Supplementary Materials for Structural Alignment for Network Pruning through Partial Regularization

Shangqian Gao[1], Zeyu Zhang[2], Yanfu Zhang[1], Feihu Huang[1], and Heng Huang[3]

[1]Electrical and Computer Engineering Department, University of Pittsburgh
[2]School of Information, University of Arizona
[3]Department of Computer Science, University of Maryland at College Park

A. Implementation Details

We list more details of the implementation in this section. To train the base model on CIFAR-10, we follow the standard training PyTorch training examples. The models are trained for 200 epochs, and the learning rates are 0.1, 0.01, and 0.001 for the first 100 epochs, 100 to 150 epochs, and 150 to 200 epochs. We select SGD as the optimizer with a momentum of 0.9 and weight decay of 10^{-4}. After pruning, we finetune the model for 160 epochs using similar optimization hyperparameters. The learning rates for fine-tuning are 0.1, 0.01, and 0.001 for the first 80 epochs, 80 to 120 epochs, and 120 to 160 epochs.

For ResNet models on ImageNet, we train the model for 90 epochs following the standard PyTorch ImageNet training script. The learning rates are 0.1, 0.01, and 0.001 for the first 30 epochs, 30 to 60 epochs, and 60 to 90 epochs. Still, SGD is selected as the optimizer with momentum of 0.9 and weight decay of 10^{-4}. For MobileNet-V2, we train the model for 150 epochs with the cos-annealing learning rate and a start learning rate of 0.045, and weight decay 4×10^{-5} as mentioned in their original paper [2]. For each model on ImageNet, we fine-tune them for 100 epochs. The learning rate schedule and the start learning rate are the same as the training of the based model. Except that when fine-tuning ResNets, we set the learning rate to 0.0001 for 90 to 100 epochs. We also list p and E_{start} in Tab. 1. As mentioned in the paper, we set E_{start} to around 20% of the total training time.

As we discussed in the paper, AGN is composed of dense layers and GRUs, and now we present the architecture of AGN in Tab. 2. In Tab. 2, $z \in \Re^{L\times 32}$ is the input to the AGN, and z is initially sampled from a normal distribution, and it is then fixed during training. Outputs o_l are continuous values. We use the following equation to covert it into $\mathbf{v}_l$:

$$\mathbf{v}_l = \text{round}(\text{sigmoid}((o_l + g + b)/\tau)), \qquad (1)$$

Architecture	Dataset	p	E_{start}
ResNet-56	CIFAR-10	0.48	40
MobileNet-V2		0.54	40
ResNet-34	ImageNet	0.54	18
ResNet-50		0.37	18
ResNet-101		0.41	18
MobileNet-V2		0.65	30

Table 1: Choices of p and E_{start} for different models.

where sigmoid$(\cdot)$ is the sigmoid function, round$(\cdot)$ is the rounding function, g is sampled from Gumbel distribution ($g \sim$ Gumbel$(0, 1)$), b is a constant value to make sure pruning starts from the whole model, and τ is the temperature hyper-parameter. As shown in Eq. 1, straight-through Gumbel-Sigmoid [1] are used to produce the final binary vector $\mathbf{v}$. The function of AGN can also be understood as translating z into the final architecture vector $\mathbf{v}$. For all experiments, we set $\tau = 0.4$ and $b = 3.0$.

Table 2: The architecture of AGN.

Input z
GRU(32,64)$\rightarrow$ LayerNorm$\rightarrow$ GeLU
Dense$_l$(64, C_l)$\rightarrow$Outputs o_l, $l = 1, \cdots, L$

Note that the additional training costs of our method are trivial compared to the original training process, mainly because we only train AGN in a small sub-dataset of the whole dataset. Take MobileNet-V2 as an example, its average model training time is 976 seconds per epoch, and the average AGN training time is 95 seconds per epoch. The overhead is **around 10%** of the original training time.

B. Regularization with Blocks

Recent CNN designs often use a block as a building block. In the paper, we do not explicitly talk about this setup to simplify notations. Usually, we group $\mathbf{v}_l$ based on the definition of blocks, and L is the total number of unique $\mathbf{v}_l$, and it could be different from the actual number of layers. To avoid conflicts, we rewrite $\mathbf{v}_l$ as $\mathbf{v}_k$, and $k = 1, \cdots, K$, and K is the total number of unique $\mathbf{v}_k$ (a single $\mathbf{v}_k$ can be used for multiple layers along different dimension). Let us use the basic block from ResNets as an example. With this setting, $\mathcal{R}_w$ can be written as:

$$R_w(\mathcal{W}) = \sum_{j=1}^{B} \sum_{i \in S_k} \frac{\hat{N}_k}{\hat{N}} (\|\mathcal{W}_j^{\text{upper}}{}_{[i,:,:,:]}\|_{\text{GL}} + \|\mathcal{W}_j^{\text{lower}}{}_{[:,i,:,:]}\|_{\text{GL}}), \tag{2}$$

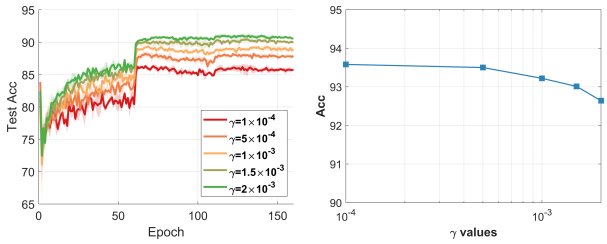
where k is the corresponding index of $\mathbf{v}_k$ for the jth block, and B is the total number of blocks. For the upper layer in the jth block, the regularization is applied on the output dimension, and the regularization is applied on the input dimension for the lower layer in the jth block. The formulation of Eq. 2 can be easily extended to other block types, such as the bottleneck block in ResNets and the inverted residual block in MobileNet-V2.

C. Different Choices of γ

In this section, we want to discuss the impact of γ and how it affects the performance of the whole model. We plot the related results in Fig. 1. From the figure, we can see that when we increase γ, we can get a better sub-network during the training process. However, it also negatively affects the full model performance, as shown in Fig. 1b. The full model accuracy provides the baseline before pruning. Δ-Acc often increases when we have a better sub-network, but if the baseline accuracy is too low, the final fine-tuned accuracy will also be worse. On CIFAR-10, this happens when we use $\gamma \geq 1 \times 10^{-3}$ (a better Δ-Acc but a worse final accuracy). On the ImageNet dataset, we also find that when $\gamma = 5 \times 10^{-4}$, we can get a similar baseline accuracy with better sub-network accuracy. We can probably use a larger γ to get better Δ Top-1 accuracy, but it will create weaker baselines, which is not a good practice for fair comparisons.

D. Stability of Sub-network Architectures

In our paper, we apply soft regularization when training model weights. One problem is that if the sub-network architecture changes frequently, then most weights will be penalized, which brings a trivial difference between penalizing all weights. To investigate this problem, we plot the hamming distance between the sub-network architecture of two consecutive epochs in Fig. 2. We can see that the hamming distance will be smaller than 0.05 after around 10 epochs, and the hamming distance will continuously decrease after 20 epochs. This observation suggests that the sub-network architecture becomes more and more stable after 20 epochs. Even if some weights are wrongly penalized at the beginning, they still have enough time to recover their magnitudes.



(a) (b)

Figure 1: (a) The sub-network performance during training with different γ. (b) Performance of the whole model given γ. All experiments are conducted on CIFAR-10 with ResNet-56.

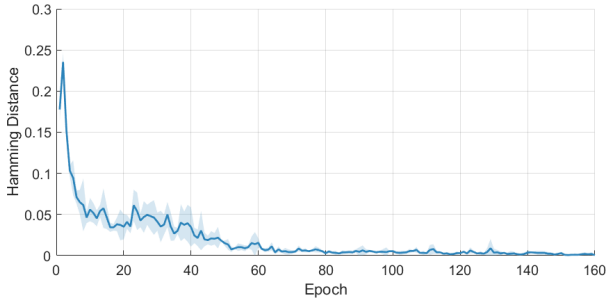


Figure 2: Hamming Distance between sub-network architecture during the training process. This experiment is conducted on CIFAR-10 with ResNet-56.

References

[1] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[2] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.