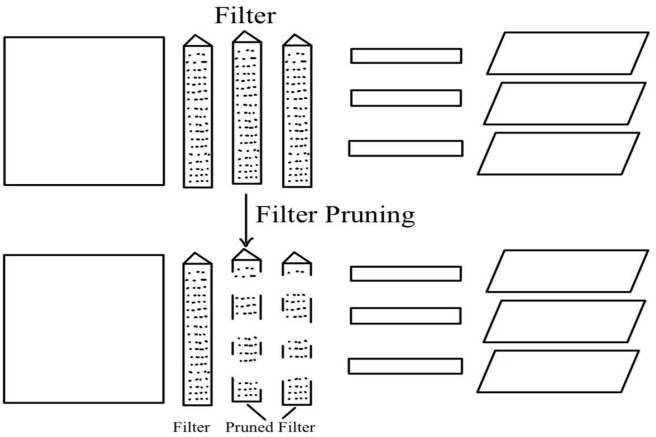

Fig 2. This is a graph of how the filter pruning works. (Picture: Original)

3.3. Connection Pruning

The number of input and output connections to that layer determines the number of parameters for a layer of a DNN model. These variables can be used to calculate the DNN model's storage and computation needs [10]. Due to the fact that the DNN model needs a lot of parameters to function, it is practical to cut out unnecessary connections between the various levels of the DNN model[11]. Fig 3 is a picture visualizing the process of connection pruning. Connections that are unnecessary will be cut off.

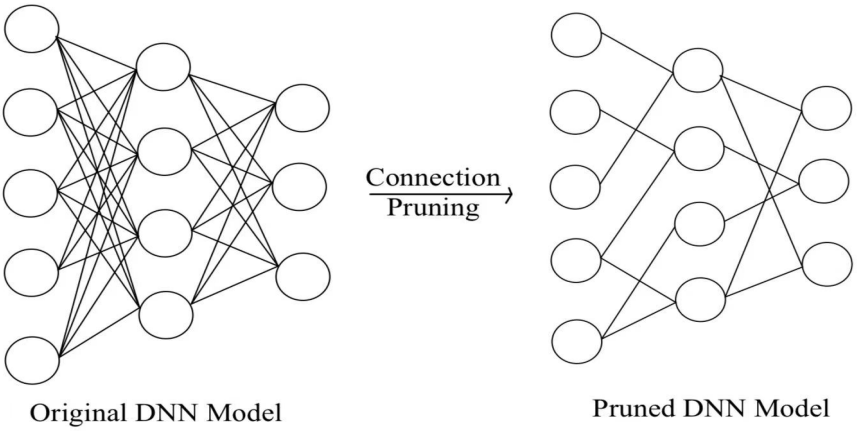

Fig 3. This is an example of the connection pruning structure (Picture: Original).

3.4. Layer Pruning

Layer pruning is the final group of network pruning approaches, in which some DNN model layers are compressed by removing a few selected layers from the network. When deploying a DNN model on a small computational device, when ultra-high DNN model compression is required, layer pruning is heavily used[12]. The loss of the semantic structure of the DNN model, which produces low-quality features, is the main problem with layer pruning. The performance is inefficient as a result of these poor features. Fig 4 shows the overall process of layer pruning. When there are three layers, the parameters are super big. After selecting inefficient or less useful layers, these layers will be pruned in order to reduce the size of the parameters.

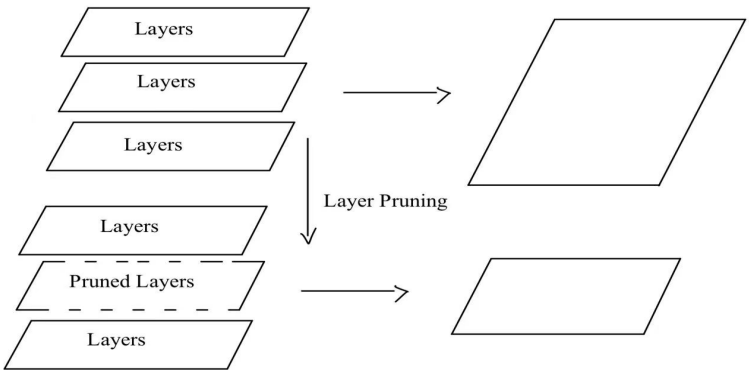

Fig 4. This picture shows the process of Layer Pruning (Picture: Original).

3.5. Application Fields

Model pruning is a technique for optimizing deep learning models by removing redundant parameters and connections, thereby reducing model size, accelerating inference, and lowering energy consumption [13]. It has found widespread applications in various domains, including computer vision, natural language processing, and recommendation systems. The following presents cases of utilizing model pruning in these domains.

Computer Vision:

Image Classification: In image classification tasks, model pruning can be applied to reduce the size of convolutional neural networks by eliminating redundant neurons and channels [14]. For instance, channel pruning methods can be employed to remove unimportant channels, thereby reducing computational overhead and parameter count, and consequently speeding up inference.

Object Detection: In object detection, model pruning can be applied to the detection head, encompassing classifiers and regressors in architectures like Faster R-CNN or YOLO [15]. Pruning can decrease detection model latency, making it suitable for real-time applications.

Natural Language Processing:

Text Classification: In text classification tasks such as sentiment analysis, model pruning techniques can downsize recurrent neural networks (RNNs) or Transformer

models [16]. Pruning can involve the removal of non-essential vocabulary and attention heads, thereby mitigating computational costs.

Machine Translation: In machine translation tasks, Transformer models often consist of numerous self-attention heads and parameters [17]. Pruning can involve the removal of unnecessary self-attention heads, reducing model size, and enhancing inference speed.

Recommendation Systems:

Recommendation Models: Models in recommendation systems are typically complex neural networks used to predict user preferences for items. Model pruning can involve eliminating features related to less popular items, reducing model complexity while retaining features crucial for predicting user preferences to expedite the recommendation process [18].

In conclusion, model pruning holds promise across various application domains, with the potential to optimize model performance by decreasing size, accelerating inference speed, and lowering energy consumption. However, it is important to strike a balance between model size and performance during pruning to ensure that accuracy degradation remains minimal.

4. Application and Case Study

This section uses a case study to showcase the effects of different pruning methods on accuracy, model size, and inference speed. We utilize Table xxx to report the performance of various pruning algorithms. For the purpose of comparison, we employ ResNet-50 as the base network and present the pruning outcomes of different algorithms on the ILSVRC-2012 dataset. ILSVRC-2012, also known as ImageNet 2012, is a competition dataset consisting of a total of 1000 categories. The training set of ILSVRC-2012 comprises 1,281,167 images, with the number of training images per category ranging from 732 to 1300. Additionally, its validation set consists of 50,000 images, where each category is evenly represented with 50 validation images. Due to the utilization of distinct deep learning frameworks among the diverse algorithms, the baseline results of the models may vary. Hence, we compare their changes relative to the baseline results. We present our experimental results comparing importance-based pruning methods and gradient-based pruning methods in Table 1.

Since pruning algorithms are essentially validated through experiments conducted on the same dataset and model, comparing different pruning algorithms is straightforward. As shown in Table 1 [19], regardless of the specific pruning algorithm, the pruned models can be regarded as sub-models of the original model. These sub-models inherit certain parameters from the original model, followed by a certain degree of fine-tuning training. This approach achieves comparable or even superior classification results to the original model while utilizing fewer parameters and computational resources.

Table 1. Performance of different pruning algorithms on ResNet-50 [19]

Gradient-Based Pruning Method	Methods	Model FLOPs	Pruning Percentage(%)	Top 1 Submodel Accuracy Rate(%)	Accuracy Changes (%)
	SFP	8.17	0	76.15	+0.00
		4.75	42	74.61	-1.54
	FPGM	8.17	0	76.15	+0.00
		4.74	42	75.59	-0.54
		3.80	53	74.83	-1.32
	HRank	8.17	0	76.15	+0.00
		4.60	44	74.98	-0.17
		3.10	62	71.01	-4.17
		1.96	76	69.10	-7.05
	ThiNet	7.72	0	72.88	+0.00
		4.88	37	72.04	-0.84
		3.41	56	71.01	-1.87
		2.20	76	68.42	-4.46
	Entropy	7.72	0	72.88	+0.00
		7.16	7	73.56	+0.68
		6.38	17	72.89	+0.01
		5.04	35	70.84	-2.04
	CURL	8.17	9	76.15	+0.00
		2.22	73	73.39	-2.76
	EagleEye	8.17	0	76.60	+0.00
		6.00	27	77.10	+0.50
		4.00	51	76.40	-0.20
		2.00	76	74.20	-2.40
	LFPC	8.17	0	76.15	+0.00
		3.20	61	74.46	-1.69
	ABCPruner	8.17	0	76.01	+0.00
		5.12	37	74.84	-1.72
		2.60	68	72.58	-3.42
		1.88	77	70.29	-5.72
	Meta Pruning	8.17	0	76.60	+0.00
		6.00	27	76.20	-0.40
		4.00	51	75.40	-1.20
		2.00	76	73.40	-3.20
	TAS	8.17	0	77.46	+0.00
		2.31	72	76.20	-1.26
	AutoSlim	8.17	0	76.15	+0.00
		6.99	27	76.10	-0.15
		4.00	51	75.60	-0.55
		2.00	76	74.00	-2.15

DMCP	8.17	0	76.60	+0.00
	5.60	31	77.00	+0.40
	4.40	46	76.20	-0.40
	2.20	73	74.40	-2.20

4.1. Benefits and Negatives

The use of reinforcement learning-based neural network structure search in place of manual design was originally introduced by Zoph and Le, despite the fact that the search process consumes a lot of processing power [20]. Gradient-based search techniques started to gain popularity after DARTS was developed because of their speed advantages [21]. At this point, researchers started to characterize structured pruning as a search process based on prototype networks, where the sub-models discovered through searches on prototype networks serve as the representation of the trimmed small models. We will give a succinct description of a few search-based pruning techniques in this section.

AMC began using reinforcement learning techniques in 2018 to look for pruning structures [22]. AMC uses states to encode data like input feature dimensions, convolution kernel sizes, and strides for each layer. Actions are predicted by the DDPG agent, with classification errors and FLOPs being rewarded [23].

The artificial bee colony algorithm is used by ABCPruner to identify the best pruning structure [24]. Instead of picking comparably essential channels for pruning, ABCPruner searches for the number of channels in each layer. A feasible solution is defined specifically as the number of channels corresponding to each layer, and the target population is made up of all feasible solutions. The artificial bee colony algorithm is utilized to conduct the search, and the sub-model's performance on the dataset is directly used to determine its fitness. On the ILSVRC-2012 dataset, ABCPruner can keep ResNet-152's accuracy while reducing 62.87% FLOPs and 60% parameter count.

For automatic channel pruning, MetaPruning makes use of meta-learning [25]. In order to find effective pruning models, MetaPruning first trains a meta-network and then uses an evolutionary process. Instead of identifying each individual channel, MetaPruning looks for the aggregate number of channels for each layer, considerably condensing the search field. On ILSVRC-2012, the Top-1 accuracy of MetaPruning is only 0.4% less accurate than the original model after eliminating 25% FLOPs of the ResNet-50 model.

NetworkAdjustment calculates the FLOPs utilization rate for each layer by using model correctness as an equation relating to computational complexity (FLOPs) [26]. Then, based on this utilization rate, the channel numbers are automatically changed for each layer. Although searches using evolutionary algorithms have improved the search process, they still take a lot of time. In DMCP, pruning is modeled as a differentiable Markov process that is then directly optimized on the network via gradient descent [27]. A separate Markov process is created by the pruning construction for each layer's chan-

nels, where changes in the channel count represent state transitions in the Markov process. On the ILSVRC-2012 dataset, DMCP can cut ResNet-50's FLOPs by around 46% while still producing a pruned model with a 0.4% drop in accuracy.

5. Future Improvements and Challenges

The current structured pruning algorithms are primarily studied in the context of image classification tasks. However, due to the versatility of neural network architectures, pruning algorithms that have proven effective in classification tasks can be conveniently transferred to other visual tasks.

With the rise of the Internet of Things (IoT) and the development of smartphones, an increasing number of mobile devices are equipped with deep learning models to provide smarter services. However, these mobile devices often have limited computational power and storage space, making lightweight models increasingly important. Structured pruning is an important and effective method for model compression, often used in conjunction with other methods such as parameter quantization and low-rank approximation to maximize model compression. Unstructured pruning disrupts the original structure of the model, leading to significant reductions in parameter count and theoretical computational load. However, the actual runtime speed might not be optimistic. In contrast to unstructured pruning, structured pruning algorithms can run on general platforms and, since they preserve the original model structure, can perform low-rank approximation on top of pruning, which is not achievable with unstructured pruning.

Nevertheless, structured pruning also comes with certain challenges. Typically, it involves a three-step process of evaluation, selection, and fine-tuning to obtain a compact model. In some cases, layer-wise selection and fine-tuning are performed, resulting in a significant time investment in obtaining a pruned model. Additionally, the majority of research in this area has focused on image classification, and there is relatively limited progress in more challenging visual tasks such as object detection and semantic segmentation.

6. Conclusion

In recent years, while deep learning models have achieved significant accomplishments, they have also brought about substantial computational and storage overhead. Model pruning, as an optimization method, aims to enhance the efficiency and inference speed of models by reducing redundant and unnecessary parameters.

According to numerous research in the literature, pruning has advanced significantly in the area of network acceleration. The two main categories of mainstream pruning techniques are structural pruning and unstructured pruning. By physically removing a group of parameters, structural pruning seeks to condense neural networks. Unstructured pruning, in contrast, zeros out particular weights without changing the network's structure. Unstructured pruning in particular is simple to use in practice and naturally adaptive to different networks. Model acceleration frequently requires specialized AI

accelerators or software. On the other hand, structural pruning finds a larger range of applications by reducing the inference overhead by physically deleting parameters from networks. In the literature, The design space of pruning algorithms encompasses a range of aspects, including pruning schemes, parameter selection, layer sparsity, and training techniques. Model pruning finds widespread applications across various domains, including but not limited to model compression, acceleration of deep learning, deployment on mobile devices, and edge computing.

This review comprehensively summarizes the developmental trajectory, different techniques, and application domains of deep learning model pruning. The importance of model pruning in optimizing deep learning models is emphasized, along with its advantages in improving efficiency and reducing computational costs. Through this review, researchers can gain an understanding of the evolution of model pruning techniques, select appropriate pruning methods, and apply pruning techniques in specific domains, thereby providing comprehensive guidance for the optimization of deep learning models.

Reference

1. Y. Zhao, et al, "Low-rank plus diagonal adaptation for deep neural networks," in 2016 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 5005-5009, (2016).
2. J.V. Guttag, "Introduction to Computation and Programming Using Python, third edition: With Application to Computational Modeling and Understanding Data, " MIT Press, (2021).
3. C. Wang, et al, "EigenDamage: Structured Pruning in the Kronecker-Factored Eigenbasis," ArXiv, (2019).
4. Y. Guo, et al, "Sparse DNNs with Improved Adversarial Robustness," in Advances in Neural Information Processing Systems, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, vol. 31, Curran Associates, Inc., (2018).
5. R. Mishra, et al, "A Survey on Deep Neural Network Compression: Challenges, Overview, and Solutions," ArXiv, October (2020).
6. X. Liu, et al, "Improving the Interpretability of Deep Neural Networks with Knowledge Distillation," in 2018 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 905-912, (2018).
7. S. Anwar, et al, "Structured Pruning of Deep Convolutional Neural Networks," CoRR, vol. abs/1512.08571, (2015).
8. H. Peng, et al, "Collaborative Channel Pruning for Deep Networks." In Proceedings of the 36th International Conference on Machine Learning, pp. 5113-5122, (2019).
9. P. Singh, et al, "Stability Based Filter Pruning for Accelerating Deep CNNs." In 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1166-1174, (2019).
10. R. Mishra, et al, "A Road Health Monitoring System Using Sensors in Optimal Deep Neural Network," in IEEE Sensors Journal, vol. 21, no. 14, pp. 15527-15534, (2021).
11. N.T. Siebel, et al, "Efficient neural network pruning during neuro-evolution." In 2009 International Joint Conference on Neural Networks, pp. 2920-2927, (2009).
12. N. Liu, et al, "AutoCompress: An Automatic DNN Structured Pruning Framework for Ultra-High Compression Rates." In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 04, pp. 4876-4883, Apr. (2020).

13. T. Liang, et al, "Pruning and quantization for deep neural network acceleration: A survey." Neurocomputing, vol. 461, pp. 370-403, (2021).
14. Z. Liu, et al, "Rethinking the Value of Network Pruning," CoRR, vol. abs/1810.05270, (2018).
15. Y. Cai, et al., "Yolobile: Real-time Object Detection on Mobile Devices via Compression-Compilation Co-design." In Proceedings of the AAAI conference on artificial intelligence, Vol. 35, No. 2, pp. 955-963 (2021).
16. M. Gupta, et al, "Compression of Deep Learning Models for NLP." In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pages 3507–3508, (2020).
17. M. Behnke, et al, "Losing heads in the lottery: Pruning transformer attention in neural machine translation." In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 2664-2674 (2020).
18. Y. Sun, et al., "FM2: Field-matrixed factorization machines for recommender systems." In Proceedings of the Web Conference, pp. 2828-2837 (2021).
19. J. Luo, et al, "AutoPruner: An end-to-end trainable filter pruning method for efficient deep model inference." Pattern Recognition, 107:107461, (2020).
20. B. Zoph, et al, "Neural Architecture Search with Reinforcement Learning." arXiv preprint arXiv:1611.01578 (2017).
21. H. Liu, et al, "DARTS: Differentiable Architecture Search." arXiv preprint arXiv:1806.09055 (2019).
22. Y. He, et al, "AMC: AutoML for Model Compression and Acceleration on Mobile Devices." In Proceedings of the Computer Vision -- ECCV 2018, pp. 815-832, Springer International Publishing, (2018).
23. T. P. Lillicrap, et al, "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971 (2019).
24. M. Lin, et al. "Channel pruning via automatic structure search." arXiv preprint arXiv:2001.08565 (2020).
25. Z. Liu, et al., "MetaPruning: Meta-learning for automatic neural network channel pruning." In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 3296–3305 (2019).
26. Z. Chen, et al., "Network adjustment: Channel search guided by FLOPs utilization ratio." In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.10658–10667 (2020).
27. S. Guo, et al., "DMCP: Differentiable Markov channel pruning for neural networks." In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.1539– 1547 (2020).