Supplementary Material for
Drop to Adapt: Learning Discriminative Features for Unsupervised Domain Adaptation

Seungmin Lee* Seoul National Univ.
Dongwan Kim* Seoul National Univ.
Namil Kim NAVER LABS
Seong-Gyun Jeong CODE42.ai

In this supplementary material, we derive an approximation of the channel-wise adversarial dropout (§ Appendix A) and provide implementation details of the experiments (§ Appendix B). Lastly, we provide additional GradCAM visualizations (§ Appendix C).

Appendix A. Approximation of Channel-wise Adversarial Dropout

Without loss of generality, the dropout mask $\mathbf{m}$ is vectorized to $\mathbf{v} = vec(\mathbf{m}) \in \mathbb{R}^{CHW}$. Similarly, $\mathbf{v}^0$ and $\mathbf{v}^s$ represent vectorized forms of $\mathbf{m}^0$ and $\mathbf{m}^s$, respectively. After vectorization of $\mathbf{m}$, we refer to the elements of $\mathbf{m}(i)$ with a set of indices π_i, and impose the channel-wise dropout constraints as follows:

$$\mathbf{v}[\pi_i] = vec(\mathbf{m}(i)) = \mathbf{0} \text{ or } \mathbf{1} \in \mathbb{R}^{HW}. \tag{a-1}$$

Let denote $d(\mathbf{x}, \mathbf{v}; \mathbf{v}^s) = D\left[h(\mathbf{x}; \mathbf{v}^s), h(\mathbf{x}; \mathbf{v})\right]$ as the divergence between two outputs using different dropout masks for convenience sake. Assuming d is a differentiable function with respect to $\mathbf{v}$, it can be approximated by a first-order Taylor expansion:

$$d(\mathbf{x}, \mathbf{v}; \mathbf{v}^s) \approx d(\mathbf{x}, \mathbf{v}^0; \mathbf{v}^s) + (\mathbf{v} - \mathbf{v}^0)^T \mathbf{J} \text{ where } \mathbf{J} = \nabla_{\mathbf{v}} d(\mathbf{x}, \mathbf{v}; \mathbf{v}^s)\Big|_{\mathbf{v}=\mathbf{v}^0}.$$

This equation shows that the Jacobian is proportional to the divergence. In other words,

$$d(\mathbf{x}, \mathbf{v}; \mathbf{v}^s) \propto \mathbf{v}^T \mathbf{J}. \tag{a-2}$$

We now see that the elements of $\mathbf{J}$ correspond to the *impact values*, which indicate the contribution of each activation over the divergence metric. Thus, for the given Jacobian, we can systematically modify the elements of $\mathbf{v}$ to maximize the divergence. However, due to the channel-wise dropout constraint from Eq. (a-1), we cannot modify each element individually. Instead, we reformulate the above relationship as:

$$d(\mathbf{x}, \mathbf{v}; \mathbf{v}^s) \propto \sum_i^C \mathbf{v}[\pi_i]^T \mathbf{J}[\pi_i]. \tag{a-3}$$

The impact value $\boldsymbol{s}$ of the i-th activation map in $h_l(\mathbf{x})$ can be defined as:

$$\boldsymbol{s}_i = \mathbf{1}^T \mathbf{J}[\pi_i], \tag{a-4}$$

Consequently, after computing the impact values $\boldsymbol{s}$, we solve 0/1 Knapsack problem as proposed in [3] while holding the constraints (a-1).

* denotes equal contribution.
This work was done while the authors were at NAVER LABS.

Table A-1. Hyperparameters

Experiment	Backbone	λ_1	λ_2	λ_3	$\bar{\delta_e}$	$\bar{\delta_c}$	T_r	ϵ
Small dataset								
SVHN $\rightarrow$ MNIST	9 Conv+1 FC	2	0.01	0.1	0.1	0.05	80	3.5
MNIST $\rightarrow$ USPS	3 Conv+2 FC	2	0.01	0	0.1	0.05	80	0
USPS $\rightarrow$ MNIST	3 Conv+2 FC	2	0.01	0.1	0.1	0.05	80	3.5
STL $\rightarrow$ CIFAR	9 Conv+1 FC	2	0.01	0.1	0	0.05	60	3.5
CIFAR $\rightarrow$ STL	9 Conv+1 FC	2	0.01	0	0	0.05	80	0
Large dataset								
VisDA-2017 Classification	ResNet-50	2	0.02	0.2	0.1	0.01	20	15
VisDA-2017 Classification	ResNet-101	2	0.02	0.2	0.1	0.01	30	15
Semantic segmentation								
GTA5 $\rightarrow$ Cityscapes	ResNet-50 FCN	2	0.01	0	0	0.02	1	0

Appendix B. Implementation Details

Training with DTA Loss

We apply a ramp-up factor on DTA loss function L_{DTA} to stabilize the training process. Instead of directly modulating the weight term λ_1, we gradually increase the perturbation magnitudes δ_e and δ_c which decide the number of hidden units to be eliminated. It allows us to regulate the consistency term, and to train the network being robust to various levels of perturbation generated by the adversarial dropout. We update the ramp-up factors with the following schedule:

$$\beta^{(t)} = \min(1, \tfrac{t}{T_r}), \tag{a-5}$$

where T_r represents the ramp-up period, and $\beta^{(t)}$ denotes the ramp up factor at the current epoch t. Finally, the perturbation magnitude is defined as:

$$\delta^{(t)} = \beta^{(t)}\bar{\delta}, \tag{a-6}$$

where $\bar{\delta}$ denotes the maximum level of perturbation. In practice, the same ramp-up period T_r is applied for both δ_e and δ_c.

Hyperparameters

Table A-1 presents the hyperparameters used in our experiments. We followed a similar hyperparameter search protocol as Shu *et al.* [5], where we sample a very small subset of labels from the target domain training set. For each objective function, we limit the hyperparameter search to a predefined set of values: $\lambda_1 = \{2\}$, $\lambda_2 = \{0, 0.01, 0.02\}$, $\lambda_3 = \{0, 0.1, 0.2\}$, $\delta_e = \{0, 0.1\}$, $\delta_c = \{0, 0.01, 0.02, 0.05\}$, and $\epsilon = \{0, 3.5, 15\}$. Furthermore, we provide the rest of parameters related to network training for each experimental set up.

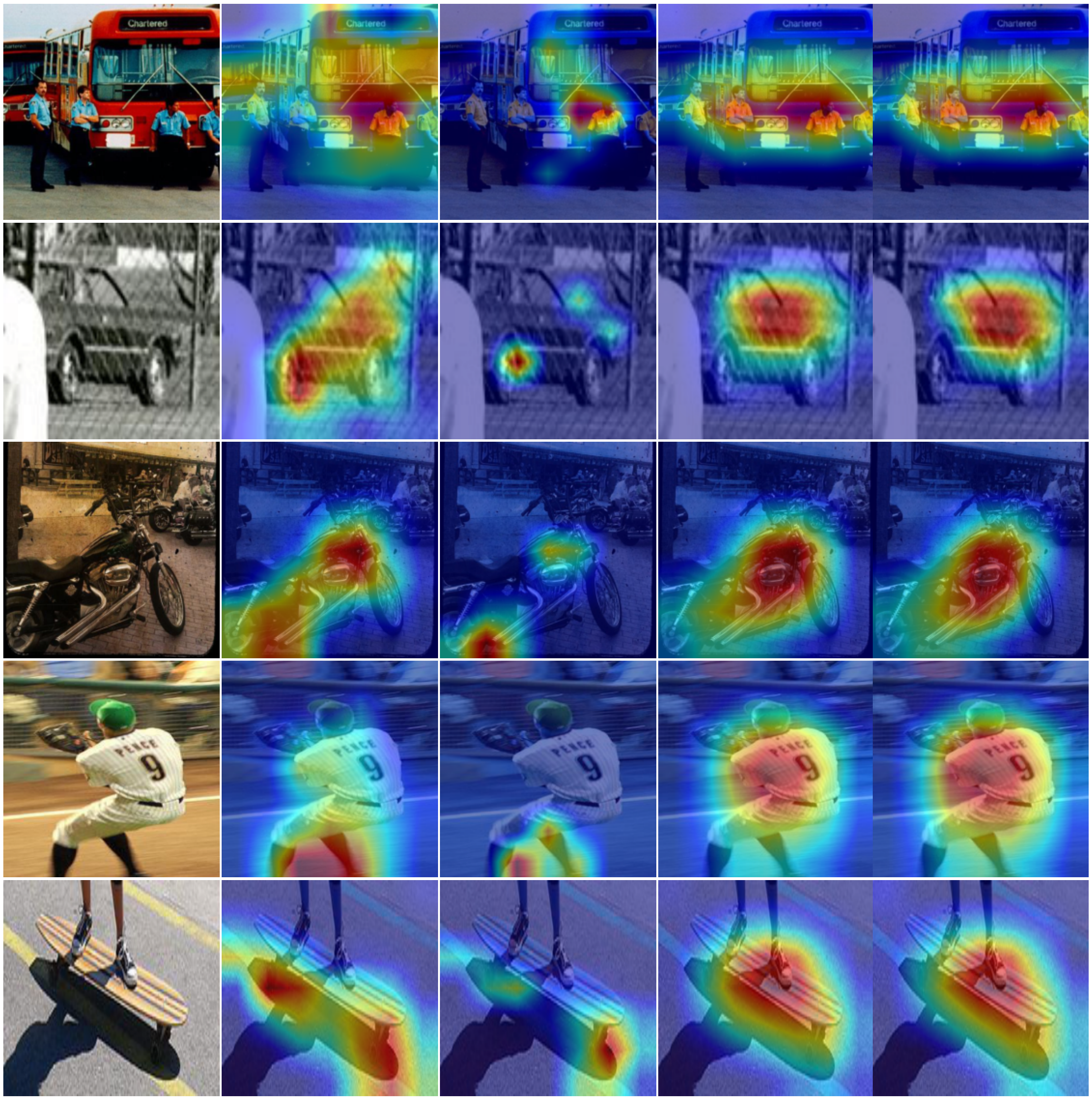
Small dataset. All small dataset experiments were trained for 90 epochs, using Adam optimizer [1] with an initial learning rate of 0.001, decaying by a factor of 0.1 every 30 epochs.

Large dataset. We conducted the VisDA-2017 classification experiments on ResNet-50 and ResNet-101. We trained the networks for 20 epochs using Stochastic Gradient Descent (SGD) with a momentum value of 0.9 and an initial learning rate of 0.001, which decays by a factor of 0.1 after 10th epoch.

Semantic segmentation. The semantic segmentation task for domain adaptation from GTA5 to Cityscapes was trained for 5 epochs using SGD with a momentum of 0.9. Since FCN [2] has no fully-connected layers, $\bar{\delta_e}$ was automatically set to 0. In addition, we used the maximum $\bar{\delta_c}$ value from the beginning because the task-specific objective were dominant in the early stages of training. In this experiment, we turned off VAT objective which hinders from learning the segmentation task.

Appendix C. Additional GradCAM visualizations

In Figure A-1, we provide additional GradCAM visualizations to highlight the effects of adversarial dropout.



(a) Input (b) SO (c) SO+AdD (d) DTA (e) DTA+AdD

Figure A-1. Effect of adversarial dropout, visualized by GradCAM [4].

References

[1] Diedrik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[2] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[3] Sungrae Park, JunKeon Park, Su-Jin Shin, and Il-Chul Moon. Adversarial dropout for supervised and semi-supervised learning. In *AAAI*, 2018.

[4] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE ICCV*, 2017.

[5] Rui Shu, Hung Bui, Hirokazu Narui, and Stefano Ermon. A DIRT-T approach to unsupervised domain adaptation. In *ICLR*, 2018.