Supplementary Material: Domain Adaptation Regularization for Spectral Pruning

Laurent Dillard[1,†]
laurent.dillard@gmail.com
Yosuke Shinya[2]
yosuke.shinya.j7r@jp.denso.com
Taiji Suzuki[3,4]
taiji@mist.i.u-tokyo.ac.jp

[1] Behold.ai
New York, USA
[2] DENSO CORPORATION
Tokyo, Japan
[3] The University of Tokyo
Tokyo, Japan
[4] AIP-RIKEN
Tokyo, Japan

A Experimental settings on digits images

The output widths of the layers of the custom model are $(64, 64, 128, 1024, 1024, 10)$. Each convolutional or dense layer except for the output layer is followed by a batch normalization layer and a ReLU layer, and the feature generator is followed by a dropout layer.

B Experimental settings on natural images

We establish our comparison based on three datasets used in [1] experiments.

- **Oxford 102 Flowers** [2]: contains 8,189 images. Train and validation splits contain each 1,020 samples with each class equally represented. Test split contains 6,149 samples with classes not equally represented.
- **CUB-200-2011 Birds** [3]: contains 11,788 images (5,994 train, 5,794 test) of 200 bird species. Although each image is annotated with bounding box, part location (head and body), and attribute labels, those annotations were not used in our experiments.
- **Stanford 40 Actions** [4]: contains 9,532 images (4,000 train, 5,532 test) of 40 categories corresponding to different human actions. Classes are equally represented on the training set and all samples contain at least one human performing the corresponding action.

We used an ImageNet pre-trained VGG19 model that provided by the torchvision package of PyTorch. We trained for 10 epochs on Oxford-102 Flowers, 5 epochs on CUB-200 Birds, and 5 epochs on Stanford 40 Actions for fine-tuning before compression. On each dataset, we trained five models to mitigate randomness, and used the model that has the median accuracy of the five models as the original (uncompressed) model. When we fine-tune models after compression, we trained for 2 epochs and 5 epochs for VGG19 fc7 compression

†Work done as an intern at AIP-RIKEN.

k	4	8	16	32	64	128
Ours w/o FT.	68.9	69.3	70.0	71.2	72.0	72.5
Ours 1 epoch FT.	64.0	63.9	64.9	67.7	69.5	67.8
Ours 2 epochs FT.	67.3	68.9	70.3	69.9	70.8	70.6
Ours 5 epochs FT.	**71.9**	**72.9**	**73.2**	**73.5**	**74.2**	**74.2**

Table 7: Accuracy transition by fine-tuning. VGG19 fc7 compression on Oxford 102 Flowers. Test dataset accuracy results (%).

and VGG19 full compression, respectively. The dropout rate of VGG19 was set to 0.5. All models were trained using PyTorch.

In the case of VGG19 fc7 compression, compression rates for the total parameters of VGG19 models are limited to 11%–12% and FLOPs reduction is negligible, because layers before the fc7 layer are not compressed.

To compare our method with DALR, we need a way to fit the numbers of parameters because the two methods do not modify the architecture of the network the same way. DALR replaces a fully connected layer by two smaller layers without changing the next layer. Our method keeps the same number of layers but also affects the input dimension of the next layer. Therefore we proceeded the following way to compare the two methods objectively. A dimension k was set for the compression using DALR then k', the dimension to keep in our method resulting in an equal number of parameters, was determined accordingly. More precisely, the fc7 layer has a weight matrix of dimension $m \times m$ (because m and n in Fig. 1 are the same value for the fc7 layer) and the next layer has a weight matrix of dimension $m \times p$. Taking into account the biases we get the following equation:

$$k' = \frac{m(2k+1+p)+k}{m+1+p}. \tag{6}$$

The iterative process of the DALR paper [1] for determining compression rate for each layer is computationally inefficient. Thus, for VGG19 full compression, we did not determine compression rate for each layer automatically for both methods. Specifically, for compression by DALR, compression rate for fc8 is set to 0.5 (a modest value for not breaking output values), and compression rate for other layers (fc6 and fc7) is set so that total compression rate becomes ∼85%. For our method, compression rate for fully connected layers is set to 0.96, and compression rate for convolutional layers is set so that total compression rate becomes over 85%. Although the FLOPs reduction by DALR is negligibly small, our method reduces FLOPs by ∼19% thanks to the compression of convolutional layers.

C Fine-tuning results on natural images

In many cases, fine-tuning degrades the performance of our method. To investigate this phenomenon, we evaluated other models that fine-tuned for various epochs on Oxford 102 Flowers. The results are presented in Table 7. The accuracy drops in the first epoch and recovers in succeeding epochs. Considering these results, the learning rate and/or the dropout rate we used are too high for our method, and they cause the performance drops by keeping parameters away from optimal values. Tuning learning rates, dropout rates, and epochs for fine-tuning by cross-validation will further improve accuracy, though it takes much time.

References

[1] Marc Masana, Joost van de Weijer, Luis Herranz, Andrew D. Bagdanov, and Jose M. Alvarez. Domain-adaptive deep network compression. In *ICCV*, 2017.

[2] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics & Image Processing*, 2008.

[3] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[4] Bangpeng Yao, Xiaoye Jiang, Aditya Khosla, Andy Lai Lin, Leonidas Guibas, and Li Fei-Fei. Human action recognition by learning bases of action attributes and parts. In *ICCV*, 2011.