Supplementary Materials for Jointly Training and Pruning CNNs via Learnable Agent Guidance and Alignment

1. Bounding our Agent's Actions

As mentioned in Section 3.3 of our paper, we calculate the minimum ($a_{l,min}$) and maximum ($a_{l,max}$) feasible pruning rates for the l-th layer before pruning it to ensure that reaching the desired FLOPs budget, FLOPs$_{desire}$, is still possible after doing so. However, before formally introducing our scheme for calculating $a_{l,min}, a_{l,max}$, we present how we implement our pruning actions in practice.

1.1. Implementation of our Agent's Actions

We describe our implementation for our agent's actions for each architecture. For all models, we take each block of a CNN model as one 'layer' in our framework.

ResNets: for our experiments on ResNet [2] models (ResNet-56 on CIFAR-10 [4] and ResNet-18/34 on ImageNet [1]), we take each residual block as one layer. It contains a structure as `Conv1-BN-ReLU-Conv2-BN` where `Conv1` and `Conv2` are the convolution layers, `BN` represents Batch Normalization [3], and `ReLU` is the ReLU activation function. For each block, given the predicted action a_l for pruning it, we remove $\lfloor a_l \times c \rfloor$ output channels of the `Conv1` layer and the same number of input channels of the `Conv2` layer where c is the number of output/input channels of `Conv1`/`Conv2`.

MobileNet-V2: for experiments using MobileNet-V2, we take each inverted residual block [6] as one layer for pruning. Each block has the structure with `Conv1-BN-ReLU6-DW_Conv-BN-ReLU6-Conv2-BN` form where `DW_Conv` is a depth-wise convolution layer. Given the predicted action a_l, we remove $\lfloor a_l \times c \rfloor$ output channels of `Conv1` and the same amount of channels of `DW_Conv` and input channels of `Conv2`.

In summary, our pruning scheme changes the inner number of channels in each block of a CNN and preserves its number of input and output channels.

1.2. Calculating Action Bounds

We calculate $a_{l,min}, a_{l,max}$ for the l-th layer based on the total model's FLOPs that we denote with FLOPs$_T$ the number of FLOPs for the previous pruned layers FLOPs$_{1:l-1}$; the number of FLOPs for the next remaining layers FLOPs$_{l+1:L}$; FLOPs$[l]$; and FLOPs$_{desire}$. The formulations are as follows:

$$a_{l,min} = 1 - \frac{\text{FLOPs}_{desire} - \text{FLOPs}_{1:l-1}}{\text{FLOPs}[l]} \tag{1}$$

$$a_{l,max} = 1 - \frac{\text{FLOPs}_{desire} - \text{FLOPs}_{1:l-1} - \text{FLOPs}_{l+1:L}}{\text{FLOPs}[l]} \tag{2}$$

In these equations, $a_{l,max}$ prevents very high pruning rates that even if all the next layers are kept intact, reaching FLOPs$_{desire}$ get infeasible. Similarly, $a_{l,min}$ provides the minimum pruning rate for the current layer given all the next layers are pruned completely. We clip the predicted action a_l to lie in $[a_{l,min}, a_{l,max}]$ when pruning the l-th layer.

2. Experimental Settings

We provide more details of our experimental settings in the following.

CIFAR-10: For CIFAR-10 experiments, we evaluate our method on ResNet-56 [2] and MobileNet-V2 [6]. In our iterative pruning phase, we train both of the CNN models for 200 epochs with the batch size of 128 using SGD with momentum [7] of 0.9, weight decay of $1\text{e}{-4}$, and starting learning rate of 0.1. We decay the learning rate by 0.1 on epochs 100 and 150. We take 5000 samples of the training dataset as a subset for calculating the agent's reward. For all cases, we start to train the RL agent after 10 warmup epochs of the model's weights. Specifically, we collect initial data for the replay buffer of the RL agent from epochs 10 to 20. Then, for both ResNet-56 and MobileNet-V2, we update the agent from epoch 20 until the epoch 90, and we train only the model's weights from epoch 90 to 200. After the iterative stage, we prune the model's architecture and finetune it with the same settings for the base model.

ImageNet: We use ResNet-18, ResNet-34, and MobileNet-V2 for ImageNet experiments. For the iterative training stage of ResNets, we use SGD as the optimizer with the momentum of 0.9, weight decay of $1\text{e}{-4}$, and the start

learning rate of 0.1. We train ResNet models for 90 epochs, and we decay the learning rate to 0.01 and 0.001 at epochs 30 and 60. For MobileNet-V2, we do so for 155 epochs with a batch size of 256. We train the model's weights using SGD with the momentum of 0.9, weight decay of $1\mathrm{e}{-4}$, and starting learning rate of 0.05 decayed using cosine scheduling [5]. For all cases, we use 50000 samples of the training dataset to evaluate rewards of the agent. Similar to CIFAR-10 experiments, we train the model's weights for 10 warmup epochs followed by 10 epochs for filling the replay buffer of the RL agent. Then, for MobileNet-V2, we train the agent's policy from epochs 20 to 90, and we only train the model's weights from epoch 90 to 155. After the pruning stage, we fine-tune the pruned model with the same training parameters as the base model. For ResNet models, we train the agent's policy from epochs 20 to 70, and the model's weights are trained from epochs 70 to 90.

Ablation Experiments: we follow the same settings as mentioned above for our ablation experiments in Tab. 3 of the paper. For the visualizations in Fig. 2, we use the same settings except that we perform our iterative pruning scheme for 90 epochs.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[3] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015. 1

[4] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 1

[5] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. 2

[6] 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. 1

[7] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013. 1