Learning Pruned Structure and Weights Simultaneously from Scratch: an Attention based Approach

Qisheng He  
Department of Computer Science  
Wayne State University  
Detroit, USA  
Qisheng.He@wayne.edu

Weisong Shi  
Department of Computer Science  
Wayne State University  
Detroit, USA  
weisong@wayne.edu

Ming Dong  
Department of Computer Science  
Wayne State University  
Detroit, USA  
mdong@wayne.edu

Abstract—As a deep learning model typically contains millions of trainable weights, there has been a growing demand for a more efficient network structure with reduced storage space and improved run-time efficiency. Pruning is one of the most popular network compression techniques. In this paper, we propose a novel unstructured pruning pipeline, Attention-based Simultaneous sparse structure and Weight Learning (ASWL). In ASWL, an efficient algorithm is proposed to calculate the pruning ratios layer-wisely from attentions, and both weights for the dense network and the sparse network are tracked so that the pruned structure is simultaneously learned from randomly initialized weights. Our experiments on MNIST, Cifar10, and ImageNet show that ASWL achieves superior pruning results in terms of accuracy, pruning ratio and operating efficiency when compared with state-of-the-art network pruning methods.

Index Terms—Deep Learning, Deep Neural Network Compression, Network Pruning

I. INTRODUCTION

Deep learning models such as Convolutional Neural Network (CNN) have achieved great success in various computer vision tasks. A key avenue for deploying deep learning models is a mobile device or an edge server in order to reduce latency and ensure data privacy for users. As a deep learning model typically contains millions of trainable weights, this practice has been accompanied by a growing demand for a more efficient network structure with reduced storage space and improved run-time operating efficiency.

Recently, there has been a resurgence in neural network compression techniques (e.g., pruning and quantization) [6]. One can compress a given neural network architecture into an extremely small size without compromising on the model performance. Even efficient like [17] networks can be further compressed even though they already have a small footprint.

Pruning, which removes connections or neurons in a network, is one of the most popular network compression techniques. It reduces both the size and the computational complexity of a model since pruned weights do not need to be calculated or stored. Moreover, pruning is size-efficient as deep sparse models consistently outperform shallow dense models with almost no loss of or even a greater accuracy [24].

When network pruning was first introduced for compression, redundant connections were pruned using a three-step pipeline illustrated in Fig. 1(a): first, the network is trained to learn which connections are important; next, the unimportant connections are pruned; finally, the network is retrained to fine-tune the weights of the remaining connections [7].

Research is partially supported by the National Cancer Institute of the National Institutes of Health (NIH R01HL153720; Subcontract PI: Dong). The content is solely the responsibility of the authors and does not necessarily represent the official view of the NIH.
pruning in a layer-wise optimal brain surgeon [4].

It was shown that structure learning and weights optimization could be combined to simplify the pruning process as Fig 1(b) shows. Fine-pruning [13] proposed a principled method in which the pruning ratio of each layer is first predicted by the Bayesian optimization with a pre-trained network, and then the network is jointly fine-tuned and compressed with the predicted pruning ratios. The joint training was frequently used in quantization methods known as the Straight-Through Estimator (STE) [23] to avoid zero gradients during backpropagation. The advantage of such a pipeline is that it allows the pruning status to adapt during the change of weights [21]. However, pruning with pre-training is a time-consuming procedure. One needs to repeat pruning and fine-tuning several times to find a good sparse configuration [3].

Later work, therefore, shows that the weights of a model and its pruned structure can be directly learned from randomly initialized weights (Fig. 1(c)). Pruning from Scratch [22] proposed a novel network pruning pipeline that first learns the pruned structure directly from randomly initialized weights and then optimizes the weights of the pruned network. This kind of pipeline bypasses the time-consuming pre-training procedure. Nevertheless, the structure learning and weights optimization proceed separately here so that it only prunes weights based on fixed regularization before training the weights.

Recently, the attention mechanism was brought into network pruning and achieved great success [5]. In this paper, we propose a new pruning pipeline, Attention-based Simultaneous sparse structure and weight Learning (ASWL). As shown in Fig. 1(d), in ASWL, the layer-wise sparsity and weights are jointly learned from scratch in a unified training pipeline. Specifically, we first use the attention mechanism to learn the importance of each layer in a network and determine the corresponding pruning ratio. Then, we jointly prune the layer under the guidance of the pruning ratio and update the unpruned weights. The major contributions of our work are summarized as follows:

- ASWL provides a unified framework that combines both layer-wise pruning and weights optimization to learn a pruned network from randomly initialized weights. During ASWL training, both weights of the dense network and the sparse network are tracked so that the pruning ratio for each layer is simultaneously learned as the weights change.
- In ASWL, layer-wise pruning decisions are made through a novel attention-based approach. Attention scales in the attention mechanism are set for each layer instead of previous channel-wise or weight-wise attention mechanism. A pruning ratio is then directly computed based on the learned attention value for each layer.
- ASWL no longer needs the time-consuming pre-training procedure in network pruning but provides equivalent pruning results. Through extensive experiments on benchmark datasets, we demonstrate that ASWL leads to superior pruning results in terms of accuracy, network size, and operating efficiency when compared with state-of-the-art pruning methods.

II. ATTENTION-BASED SIMULTANEOUS LEARNING

The objective of our new training pipeline, Attention-based Simultaneous sparse structure and weight Learning (ASWL), is to train both pruned structure and model weights from randomly initialized weights. Different from existing pruning methods, ASWL obtains layer-wise pruned deep learning models without pre-training. As shown in Fig. 2, we first convert a model into an attention-based model by replacing its traditional weighted layers with attention-based weighted layers. Instead of learning the pruned structure by regrowing connections after a fixed number of iterations, we pruned the weights layer-wise based on the pruning ratio calculated from the learned attention of the target layer in every iteration. In the mean time, the unpruned parameters are tracked and learned directly from the forward pass of the pruned structure. Such simultaneous learning tracks both pruned and unpruned weights during training and thus gives an adaptive pruning decision on the sparse structure.

A. Attention-based Neural Networks

The sparsity of a pruned network is weakly related to its output and hence is hardly to be learned directly [12]. Recent research shows that the attention mechanism provides a promising solution [5]. Unlike the traditional methods which set attention scalars channel-wisely or weight-wisely, we introduced a much simpler layer-wise attention mechanism, where an attention scalar value was defined across all $L$ layers in our attention-based neural network. Such model is denoted as $f(x; w, A)$ where $x$ is the input of the model, $w$ is the traditional trainable weights, and $A = \{a_1, a_2, ..., a_L\}$ is the layer-wise attention values. Given a scalar gate value $a_l \in (0, 1]$ for the $l^{th}$ layer as the attention, it is multiplied with the output of the layer. That is, assuming the output of the $l^{th}$ original layer is $f_l(x_{l-1}; w_l)$, the attended output is modified as $f((x_{l-1}; w_l, a_l) = a_l \cdot f_l(x_{l-1}; w_l))$. Each attention value is initialized to 0.5 at the beginning. During training, both weights and attentions are updated. With pruning applied, the optimization objective of an attention-based neural network with compressed weights $\hat{w}$ is:

$$\min_{\hat{w}, A} \sum_i^N \Omega(f(x_i; \hat{w}, A)) + \gamma \Psi(A) + \lambda \sum_j \hat{w}_j^2$$

(1)
where $\mathcal{L}(\cdot)$ denotes the cross-entropy loss, $\Psi(\cdot)$ is the sparsity regularizer for structure learning (discussed in the next section), L2 regularizer encourages all weights to be small, and $\gamma$ and $\lambda$ are the coefficients for the sparsity regularizer and L2 regularizer, respectively.

### B. Pruning Ratios and Sparsity Regularizer

Layer-wise pruning methods typically require a pre-trained model as the starting point to search for a pruned structure. Recent work in [13] and [9] use naive Bayesian to optimize the pruning ratio layer-wisely in a given network. However, employing this kind of optimization in every training iteration is prohibitively time-consuming. In order to facilitate simultaneous optimization of structure and weights in each iteration, we propose a more efficient algorithm to calculate the pruning ratios directly from the attentions in our attention-based model.

In [5], [22], it was shown that scaling the network weights will suppress the unimportant ones, resulting in a pruning effect. As a result, the learned attentions can be used to represent the importance of each layer. That is, if a layer has a larger attention, it is considered to be more important so that it needs to be pruned less, and vice versa. Here, we introduce a positive hyper-parameter, the pruning factor $\alpha$, to gain more control when computing the pruning ratio $p_l$ based on the given attention $a_l$ of the $l^{th}$ layer:

$$p_l(a_l) = (1 - a_l)^\alpha, \quad (2)$$

where $a_l$ is the attention for the $l^{th}$ layer and $\alpha$ is the pruning factor. To ensure not all weights are pruned, we limited the maximum pruning ratio to be 99%. Since our attentions are applied along the layer dimension when weights are pruned, the overall sparsity $S$ of our attention-based model is computed as:

$$S(A) = \frac{\sum_l (1 - p_l(a_l)) n_{w_l}}{\sum_l n_{w_l}}, \quad (3)$$

where $p_l$ is the pruning ratio and $n_{w_l}$ is the total number of unpruned weights in the $l^{th}$ layer. Note that $n_{w_l}$ is a constant as it is pre-determined for each layer.

Regularizers like L1 or L2 encourage the network weights $w$ to be small, but not necessarily zeros. Thus, an additional regularizer is required in ASWL to encourage the pruning procedure to remove unimportant weights in each layer. Moreover, the L1/L2 regularizer encourages attentions to be zeroes without giving any consideration on the sparsity of the entire model, especially when the pruning factor is not set to 1. As a result, we adopted the sparsity regularizer proposed in [22] as follows and combined it with L2 in ASWL:

$$\Psi(A) = S(A)^2 \quad (4)$$

The square sparsity regularizer is differentiable and will help minimize the layer-wise sparsity during optimization.

### C. Simultaneous Sparse Structure Learning and Weight Optimization

Traditional pruning methods learn the pruned structure first, and then optimize the weights based on the pruned structure. The pruned structure can be easily found if a pre-trained model is available. However, with the weights training from random initialized values, efficient weights might change gradually from shallow to deep layers [3]. Meanwhile, a dense neural network contains a subnetwork that has the same accuracy even without fine-tuning [2]. These results motivated us to perform simultaneous learning on both the sparse structure and weights in ASWL by tracking both the dense network (through the backward propagation) and the sparse subnetwork (through the forward pass).

In ASWL training, both weights for the dense network and the sparse network are tracked so that the structure is simultaneously learned as the weights change. The parameters of the sparse network are pruned by removing at least the bottom $p_l$ percent dense weights in the $i_{th}$ layer based on the absolute weight values $|w_{i}|$. After training, the attention values are applied to the weights and discarded together with the weights in the dense network. Only the weights in the sparse network are stored.

Each training iteration in ASWL contains four steps: (1) the network classification loss is calculated through a forward pass based on the compressed weights $\hat{w}$ and attentions $A$, (2) both uncompressed weights $w$ and attentions $A$ are updated through back-propagation, (3) the pruning ratio $p$ is computed by the new attentions $A$ and the pruning factor $\alpha$, and (4) the model is compressed layer-wise using the pruning ratio $p$, and the compressed weights $\hat{w}$ are updated. The detailed training procedure is summarized in Algorithm 1. In ASWL, through simultaneous optimization, weights that have been pruned at first may be recovered later, and the weights defined as important at first can be pruned, all depending on the evolution of the network structure.

#### D. Convergence Analysis

Both uncompressed weights $w$ and attentions $A$ are updated in ASWL through back-propagation, and their convergence is...
guaranteed. Next, we shown that the convergence of ASWL by considering an attention model $f_A(A, w)$.

**Lemma 1.** If no normalization layer is applied after layer $i$, the attention scalar $a_i$ will be accumulated to the final output of the attention model, otherwise, the scalar will not affect the final results due to normalization. That is, the following equation holds for our attention model:

$$f_A(A, w) = \begin{cases} (\prod A) \cdot f(w) & \text{if } f_A \text{ doesn’t contain } L_{norm} \\ f(w) & \text{otherwise} \end{cases}$$

where $L_{norm}$ denotes a normalization layer.

We assumes that the training objective of the weights is smooth, that is: existing a constant $C > 0$ so that $\|\nabla f(w) - \nabla f(v)\| \leq C\|w-v\|$ for $v, w \in \mathbb{R}$. We also assumes that the stochastic gradients of the pruned weights $g(\hat{w}_t)$ are bounded, i.e., $\mathbb{E}[g(\hat{w}_t)]^2 \leq G^2$. Based on the above assumptions and the lemma, we have the following theorem for a non-convex function $f$ like deep neural networks:

**Theorem 1.** If function $f$ is non-convex, let the learning rate be $c = \frac{1}{\sqrt{T}}$, where $c = \sqrt{\frac{T(\mathbb{E}(f_0) - 0.01|A|f(w^*))}{C^2T^2}}$, and $T$ is the number of training iterations, for a pruned attention model $\hat{u}$ chosen from the ASWL iterations $\{\hat{a}_0, \hat{w}_0\}, ..., \{a_T, \hat{w}_T\}$, where the attention scalar $a_i \in A$ and takes a value in $[0.01, 1]$, the following holds:

$$\mathbb{E}[\nabla f(\hat{u})]^2 = O(\sqrt{C(f(w_0) - 0.01|A|f(w^*))}G$$

$$+ C^2\mathbb{E}[(\delta_t ||w_t||^2)])$$

where $\delta_t = ||w_t - \hat{w}_t||^2/||w_t||^2$ is the quality of the pruning, and $w^*$ is the global optimal of the model weights.

The above theorem can be proved with Lemma 1 following the same procedure as in [12]. Due to the space limit, the proofs of both lemma and theorem are omitted.

### III. Experimental Results

In this section, we perform extensive experiments with ASWL on VGG-16 [19], ResNet50 [8], and MobileNetV2 [17]. For VGG-16 and ResNet50, we simply replace the traditional convolutional layer and dense layer with our attention-based convolutional layer and dense layer. For MobileNetV2, we replace the 1 × 1 point-wise convolutional layer with an attention-based convolutional layer but left the depth-wise layer uncompressed since 99% of the parameters and calculations are contained in point-wise convolutional layers. There are totally three hyperparameters in ASWL: the coefficients of sparsity and L2 regularizer ($\gamma$ and $\lambda$), and the pruning factor $\alpha$. They are specified later in different experiments.

Our ASWL training pipeline was implemented in TensorFlow [1]. All models are trained on a computer with Intel i7 8700K CPU, 16GB RAM, and two NVIDIA RTX 2080 Ti graphic cards, each of which has 11GB of GDDR SDRAM. The source code of this work with detailed comments is available on GitHub

1https://github.com/kisonho/aswl.git

Specifically, we conducted our experiments on the standard MNIST [11], Cifar-10 [10] and ImageNet (ILSVRC-2012 in version 2.0.1) [16] datasets and compared with the following unstructured pruning methods: Learning both Weights and Connections for Efficient Neural Networks (ENN) [7], Rethinking Network Pruning (RNP) [14], Convolutional neural network pruning with WoodFisher (WF) [20], Sparse Networks from Scratch (SNS) [3], and Dynamic Pruning with Feedback (DPF) [12]. We also compared ASWL with following structured pruning methods: Discrimination-aware Channel Pruning (DCP) [25], Pruning from Scratch (PFS) [22], Pruning Filter in Filter (PFF) [15], and Structural pruning via latency-saliency knapsack (HALP) [18]. Note that structured and unstructured pruning methods are fundamentally different: structured pruning methods often involve a time-consuming traditional three-step pipeline (pre-training, pruning and fine-tuning) while unstructured pruning methods generally yield higher accuracy with reduced weight redundancy. Their results are thus not directly comparable. The structured pruning results are listed for reference only.
TABLE I: ASWL training results with different pruning factors on VGG-16 (top row), MobileNetV2 (middle row), and ResNet50 (bottom row) for MNIST.

| Model     | Pruning Factor | Baseline Acc. | ∆ Accuracy | Pruning Ratio |
|-----------|---------------|---------------|------------|--------------|
| VGG16-1   | 1             | 99.51%        | +0.04%     | 90.69%       |
| VGG16-1.5 | 1.5           | 99.47%        | +0.09%     | 88.56%       |
| VGG16-2   | 2             | 99.3%         | +0.03%     | 87.19%       |
| MobileNetV2-1 | 1 | 99.56% | +0.00% | 90.10% |
| MobileNetV2-1.5 | 1.5 | 99.56% | -0.07% | 87.32% |
| MobileNetV2-2 | 2 | 99.49% | -0.09% | 85.03% |
| ResNet56-1 | 1             | 99.50%        | +0.00%     | 83.70%       |
| ResNet56-1.5 | 1.5 | 99.50% | +0.00% | 83.70% |
| ResNet56-2 | 2             | 99.50%        | +0.00%     | 83.70%       |

TABLE II: ASWL training results on Cifar10, and the comparison with state-of-the-art network pruning methods. In the “from scratch” column, ✓ indicates static pruned structure during training, and × indicates simultaneous structure and weight training. In the “from scratch” column, ✓ indicates pruning from pre-trained, and ✓ indicates pruning from scratch. In the “unstructured pruning” column, × indicates structured pruning, and ✓ indicates unstructured. For each method, accuracy, and pruning ratio are reported, and the best results are highlighted in bold. The same applies to Table III.

| Model     | Method | Simul. Training | From Scratch | Unstructured Pruning | Baseline Acc. | ∆ Acc. | Pr. Ratio |
|-----------|--------|-----------------|--------------|----------------------|---------------|--------|----------|
| VGG16     | DCP [25] | × | × | × | 93.80% | -0.11% | 92.80% |
|           | PFS [22] | × | ✓ | × | 93.44% | +0.19% | 93.60% |
|           | PFF [15] | × | ✓ | ✓ | 93.25% | -0.40% | 92.80% |
|           | RNP [14] | × | × | ✓ | 93.69% | -0.04% | 90.10% |
|           | SNS [3] | ✓ | ✓ | ✓ | 92.64% | +0.76% | 92.00% |
|           | ASWL (Ours) | ✓ | ✓ | ✓ | 93.44% | +0.28% | 96.00% |
| ResNet56  | DCP [25] | × | × | × | 93.80% | -0.31% | 92.80% |
|           | PFS [22] | × | ✓ | × | 93.23% | -0.18% | 93.10% |
|           | PFF [15] | × | ✓ | ✓ | 93.10% | +0.12% | 77.70% |
|           | RNP [14] | × | × | ✓ | 93.80% | -0.31% | 93.10% |
|           | DPF-90 [12] | ✓ | ✓ | ✓ | 94.51% | -0.56% | 90.00% |
|           | DPF-95 [12] | ✓ | ✓ | ✓ | 94.51% | -1.26% | 95.00% |
|           | ASWL (Ours) | ✓ | ✓ | ✓ | 93.44% | +0.28% | 96.00% |

A. Results on MNIST

We first trained selected models with ASWL on MNIST, which contains 10 different handwriting digits with 60,000 training images and 10,000 testing images. Each of the models was trained with various pruning factors of 1, 1.5, and 2 for 100 epochs by the Adam optimizer at a learning rate of 0.001 and 0.98 decay for each epoch. The attentions of each layer are initialized at 0.5. The hyper-parameter γ (the sparsity regularizer coefficient) is used to help us achieve a desired pruning ratio and was set at 0.5 for all models. Following [19], [8], and [17], the other hyper-parameter λ (the L2 regularizer coefficient) is set at 5·10^{-4}, 0.0001, and 0.00004 for VGG16, ResNet56, MobileNetV2, respectively. For VGG16, we used an initial learning rate of 0.1 and multiplied 0.5 for every 20 epochs with the SGD optimizer (momentum 0.9), and trained for 250 epochs at a batch size of 128. For ResNet56, we followed the same settings in [8]. Again, the hyper-parameter γ (sparsity regularizer coefficient) is used to help us achieve a desired pruning ratio and was set at 5·10^{-4}, 0.0001, and 0.00004 for VGG16, ResNet56, MobileNetV2, respectively. The models with the best results are selected.

Table I shows the ASWL training results on MNIST with VGG16 (Top), MobileNetV2 (Middle), and ResNet56 (Bottom). In most situations, the pruning ratio progressively increases when we reduce the pruning factor. There is no obvious relation found between model accuracy and pruning factors, while a smaller pruning factor offers a greater pruning ratio. With about 10% to 25% of weights in the original dense models, our ASWL provides similar or many times higher accuracy.

Fig. 3 (a) shows the pruning details of each layer in VGG-16 with a pruning factor of α = 1.5. Deeper attention-based convolutional layers are pruned more than the shallower ones, while the last two attention-based dense layers have pruning ratios much less than the convolutional layers. The last layer is an attention-based dense layer.

B. Results on Cifar10

Cifar10 is a dataset that contains 10 different classes with 50,000 training images and 10,000 testing images. We trained VGG16 and ResNet56 with pruning factor of 1.5 and 1, respectively. For VGG16, we used an initial learning rate of 0.1 and multiplied 0.5 for every 20 epochs with the SGD optimizer (momentum 0.9), and trained for 250 epochs at a batch size of 128. For ResNet56, we followed the same settings in [8]. Again, the hyper-parameter γ (sparsity regularizer coefficient) is used to help us achieve a desired pruning ratio and was set at 5·10^{-4}, 0.0001, and 0.00004 for VGG16, ResNet56, MobileNetV2, respectively. The layer-wise pruning details of ResNet56 are shown in Fig. 3 (c), where the last layer is an attention-based dense layer.
TABLE III: ASWL training result for ResNet-50 on ImageNet, and the comparison with state-of-the-art network pruning methods. The baseline accuracy of the uncompressed ResNet-50 is 76.1% [8]. ST. indicates simultaneous training, FS. indicates from scratch, and UP. indicates unstructured pruning. Note that HALP [18] used reduction of FLOPS as the guidance of pruning and did not report the final pruning ratio.

| Method     | ST. | FS. | UP. | TI Acc. % | Pr. Ratio % |
|------------|-----|-----|-----|-----------|-------------|
| CP 0.5x [25] | ✓   | ×   | ×   | 75.4%     | 48.5%       |
| HALP [18]   | ×   | ×   | ×   | 74.3%     | 70% (FLOPS) |
| PFS 0.75x [22] | ✓   | ✓   | ×   | 75.6%     | 63.9%       |
| ENN [7]     | ✓   | ✓   | ✓   | 76.1%     | 60.0%       |
| WF-80 [20]  | ✓   | ✓   | ✓   | 76.8%     | 80.0%       |
| WF-90 [20]  | ✓   | ✓   | ✓   | 75.2%     | 90.0%       |
| SNS-20 [3]  | ✓   | ✓   | ✓   | 73.8%     | 80.0%       |
| DPF-80 [12] | ✓   | ✓   | ✓   | 75.5%     | 73.5%       |
| DPF-90 [12] | ✓   | ✓   | ✓   | 74.6%     | 82.6%       |
| ASWL (Ours) | ✓   | ✓   | ✓   | 76.5%     | 86.1%       |

TABLE III: ASWL training result for ResNet-50 on ImageNet, and the comparison with state-of-the-art network pruning methods. The baseline accuracy of the uncompressed ResNet-50 is 76.1% [8]. ST. indicates simultaneous training, FS. indicates from scratch, and UP. indicates unstructured pruning. Note that HALP [18] used reduction of FLOPS as the guidance of pruning and did not report the final pruning ratio.

2.5 and 5 for VGG16 and ResNet 56, respectively. Following [19] and [8], the other hyper-parameter λ is set at 5 · 10^{-4} and 0.0001 for VGG16 and ResNet 56, respectively. The models with the best results are selected.

Table II compares ASWL with state-of-the-art pruning methods on Cifar10 with VGG16 (top) and ResNet56 (bottom). Clearly, ASWL achieved a higher accuracy than the baseline on both models. Overall, our ASWL model achieved an outstanding pruning ratio with the highest increase of accuracy when compared to the baseline. These results clearly demonstrate the advantages of simultaneous training and layer-wise attention-based pruning in ASWL.

C. Results on ImageNet

Compared with MNIST and Cifar10, ImageNet is a much larger dataset that contains 1000 classes with 1.2M training images and 50K testing images. We trained ResNet-50 using ASWL following the same training setting in [8] for ResNet-50. The hyper-parameter γ was set at 0.5. Additionally, the pruning factor was set to 1. The models with the best results are selected.

Table III compares ASWL with state-of-the-art pruning methods on ResNet-50. Through simultaneous training with layer-wise pruning from randomly initialized weights, ASWL achieves the top-1 accuracy of 76.5% with a great pruning ratio (86.1%). Note that this accuracy is higher than the uncompressed ResNet-50 baseline model (76.1%). Considering the balance between the top-1 accuracy and the pruning ratio, ASWL provides a superior pruning result when compared to all existing pruning methods. If we consider accuracy alone, it is the second best.

IV. CONCLUSION

In this paper, we proposed a novel pruning pipeline, Attention-based Simultaneous sparse structure and weight Learning (ASWL). In ASWL, we first use the attention mechanism to learn the importance of each layer in a network and determine the corresponding pruning ratio. Then, the layer-wise sparsity and weights are jointly learned from scratch in a unified training procedure. Our extensive experiments on benchmark datasets show that ASWL achieves outstanding pruning results in terms of accuracy, pruning ratio, and operating efficiency.

REFERENCES

[1] M. Abadi, A. Agarwal et al., “Tensorflow: Large-scale machine learning on heterogeneous distributed systems,” arXiv preprint arXiv:1603.04467, 2016.
[2] A. da Cunha, E. Natale, and L. Viennot, “Proving the lottery ticket hypothesis for convolutional neural networks,” in ICLR, 2022. [Online]. Available: https://openreview.net/forum?id=VYkJr9-619-4520.
[3] T. Dettmers and L. Zettlemoyer, “Sparse networks from scratch: Faster training without losing performance,” 2020.
[4] X. Dong, S. Chen, and S. Pan, “Learning to prune deep neural networks via layer-wise optimal brain surgeon,” in NIPS, 2017, Conference Proceedings, pp. 4857–4867.
[5] G. B. Hacene, C. Lassance et al., “Attention based pruning for shift networks,” in ICPR. IEEE, 2021, pp. 4054–4061.
[6] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” arXiv preprint arXiv:1510.00149, 2015.
[7] S. Han, J. Pool et al., “Learning both weights and connections for efficient neural network,” in NIPS, 2015, Conference Proceedings, pp. 1135–1143.
[8] K. He, X. Zhang et al., “Deep residual learning for image recognition,” in CVPR, 2016, Conference Proceedings, pp. 770–778.
[9] Y. He, J. Lin et al., “Amc: Automi for model compression and acceleration on mobile devices,” in ECCV, 2018, pp. 784–800.
[10] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” 2009.
[11] Y. LeCun, C. Cortes, and C. J. Burges, “The mnist database of handwritten digits, 1998,” URL http://yann. lecun. com/exdb/mnist, vol. 10, p. 34, 1998.
[12] T. Lin, S. U. Stich et al., “Dynamic model pruning with feedback,” in ICLR, 2020.
[13] K. Liu, B. Dolan-Gavitt et al., “Fine-pruning: Defending against backdooring attacks on deep neural networks,” in Research in Attacks, Intrusions, and Defenses, M. Bailey, T. Holz et al., Eds. Cham: Springer International Publishing, 2018, pp. 273–294.
[14] Z. Liu, M. Sun et al., “Rethinking the value of network pruning,” in ICLR, 2019.
[15] F. Meng, H. Cheng et al., “Pruning filter in filter,” Advances in Neural Information Processing Systems, vol. 33, pp. 17629–17640, 2020.
[16] O. Russakovsky, J. Deng et al., “Imagenet large scale visual recognition challenge,” International journal of computer vision, vol. 115, no. 3, pp. 211–252, 2015.
[17] M. Sandler, A. Howard et al., “Mobilenetv2: Inverted residuals and linear bottlenecks,” in CVPR, 2018, Conference Proceedings, pp. 4510–4520.
[18] M. Shen, H. Yin, P. Molchanov et al., “Structural pruning via latency-saliency knapsack,” NIPS, vol. 35, pp. 12894–12908, 2022.
[19] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” Computational and Biological Learning Society, 2015, pp. 1–14.
[20] S. P. Singh and D. Alistarh, “Woodisher: Efficient second-order approximation for neural network compression,” arXiv e-prints, pp. arXiv–2004, 2020.
[21] F. Tung and G. Mori, “Clip-q: Deep network compression learning by in-parallel pruning-quantization,” in CVPR, 2018, Conference Proceedings, pp. 7873–7882.
[22] Y. Wang, X. Zhang et al., “Pruning from scratch,” in AAAI, 2020, pp. 12273–12280.
[23] P. Yin, J. Lyu et al., “Understanding straight-through estimator in training activation quantized neural nets,” in ICLR, 2019.
[24] M. Zhu and S. Gupta, “To prune, or not to prune: Exploring the efficacy of pruning for model compression,” in ICLR. OpenReview.net, 2018.
[25] Z. Zhuang, M. Tan et al., “Discrimination-aware channel pruning for deep neural networks,” in NIPS, 2018, pp. 875–886.