Supplementary Material for "DHP: Differentiable Meta Pruning via HyperNetworks"

Yawei Li[1], Shuhang Gu[1,2], Kai Zhang[1], Luc Van Gool[1,3], Radu Timofte[1]

[1]Computer Vision Lab, ETH Zürich, [2]The University of Sydney, [3]KU Leuven
{yawei.li, kai.zhang, vangool, radu.timofte}@vision.ee.ethz.ch
shuhanggu@gmail.com

In this supplementary material, we first detail the training and testing protocol of the networks for different tasks in Sec. 1. Then the latent vector sharing strategy is shown in Sec. 2. More results are shown in Sec. 3.

1 Training and Testing Protocol

As explained in the main paper, the proposed DHP method does not rely on the pretrained model. Thus, all of the networks are trained and pruned from scratch. The hypernetworks are first randomly initialized. Proximal gradient is used to sparsify the latent vectors. When the difference between the target and the actual FLOPs compression ratio is below 2%, the pruning procedure stops. Then the pruned latent vectors as well as the pruned outputs of the hypernetworks are derived. After that, the outputs of hypernetworks are used as the weight parameters of the backbone network and updated by SGD or Adam algorithm directly. After the pruning procedure, the hypernetworks are removed. The training continues and the training protocol are the same as that utilized for training the original network. The number of pruning epochs is much smaller than that used for training the original network. The following of the section describes the training protocols of different tasks.

1.1 Image Classification

CIFAR10 CIFAR10 [6] contains 10 different classes. The training and testing subsets contain 50,000 and 10,000 images with resolution 32×32, respectively. As done by prior works [3, 4], we normalize all images using channel-wise mean and standard deviation of the the training set. Standard data augmentation is also applied. The networks are trained for 300 epochs with SGD optimizer and an initial learning rate of 0.1. The learning rate is decayed by 10 after 50% and 75% of the epochs. The momentum of SGD is 0.9. Weight decay factor is set to 0.0001. The batch size is 64.

Tiny-ImageNet For image classification, the pruning method is also compared on Tiny-Imagenet. It has 200 classes. Each class has 500 training images and

50 validation images. The resolution of the images is 64×64. The images are normalized with channel-wise mean and standard deviation. Horizontal flip is used to augment the dataset. The networks are trained for 220 epochs with SGD. The initial learning rate is 0.1. The learning rate is decayed by a factor of 10 at Epoch 200, Epoch 205, Epoch 210, and Epoch 215. The momentum of SGD is 0.9. Weight decay factor is set to 0.0001. The batch size is 64.

1.2 Super-Resolution

Training protocol For image super-resolution, the networks are trained on DIV2K [1] dataset. It contains 800 training images, 100 validation images, and 100 test images. Image patches are extracted from the training images. For EDSR, the patch size of the low-resolution input patch is 48×48 while for SRResNet the patch size is 24×24. The batch size is 16. The networks are optimized with Adam optimizer. The default hyper-parameter is used for Adam optimizer. The weight decay factor is 0.0001. The networks are trained for 300 epochs. The learning rate starts from 0.0001 and decays by 10 after 200 epochs. The networks are tested on Set5 [2], Set14 [8], B100 [7], Urban100 [5], and DIV2K validation set.

Simplified EDSR architecture In order to speed up the training of EDSR, a simplified version of EDSR is adopted. The original EDSR contains 32 residual blocks and each convolutional layer in the residual blocks has 256 channels. The simplified version has 8 residual blocks and each has two convolutional layers with 128 channels.

1.3 Denoising

For image denoising, the networks were trained on the gray version of DIV2K dataset and tested on BSD68 and DIV2K validation set. As done for image super-resolution, image patches are extracted from the training images. For DnCNN, the patch size of the input image is 64×64 and the batch size is 64. For UNet, the patch size is 128×128 and the batch size 16. Gaussian noise is added to degrade the input patches on the fly with noise level $\sigma = 70$. Adam optimizer is used to train the network. The weight decay factor is 0.0001. The networks are trained for 60 epochs and each epoch contains 10,000 iterations. So in total, the training continues for 600k iterations. The learning rate starts with 0.0001 and decays by 10 at Epoch 40.

2 Latent Vector Sharing Strategy for Different Networks

2.1 Basic criteria

To construct the hypernetworks, all of convolutional layers including standard convolution, depth-wise convolution, point-wise convolution, group convolution,

and transposed convolution are attached a latent vector. The latent vectors act as the handle for network pruning. Thus, by dealing with only the latent vector, we can control how the convolutional layers are pruned. But there could be complicated cases in the modern network architecture where the latent vectors have to be shared among different layers. Thus, during the development of the algorithm, we summarize some basic rules for latent vector sharing. In the following, we first describe the general rules for latent vectors and then detail the specific rules for special network blocks.

I Every convolutional layer is attached a latent vector.
II The channel that the latent vector controls and the dimension of the latent vector vary with the types of convolutional layers.
   (a) For standard convolution, point-wise convolution and transposed convolution, the latent vector controls the output channel of the layer and the dimension of the latent vector is the same as the number of output channels.
   (b) For depth-wise convolution and group convolution, the latent vector controls the input channels per group. The dimension of the latent vector is the same as the number of input channels per group. That is, the latent vector of depth-wise convolution contains only one element.
III The latent vectors are shared among consecutive layers. This is because the output and input channels of consecutive layers are correlated. Thus, the hypernetworks receive the latent vectors of the previous layer and the current layer as input.
IV Not every latent vector needs to be sparsified. The latent vectors free from sparsification are list as follows.
   (a) The latent vector that controls the input channel of the first convolutional layer. This latent vector has the same dimension with the input image channels, *e.g.* 3 for RGB images and 1 for gray images. Of course, the input images do not need to be pruned.
   (b) The latent vector attached to depth-wise convolution and group convolution. This latent vector controls the input channels per group. To compress depth-wise and group convolution, the number of groups is reduced, which is controlled by the latent vectors of the previous layer.

2.2 Residual block

The residual networks including ResNet, SRResNet, and EDSR are constructed by stacking a number of residual blocks. Depending on the dimension of the feature maps, the residual networks contain several stages with progressively reducing feature map dimension and increasing number of feature maps. (Note that the feature map dimension of EDSR and SRResNet does not change for all of the residual blocks. So there is only one stage for those networks.) For the residual blocks within the same stage, their output channels are correlated due to the existence of the skip connections. In order to prune the second convolution of the residual blocks within the same stage, we use a shared latent vector for

them. Thus, by only dealing with this shared latent vector, all of the second convolutions of the residual blocks can be pruned together. Please refer to Table S1 for the ablation study on latent vector sharing and non-sharing strategies.

2.3 Dense block

Similar to residual networks, DenseNet also contains several stages with different feature map configurations. But different from residual networks, each dense block concatenates its input and output to form the final output of the block. As a result, each dense block receives as input the outputs of all of the previous dense blocks within the same stage. Thus, the hypernetwork of a dense block also has to receive the latent vectors of the corresponding dense blocks as input.

2.4 Inverted residual block

The inverted residual blocks are just a special case of residual blocks. So how the latent vectors are shared across different blocks is the same with the normal residual blocks. Here we specifically address the sharing strategy within the block due to the existence of depth-wise convolution. The inverted residual block has the architecture of "point-wise conv + depth-wise conv + point-wise conv". As explained earlier, the latent vector of depth-wise convolution controls the input channels per group. Thus, the latent vector of the first point-wise convolution controls not only its output channels but also the input channels of the depth-wise convolution and the input channels of the second point-wise convolution. Thus, this latent vector has to be passed to the hypernetworks of the those convolutional layers.

2.5 Upsampler of super-resolution networks

The image super-resolution networks are attached with upsampler blocks at the tail of the networks to increase the spatial resolution of the feature map. For the scaling factor of $\times 4$, two upsamplers are attached and each doubles the spatial resolution. Each of the upsampler block contains a standard convolutional layer that increases the number of feature maps by a factor of 4 and a pixel shuffler that shuffles every 4 consecutive feature maps into the spatial dimension. Thus, the output channel of the convolutional layer in the upsmapler is correlated to its input channel. If one input channel is pruned, then four corresponding consecutive output channels should also be pruned. To achieve this control of pruning, a common latent vector is used for the input and output channels. The dimension of this latent vector is the same with the input channel size. This vector is repeated and interleaved to form the one controlling the output channel.

Table S1. Ablation study on ResNet56 for CIFAR10 image classification: exploring latent vector sharing strategy among correlated convolutional layers. "Share" denotes whether the latent vector sharing strategy described in Subsec. 2.2 is adopted. The $\ell_{2,1}$ regularizer means that when the latent vectors are not shared among the correlated layers, the group sparsity regularizer $\ell_{2,1}$ is enforced on their latent vectors. Otherwise, the normal ℓ_1 sparsity regularizer is used. λ and τ are the regularization factor and mask threshold introduced in the main paper. As shown in this table, the latent vector sharing strategy consistently outperforms the non-sharing counterparts

Share	Regularizer	λ	τ	Target FLOPs Ratio (%)	Actual FLOPs Ratio (%)	Actual Parameter Ratio (%)	Top-1 Error (%)
Yes	ℓ_1	2^{-4}	5^{-3}	38	39.96	52.49	7.41
No	ℓ_1	2^{-4}	5^{-3}	38	39.75	55.43	7.32
No	$\ell_{2,1}$	2^{-4}	5^{-3}	38	39.40	54.24	7.91
Yes	ℓ_1	3^{-4}	5^{-3}	38	39.60	49.00	6.86
No	ℓ_1	3^{-4}	5^{-3}	38	39.30	58.35	7.98
No	$\ell_{2,1}$	3^{-4}	5^{-3}	38	39.54	54.04	7.03
Yes	ℓ_1	2^{-4}	5^{-3}	50	51.27	56.84	7.13
No	ℓ_1	2^{-4}	5^{-3}	50	51.44	65.47	6.85
No	$\ell_{2,1}$	2^{-4}	5^{-3}	50	50.96	64.05	6.85
Yes	ℓ_1	3^{-4}	5^{-3}	50	51.68	57.74	6.52
No	ℓ_1	3^{-4}	5^{-3}	50	50.23	62.83	7.11
No	$\ell_{2,1}$	3^{-4}	5^{-3}	50	50.18	59.15	6.74

3 More Results

The ablation study on the latent vector sharing strategy is shown in Table S1. As shown in the table, the latent vector sharing strategy outperforms the non-sharing strategy consistently except for case $\lambda = 2^{-4}$, $\tau = 5^{-3}$ and 50% target FLOPs compression ratio. The inconsistency is largely due to the gap between the actual parameter compression ratio of different strategies. Due to this fact, various latent vector sharing rules are developed for easier and better automatic network pruning.

The main paper utilizes ℓ_1 regularizer to sparsify the the latent vectors. We tried to replace ℓ_1 regularizer with ℓ_2 regularizer and kept the same pruning strategy. The experiments are conducted on ResNet. The comparison results are shown in Table S2. Compared with ℓ_1 regularizer, slightly worse results can be observed for ℓ_2 regularizer. For ResNet-110 and ResNet-164, ℓ_2 regularizer leads to results comparable with ℓ_1 regularization but at a larger parameter budget. When the regularizer is changed to ℓ_2, the proximal operator becomes

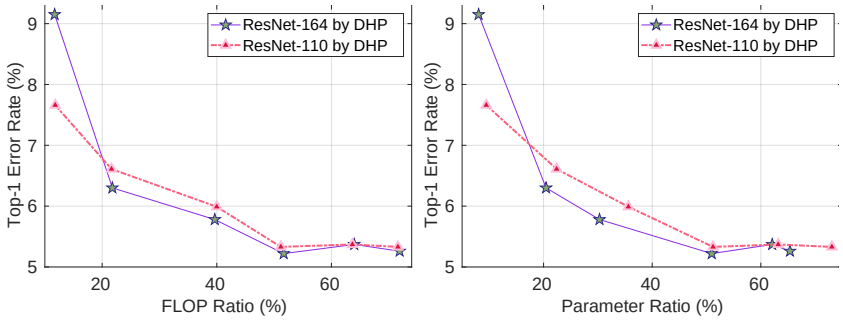
$$\mathbf{z}[k+1] = \left(1 - \frac{\lambda\mu}{\max\{\|\mathbf{z}[k+\Delta]\|, \lambda\mu\}}\right)\mathbf{z}[k+\Delta]. \tag{S1}$$

By the formulation and experiments, ℓ_1 norm leads to faster convergence. To have a reasonable convergence speed, the λ used for ℓ_2 regularizer is 20 times larger than that for ℓ_1 regularizer.

More results on ResNet-110 and ResNet-164 are shown in Fig. S1. When the compression ratio is not too severe (above 50%), the accuracy does not drop too

Table S2. Comparison between ℓ_1 norm and ℓ_2 norm regularization. The experiments are done on ResNet

Layer	Regularizer	Top1 Error	FLOPs	Params
20	ℓ_1	8.46	51.8	56.13
	ℓ_2	8.66	51.59	54.19
110	ℓ_1	5.73	51.62	54.13
	ℓ_2	5.77	51.37	72.37
164	ℓ_1	5.22	51.67	50.97
	ℓ_2	5.18	50.87	60.66



(a) FLOP compression ratio. (b) Parameter compression ratio.

Fig. S1. Top-1 error *vs.* FLOP and parameter compression ratio on ResNet-164 and ResNet-110

much. The extreme compression prunes about 90% FLOPs and parameters of the original network. For ResNet-164, the extreme compression only keeps 8.04% parameters. Thus, the drop in the accuracy is reasonable.

References

1. Agustsson, E., Timofte, R.: NTIRE 2017 challenge on single image super-resolution: Dataset and study. In: Proc. CVPRW (July 2017)
2. Bevilacqua, M., Roumy, A., Guillemot, C., Alberi-Morel, M.L.: Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In: Proc. BMVC (2012)
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. CVPR. pp. 770–778 (2016)
4. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proc. CVPR. pp. 2261–2269 (2017)
5. Huang, J.B., Singh, A., Ahuja, N.: Single image super-resolution from transformed self-exemplars. In: Proc. CVPR. pp. 5197–5206 (2015)
6. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep., Citeseer (2009)
7. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proc. ICCV. vol. 2, pp. 416–423 (July 2001)

8. Zeyde, R., Elad, M., Protter, M.: On single image scale-up using sparse-representations. In: International Conference on Curves and Surfaces. pp. 711–730. Springer (2010)