2020 7th International Conference on Information Science and Control Engineering (ICISCE) | 978-1-7281-6406-9/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICISCE50968.2020.00218

Recommender for Channel Pruning with Weakly Supervised Meta Learning

Wenfeng Yin*, Gang Dong, Yaqian Zhao, Rengang Li
State Key Laboratory of High-end Server & Storage Technology,
Inspur Electronic Information Industry Co., Ltd.,
Jinan 250101, China
Email: yinwenfeng@inspur.com

***Abstract*—The switchable neural network makes it feasible to deploy a deep neural network to a specific hardware platform by adjusting architectures but without retraining parameters. However, searching out the suitable architecture to adjust is time consuming and computation-tense. To accelerate searching architectures, this paper designs a recommender which takes one round to search and evaluate without repeating. In training stages, the recommender is alternately trained with a meta network which offers the recommender gradients as weakly supervision. In searching stages, the trained recommender extracts features from a specified layer's outputs, and generates suggested channels to prune for every layer. The recommender is deployed and evaluated on three common classification neural networks, including ResNet, MobileNet-v1 and MobileNet-v2. Experimental results validate that the proposed recommender is able to obtain compressed architectures whose accuracies surpass the switchable neural network's accuracy at a speed faster than evolutionary algorithm based searching methods.**

***Keywords*-pruning; meta learning; weakly supervised learning**

I. Introduction

Since parameters of neural networks are redundant, deep neural networks are supposed to shrink to the size suitable for their deploying in a specific hardware platform. Although neural network compression offers solutions to effectively reduce neural network parameters with less accuracy losses, its training costs time and a large number of computations. Each time a deep neural network is compressed for a specified runtime platform, the compressed neural network has to be retrained to recover its accuracy. To decrease computational consumptions in such compressing means, there has been proposed a kind of methods to avoid retraining by training a switchable neural network. Switchable neural networks, such as the slimmable neural network [1] [2] and the Once for All (OFA) [3], enable flexible and multiple compressions with training once. However, it still cost time on selecting a high-performance architecture for deploying switchable neural networks. Reducing the time delay on searching and evaluating architectures facilitates scenarios that dynamically compress neural networks as runtime circumstances vary.

Existing one-shot neural network architecture search (NAS) methods separate searching from training to accelerate architecture evaluations, achieved by sharing weights between selected architectures and the complete model. In such means, it is avoided to train the selected architectures from scratch. Once an one-shot model is trained, one-shot NAS methods apply evolutionary algorithm based search methods [4] or random search methods [5] to achieve multiple evaluations. Although one-shot NAS methods get rid of retraining selected architectures, the architecture evaluation procedure still recurrently executes inferring and evaluating until the number of iterations reaches the predefined limit. This paper attempts to design an architecture recommender to predict suitable architectures by model inferring once on one batch of data, so as to further simplify such iterative evaluation procedures as above.

In addition to sharing complete models' weights with selected architectures, meta networks offer another solution for avoiding retraining selected architectures to accelerate architecture evaluations [6]. Specifically, meta networks are employed for generating weights for selected architectures [7]. Meta networks have been investigated for applications in neural network compressions. For example, the metapruning method [8] assigns a meta network for each of convolutional layers, to separately generate weights. For a given compressed neural network, its architecture evaluation is achieved by inferring with weights predicted by meta networks. These meta networks are trained by gradients obtained from their connected convolutional layers which have access to ground truth labels. Such supervised training indicates that it is feasible to finetune a hypernetwork with assists of a cascade meta network enabling weakly supervised meta learning.

In order to accelerate searching architectures for flexible deployments, this paper designs a recommender which predicts suitable architectures with neural network inferring once on one batch of data. Since the ground truth labels of high-performance architectures on a specific runtime platform are lacked, this paper trains the recommender in weakly supervised meta learning by taking gradients from a cascade meta neural network. The recommender and the assistant meta neural network are alternately trained in training stages. During searching stages, the recommender extracts features from specified layers of a neural network and generates architectures whose accuracy and FLOPs satisfy requirements of the runtime platform. Evaluations of

978-1-7281-6406-9/20/$31.00 ©2020 IEEE
DOI 10.1109/ICISCE50968.2020.00218

the recommender are carried out on three common neural networks, including ResNet, MobileNet-v1 and MobileNet-v2. Results show the recommender accelerates searching architectures and meanwhile preserves the accuracy.

II. Methodology

To achieve weakly supervised meta learning in situation that labels of high-performance model architectures are lack, the assistant meta network and the recommender are designed with different tasks. The recommender is to predict channel numbers for first i layers based on current channel features derived from the layer i. The task of meta network is to produce updating weights for layer i and further offer gradients for recommender's training. In order to receive gradients from the meta network, the recommender is cascade to the meta network. By such design, the recommender is trained with unlabeled data of channel numbers combinations, and makes no effort to produce pseudo labels for unlabeled data in contrast to what existing unsupervised meta learning methods do. For example, one unsupervised method [9] predicts pseudo labels for unlabeled data by clustering and then apply such supervised meta learning methods as the model-agnostic meta-learning to execute learning tasks. Even in unsupervised learning scenarios that meta networks are employed to predict errors and weight updates as in [10], meta networks still learn rules with labeled data.

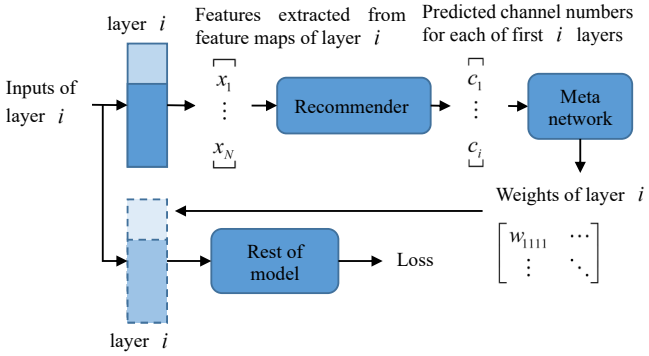

Figure 1. Training Flow of the recommender.

A. Training flow and Predicting flow

The training flow of our designed recommender is illustrated in Figure 1. Assuming that the ith layer is selected as the input provider of recommender, the recommender generates channel numbers for first i layers in a specific neural network. It is assumed that the channel number combination of first $i-1$ layers has influences in the ith layer's feature distributions in different channels. In other words, feature maps of the ith layer contain information indicating how to adjust the channel number combination of first i layers. The recommender extracts features related to channels from output feature maps of layer i, denoted as $[x_1, \cdots, x_i]$. Under different training limitations, the recommender is finetuned for producing a prediction of first i layers' channel numbers, marked as $[c_1, \cdots, c_i]$. Contrarily, existing pruning methods compress layers by layers. One research [11] uses a reinforcement learning agent to predict channel compression ratios in a layer-by-layer manner. NetAdapt [12] selects K different layers to prune and finetune per iteration, generating K subnetworks each of which only compresses one layer, and then chooses one subnetwork with best performance as the input to next iteration until the limitations such as FLOPs restrictions are reached. Even though NetAdapt compresses layers separately, it concerns relations between layers during pruning filters in a layer. Thus the proposed recommender is trained to meanwhile predict suitable channel numbers for first i layers, preserving relations between layers.

The assistant meta network receives the predicted channel number combination as its input, and generates new weights for layer i. A substitute layer of layer i is constructed with these generated weights to obtain updated output feature maps of layer i. By such way, it is not necessary to build a compressed neural network according to predicted channel numbers so as to evaluate the performance of a channel number combination. It is feasible to perform evaluations on output feature maps of the substitute layer of layer i. Moreover, the assistant meta network is able to derive gradients from the rest of model by the substitute layer i inferring with its original inputs, and then pass gradients to the recommender enabling weakly supervised learning.

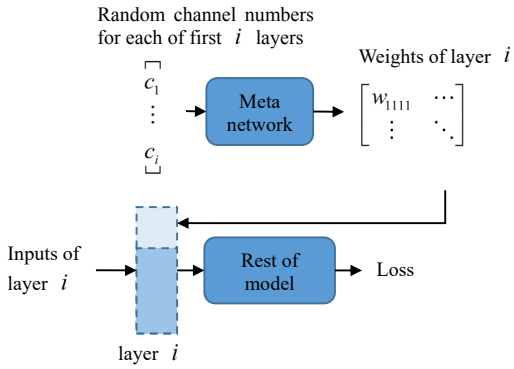

Figure 2. Training Flow of the meta network.

As to the ith layer's location in a specific neural network, the middle convolutional layer of a residual block [13], the first convolutional layer of a depth-wise separable convolution block [14], the middle convolutional layer of an inverted residual with linear bottleneck block [15] or single convolutional layer are selectable. In the case of choosing the middle convolutional layer of a residual block as the input provider of recommender, it is supposed to build a complete substitute block for the residual block rather than merely a substitute layer for the middle layer. However, the meta

network only produce weights for the middle layer of the substitute residual block. In addition, the loss optimization is computed on outputs of the substitute residual block rather than outputs of the middle layer i.

In the meta network's training procedures, inputs of meta network are randomly generated subject to training limitations such as FLOPs restrictions. As shown in Figure 2, same with the training stage of recommender, the meta network produce weights to update the substitute layer i and further connect to the rest of a neural network to obtain gradients.

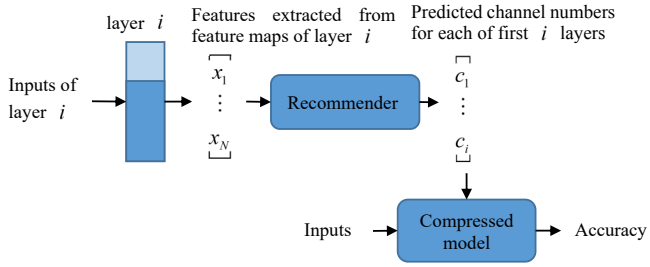

Figure 3. Search Flow of the recommender.

After the recommender and the meta network finish their alternate training, the recommender independently perform prediction tasks during a neural network's inferring stage. Different from the training procedure of recommender, a compressed model is built up based on the predicted channel numbers, so as to obtain reliable accuracies as evaluation standards. The search flow of recommender is depicted in Figure 3.

B. Structure of recommender and assistant network

The structure of our designed recommender consists of one switchable fully connected layers, one switchable batchnorm layer and one normal fully connected layers. The switchable fully connected layer, as the input layer, shares a group of weights among any one compressed layer with neuron numbers smaller than the complete layer. The recommender benefits in the switchable fully connected layer's ability of adjusting neuron numbers, to receive input features as the layer i changes its output channel numbers. The switchable batchnorm layer contains a list of normal batchnorm layer with different neuron numbers, adapting to the change of output neuron numbers of the switchable fully connected layer. The final layer of recommender is a normal fully connected layer, whose neuron numbers equal to i, i.e. the number of layers or blocks to be compressed.

The architecture of assistant meta network is composed of one normal fully connected layer as the input layer and one switchable fully connected layer as the output layer. The output size of meta network is equivalent to weight numbers of the convolution kernel in layer i specified as the input provider of recommender.

C. Optimization with different limitations

During the training stage of recommender, three losses are optimized to assure the recommender learn to predict compressed channel numbers for first i layers of a neural network with different FLOPs limitations. As Figure 4 illustrates, in addition to the original loss of neural network, the loss of FLOPs restrictions and the loss of feature map reconstructions are considered. The optimization function is formulated as

$$\min Loss + \lambda Loss_{feature} + \mu Loss_{FLOPs}, \qquad (1)$$

where λ and μ are respectively scale factors for the loss of features $Loss_{feature}$ and the loss of FLOPs $Loss_{FLOPs}$. The loss of features measures the discrepancy between feature maps F of complete layer i and feature maps F^* of compressed layer i. To keep features discriminative, the loss of features is adopted in the optimization function with a scale factor λ. Reducing the loss of features is beneficial for preserving discriminative features in output feature maps of compressed layer i.

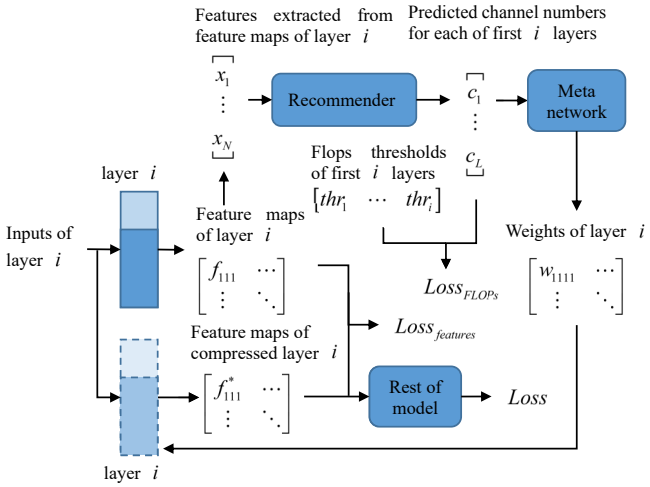

Figure 4. Loss optimization of the recommender and the meta network.

The loss of FLOPs restrictions are to achieve a FLOPs balance among all compressed layers, which offers different FLOPs threshold for each layer. The FLOPs thresholds compose an encoded vector, whose every element is set as the minor one between constant one and the ratio of one layer's FLOPs to the maximum FLOPs of all layers. Specifically, the encoded vector of FLOPs thresholds is formulated as

$$thres = [\widehat{fp}_{\max} \Big/ fp_1, \cdots, \widehat{fp}_{\max} \Big/ fp_i] \qquad (2)$$

where $\widehat{fp}_{\max}$ denotes the switchable value of maximum FLOPs of all layers and it is adjusted as $\widehat{fp}_{\max} = \alpha * \max(fp_1, \cdots, fp_i)$ for different training restrictions. fp_i is the FLOPs of layer i. α is the compression ratio of

whole neural network, which is treated as the total FLOPs compression ratio in this paper. Each element of this vector is upper bounded by one. There is no restriction settled to the sum of FLOPs thresholds in this vector, while the maximum FLOPs of all layers is faded according to the compression ratio of whole neural network. Different from directly assigning the compression ratio of whole model to every layers, this compression ratio is merely applied in the layer with maximum FLOPs.

III. Evaluations and Analyses

The performance of recommender is validated on three common neural networks' switchable models prepared by an existing switchable training solution [2]. Error rates of these switchable models are calculated as baselines estimating the accuracy decline of our recommender. The recommender and one-shot NAS methods [4] search architectures after obtaining a well-trained model, while they search in different means. Two contrastive schemes are designed with commonly used searching methods in NAS, i.e. evolutionary algorithm based and reinforcement learning based methods. Specifically, the evolutionary algorithm based scheme reproduces the searching method in [8]. And the reinforcement learning based scheme simplifies the structure of a recurrent neural network controller and keeps the computation rules for gradients and reward functions in [16]. Results of switchable models, evolutionary algorithm based methods, reinforcement learning based methods and the recommender are respectively marked with suffix US, EA, RL and meta. Evaluations are performed with different values of FLOPs thresholds corresponding to FLOPs compression ratios to the entire model.

Table I
Error rates on ResNet

Thresholds	ResNet-US	ResNet-EA	ResNet-RL	ResNet-meta
0.5	32.3%	36.6%	34.7%	34.3%
0.625	30.2%	30.7%	30.2%	29.7%
0.75	28.7%	29.1%	27.7%	28.5%
0.875	27.3%	28.5%	26.8%	27.4%
1.0	26.8%	27.4%	26.2%	26.7%

As shown in Table 1, on switchable ResNet models, the recommender achieves error rates smaller than those of evolutionary algorithm based methods. Some accuracies of recommender are slightly greater than those of switchable ResNet models under several FLOPs restrictions. The same situation shows appearances on MobileNetv1 and MobileNetv2, as listed in Table 2 and Table 3, where Netv1 and Netv2 are respectively abbreviations for MobileNetv1 and MobileNetv2. It indicates that the recommender is able to obtain compressed models with improved accuracies and small FLOPs. Reinforcement learning based methods achieve error rates smaller than those of the rest methods on large FLOPs thresholds.

Table II
Error rates on MobileNet-v1

Thresholds	Netv1-US	Netv1-EA	Netv1-RL	Netv1-meta
0.5	35.2%	39.2%	34.6%	35.4%
0.625	32.2%	34.1%	29.3%	32.1%
0.75	30.0%	31.2%	29.3%	30.3%
0.875	28.7%	29.7%	28.1%	29.6%
1.0	27.9%	28.4%	26.8%	27.9%

Table III
Error rates on MobileNet-v2

Thresholds	Netv2-US	Netv2-EA	Netv2-RL	Netv2-meta
0.5	34.7%	35.4%	35.2%	32.5%
0.625	31.2%	31.9%	32.4%	31.1%
0.75	30.1%	30.6%	30.1%	29.8%
0.875	28.3%	29.1%	28.0%	28.5%
1.0	27.8%	28.2%	27.8%	27.9%

In Table 4, results of FLOPs compressions and search delay compressions are summarized on condition that the FLOPs threshold is one. It is observed that evolutionary algorithm based methods achieve a greater compression ratio of FLOPs. The recommender preserves FLOPs and meanwhile satisfies the FLOPs compression restriction, to compensate accuracy losses. As to compressing search delays, experiments are executed on 4 GPUs, Tesla V100, with CUDA version 10.1. Moreover, the evolutionary algorithm based method merely searches for two rounds and its population size is 50. As shown in Figure 4, the recommender takes 494.84 seconds to search and evaluate compressed architectures for ResNet, while evolutionary algorithm based and reinforcement learning based methods respectively spend 1.64 hours and 1.46 hours. Correspondingly the recommender achieves accelerating 11 times and 10 times compared to above two methods. Although the reinforcement learning based method surpasses the recommender on error rates, it costs more search delays than the recommender.

Table IV
Compression on FLOPs and time dalay

	ResNet	MobileNetv1	MobileNetv2
FLOPs-EA(MB)	3157.7	450.6	251.1
FLOPS-RL(MB)	3454.6	471.2	257.5
FLOPS-meta(MB)	3835.7	502.8	268.3
Delay-EA(h)	1.64	1.16	2.32
Delay-RL(h)	1.46	0.73	4.58
Delay-meta(s)	494.84	246.22	393.58

Error rates of compressed neural networks searched by the recommender with 5 FLOPs thresholds are calculated among batches and shown in Figure 5. As FLOPs thresholds increase, error rates become stable in small ranges. When the FLOPs threshold is 0.5, error rates of recommender vary around the switchable Mobilenet-v2 models' error rate 34.7%. It shows that the recommender has searched out compressed architectures with better accuracies. And Figure 6 displays such an example of compressed architectures.

Figure 6 illustrates compression ratios of every inverted

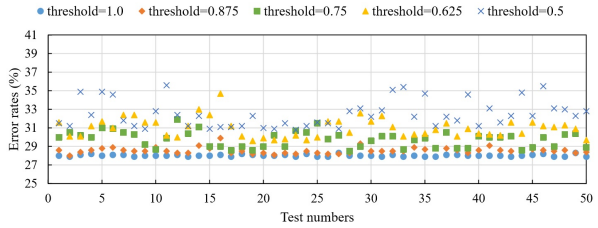

Figure 5. Error rates with 5 FLOPs thresholds among different batches on MobileNet-v2.

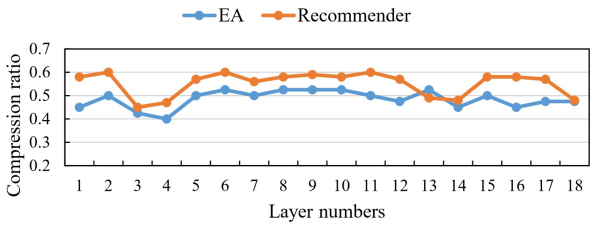

Figure 6. Compression ratios for each block in MobileNet-v2 when the FLOPs threshold is 0.5.

residual with linear bottleneck block on MobileNet-v2, respectively predicted by the recommender and evolutionary algorithm based methods, when the FLOPs threshold is 0.5. It's observed that compression ratios achieved by these two methods have similar changing trends on some layers. Since the designed FLOPs threshold in Equation 2 allows elastic FLOPs restrictions, the recommender is able to search out architectures, whose some layers beyond the FLOPs threshold 0.5 while whose total FLOPs satisfies the FLOPs threshold.

IV. Conclusion

This paper proposed a recommender to accelerate deploying the switchable neural network to different platforms. The designed recommender is a hypernetwork cascade with the other hypernetwork whose gradients is propagated to the recommender to facilitate weakly supervised learning. In other words, the recommender works as an architecture predictor while the other hypernetwork serves as an architecture evaluator. In training stages, the recommender and the evaluator hypernetwork are trained in turn. In searching stages, only the recommender is applied to generate suggestions for channel pruning. The evaluations on three common neural networks show the recommender obviously reduces the time delay of searching suitable architectures with a small accuracy decline.

References

[1] J. H. Yu, L. J. Yang, N. Xu, J. C. Yang, and T. Huang, *Slimmable Nural Networks*. Proceedings of 7th International Conference on Learning Representations, New Orleans, 2019.

[2] J. H. Yu and T. Huang, *Universally Slimmable Networks and Improved Training Techniques*. Proceedings of International Conference on Computer Vision, Seoul, Korea, 2019.

[3] H. Cai, C. Gan and S. Han, *Once for All: Train One Network and Specialize it for Efficient Deployment*. Proceedings of International Conference on Learning Representations, Ethiopia, 2020.

[4] Z. C. Guo, X. Y. Zhang, H. Y. Mu, W. Heng, Z. C. Liu, Y. C. Wei and J. Sun, *Single Path One-Shot Neural Architecture Search with Uniform Sampling*. Proceedings of 16th European Conference on Computer Vision, 2020.

[5] G. Bender, P. J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le. *Understanding and simplifying one-shot architecture search*. Proceedings of International Conference on Machine Learning, 2018: 549–558.

[6] T. Elsken, J. H. Metzen and F. Hutter, *Neural Architecture Search: A Survey*. Journal of Machine Learning Research, 2019, 20(55):1–21.

[7] A. Brock, T. Lim, J. M. Ritchie and N. Weston, *Smash: one-shot model architecture search through hypernetworks*. Proceedings of International Conference on Learning Representations, Vancouver, Canada, 2018.

[8] Z. C. Liu, H. Y. Mu, X. Y. Zhang, Z. C. Guo, X. Yang, T. K-T. Cheng and J. Sun, *MetaPruning: Meta Learning for Automatic Neural Network Channel Pruning*. Proceedings of International Conference on Computer Vision, Seoul, Korea, 2019.

[9] K. Hsu, S. Levin and C. Finn, *Unsupervised Learning via Meta-Learning*. Proceedings of 7th International Conference on Learning Representations, New Orleans, 2019.

[10] L. Metz, N. Maheswaranathan, B. Cheung and J. Sohl-Dickstein, *Learning Unsupervised Learning Rules*. Proceedings of 7th International Conference on Learning Representations, New Orleans, 2019.

[11] Y. H. He, J. Lin, Z. J. Liu, H. R. Wang, L. J. Li, and Sx. Han, *AMC: AutoML for Model Compression and Acceleration on Mobile Devices*. Proceedings of European Conference on Computer Vision, 2018.

[12] T. J. Yang, A. Howard, B. Chen, X. Zhang, A. Go, M. Sandler, V. Sze and H. Adam, *NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications*. Proceedings of European Conference on Computer Vision, 2018.

[13] K. M. He, X. Y. Zhang, S. Q. Ren and J. Sun, *Deep residual learning for image recognition*. Proceedings of IEEE conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 2016: 770–778.

[14] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. J. Wang, T. Weyand, M. Andreetto and H. Adam, *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Application*. arXiv preprint arXiv:1704.04861, 2017.

[15] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. C. Chen, *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. Proceedings of IEEE conference on Computer Vision and Pattern Recognition, 2018: 4510-4520.

[16] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le and J. Dean, *Efficient Neural Architecture Search via Parameter Sharing*. arXiv preprint arXiv:1802.03268v2, 2018.