AutoDIAL: Automatic DomaIn Alignment Layers Supplementary Material

Fabio Maria Carlucci
Sapienza, Roma, Italy

Lorenzo Porzi
IRI CSIC-UPC, Barcelona, Spain
Mapillary, Graz, Austria

Barbara Caputo
Sapienza, Roma, Italy

Elisa Ricci
FBK, Trento, Italy
University of Perugia, Italy

Samuel Rota Bulò
FBK, Trento, Italy
Mapillary, Graz, Austria

Abstract

This document provides the following additional contributions to our ICCV 2017 submission:

- *in Section 1, we provide explicit formulas for the batch statistics in our DA-layers as well as the layer's back-propagation equations;*
- *in Section 2, we provide results on the SVHN – MNIST benchmark.*
- *in Section 3, we provide some examples of feature distributions learned by AutoDIAL – Inception-BN on the Office 31 dataset.*

1. DA-layers formulas

We rewrite Eq. (1) of the main paper to make sample indexes explicit:

$$\begin{aligned} y_i^s &= \texttt{DA}(x_i^s;\alpha) = \frac{x_i^s - \mu_{st,\alpha}}{\sqrt{\epsilon + \sigma^2_{st,\alpha}}}\,, \\ y_i^t &= \texttt{DA}(x_i^t;\alpha) = \frac{x_i^t - \mu_{ts,\alpha}}{\sqrt{\epsilon + \sigma^2_{ts,\alpha}}}\,. \end{aligned} \tag{1}$$

Using this notation, the batch statistics are

$$\begin{aligned} \mu_{st,\alpha} &= \frac{\alpha}{n_s}\sum_{i=1}^{n_s} x_i^s + \frac{1-\alpha}{n_t}\sum_{i=1}^{n_t} x_i^t\,, \\ \mu_{ts,\alpha} &= \frac{1-\alpha}{n_s}\sum_{i=1}^{n_s} x_i^s + \frac{\alpha}{n_t}\sum_{i=1}^{n_t} x_i^t\,, \\ \sigma^2_{st,\alpha} &= \frac{\alpha}{n_s}\sum_{i=1}^{n_s} (x_i^s - \mu_{st,\alpha})^2 + \frac{1-\alpha}{n_t}\sum_{i=1}^{n_t} (x_i^t - \mu_{st,\alpha})^2\,, \\ \sigma^2_{ts,\alpha} &= \frac{1-\alpha}{n_s}\sum_{i=1}^{n_s} (x_i^s - \mu_{ts,\alpha})^2 + \frac{\alpha}{n_t}\sum_{i=1}^{n_t} (x_i^t - \mu_{ts,\alpha})^2\,, \end{aligned} \tag{2}$$

where n_s and n_t are, respectively, the number of source and target samples in a batch. The partial derivatives of the statistics w.r.t. the inputs are

$$\begin{aligned} \frac{\partial \mu_{st,\alpha}}{\partial x_j^s} &= \frac{\alpha}{n_s}\,, & \frac{\partial \mu_{st,\alpha}}{\partial x_j^t} &= \frac{1-\alpha}{n_t}\,, \\ \frac{\partial \mu_{ts,\alpha}}{\partial x_j^s} &= \frac{1-\alpha}{n_s}\,, & \frac{\partial \mu_{ts,\alpha}}{\partial x_j^t} &= \frac{\alpha}{n_t}\,, \end{aligned} \tag{3}$$

$$\begin{aligned} \frac{\partial \sigma_{st,\alpha}}{\partial x_j^s} &= 2\frac{\alpha}{n_s}(x_j^s - \mu_{st,\alpha})\,, \\ \frac{\partial \sigma_{st,\alpha}}{\partial x_j^t} &= 2\frac{1-\alpha}{n_t}(x_j^t - \mu_{st,\alpha})\,, \\ \frac{\partial \sigma_{ts,\alpha}}{\partial x_j^s} &= 2\frac{1-\alpha}{n_s}(x_j^s - \mu_{ts,\alpha})\,, \\ \frac{\partial \sigma_{ts,\alpha}}{\partial x_j^t} &= 2\frac{\alpha}{n_t}(x_j^t - \mu_{ts,\alpha})\,. \end{aligned} \tag{4}$$

The partial derivatives of the loss L w.r.t. the inputs are

$$\frac{\partial L}{\partial x_j^s} = \frac{1}{\sqrt{\epsilon+\sigma_{st,\alpha}^2}}\left[\frac{\partial L}{\partial y_j^s} - \frac{\alpha}{n_s}\left(\sum_{i=1}^{n_s}\frac{\partial L}{\partial y_i^s} + y_j^s\sum_{i=1}^{n_s} y_i^s\frac{\partial L}{\partial y_i^s}\right)\right] - \frac{1}{\sqrt{\epsilon+\sigma_{ts,\alpha}^2}}\frac{1-\alpha}{n_s}\left(\sum_{i=1}^{n_t}\frac{\partial L}{\partial y_i^t} + y_j^{st}\sum_{i=1}^{n_t} y_i^t\frac{\partial L}{\partial y_i^t}\right),$$

$$\frac{\partial L}{\partial x_j^t} = \frac{1}{\sqrt{\epsilon+\sigma_{ts,\alpha}^2}}\left[\frac{\partial L}{\partial y_j^t} - \frac{\alpha}{n_t}\left(\sum_{i=1}^{n_t}\frac{\partial L}{\partial y_i^t} + y_j^t\sum_{i=1}^{n_t} y_i^t\frac{\partial L}{\partial y_i^t}\right)\right] - \frac{1}{\sqrt{\epsilon+\sigma_{ts,\alpha}^2}}\frac{1-\alpha}{n_t}\left(\sum_{i=1}^{n_s}\frac{\partial L}{\partial y_i^s} + y_j^{ts}\sum_{i=1}^{n_s} y_i^s\frac{\partial L}{\partial y_i^s}\right), \tag{5}$$

where y_i^{st} and y_i^{ts} are "cross-normalized" outputs

$$y_i^{st} = \frac{x_i^s - \mu_{ts,\alpha}}{\sqrt{\epsilon+\sigma_{ts,\alpha}^2}}, \quad y_i^{ts} = \frac{x_i^t - \mu_{st,\alpha}}{\sqrt{\epsilon+\sigma_{st,\alpha}^2}}. \tag{6}$$

Using these definitions, one can also compute the partial derivative of L w.r.t. the domain mixing parameter α as

$$\begin{aligned}\frac{\partial L}{\partial \alpha} = &\left(\frac{1}{n_t}\sum_{i=1}^{n_t} y_i^{ts} - \frac{1}{n_s}\sum_{i=1}^{n_s} y_i^s\right)\sum_{i=1}^{n_s}\frac{\partial L}{\partial y_i^s} \\ &+ \left(\frac{1}{n_t}\sum_{i=1}^{n_t}(y_i^{ts})^2 - \frac{1}{n_s}\sum_{i=1}^{n_s}(y_i^s)^2\right)\frac{1}{2}\sum_{i=1}^{n_s} y_i^s\frac{\partial L}{\partial y_i^s} \\ &+ \left(\frac{1}{n_s}\sum_{i=1}^{n_s} y_i^{st} - \frac{1}{n_t}\sum_{i=1}^{n_t} y_i^t\right)\sum_{i=1}^{n_t}\frac{\partial L}{\partial y_i^t} \\ &+ \left(\frac{1}{n_s}\sum_{i=1}^{n_s}(y_i^{st})^2 - \frac{1}{n_t}\sum_{i=1}^{n_t}(y_i^t)^2\right)\frac{1}{2}\sum_{i=1}^{n_t} y_i^t\frac{\partial L}{\partial y_i^t}.\end{aligned} \tag{7}$$

Note that, as for standard Batch Normalization, the gradients do not depend on the layer's inputs, allowing for in-place computation where permitted by the deep learning framework of choice.

2. Results on the SVHN – MNIST benchmark

In this section we report the results we obtain in the SVHN [8] to MNIST [5] transfer benchmark. We follow the experimental protocol in [3], using all SVHN images as the source domain and all MNIST images as the target domain, and compare with the following baselines: CORAL [10]; the Deep Adaptation Networks (DAN) [7]; the Domain-Adversarial Neural Network (DANN) in [3]; the Deep Reconstruction Classification Network (DRCN) in [4]; the Domain Separation Networks (DSN) in [1]; the Asymmetric Tri-training Network (ATN) in [9].

As in all baselines, we adopt the network architecture in [2], adding DA-layers after each layer with parameters.

Method	Accuracy
CORAL [6]	63.1
DAN [7]	71.1
DANN [3]	73.9
DRCN [4]	82.0
DSN [1]	82.7
ATN [9]	86.2
AutoDIAL	**90.3**

Table 1. Results on the SVHN to MNIST benchmark.

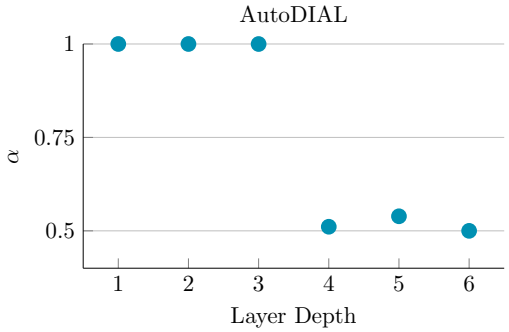

Figure 1. α parameters learned on the SVHN – MNIST dataset, plotted as a function of layer depth.

Training is performed from scratch, using the same meta-parameters as for AlexNet (see Main Paper), with the following exceptions: initial learning rate $l_0 = 0.01$; 25 epochs; learning rate schedule defined by $l_p = l_0/(1+\gamma p)^\beta$, where $\gamma = 10$, $\beta = 0.75$ and p is the learning progress linearly increasing from 0 to 1.

As shown in Table 1, we set the new state of the art on this benchmark. It is worth of note that AutoDIAL also outperforms the methods, such as ATN and DSN, which expand the capacity of the original network by adding numerous learnable parameters, while only employing a single extra learnable parameter in each DA-layer. The α parameters learned by AutoDIAL on this dataset are plotted in Fig. 1. Similarly to the case of AlexNet and Inception-BN on the Office-31 dataset, the network learns higher values of α in the bottom of the network and lower values of α in the top. In this case, however, we observe a steeper transition from 1 to 0.5, which interestingly corresponds with the transition from convolutional to fully-connected layers in the network.

3. Feature distributions

In this section we study the distributions of a set of randomly sampled features from different layers of AutoDIAL – Inception-BN, learned on the Amazon–DSLR task of the Office 31 dataset. In Fig. 2 we compare the histograms of these features, computed on the whole source and target sets

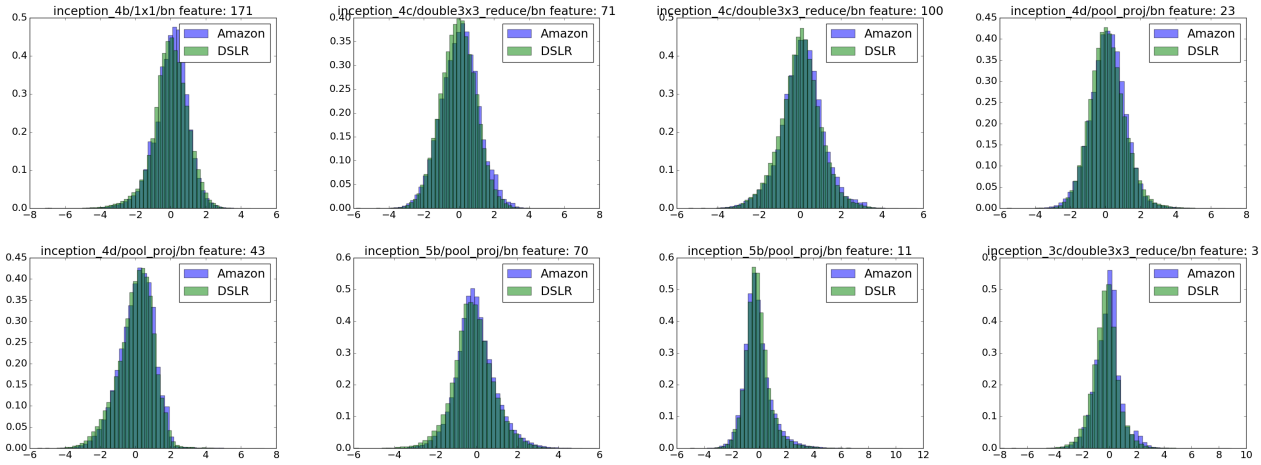

Figure 2. Distributions of randomly sampled source/target features from different layers of AutoDIAL – Inception-BN learned on the Amazon–DSLR task of the Office 31 dataset (best viewed on screen).

and taken *after* the DA-layers. The plots clearly show the aligning effect of our DA-layers, as most histograms are very closely matching. It is also interesting to note how the alignment effect seems to be mostly independent of the particular shape the distributions might take.

References

[1] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. In *NIPS*, 2016. 2

[2] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. *ICML*, 2015. 2

[3] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016. 2

[4] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *ECCV*, 2016. 2

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[6] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016. 2

[7] M. Long and J. Wang. Learning transferable features with deep adaptation networks. In *ICML*, 2015. 2

[8] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011. 2

[9] K. Saito, Y. Ushiku, and T. Harada. Asymmetric tri-training for unsupervised domain adaptation. *arXiv preprint arXiv:1702.08400*, 2017. 2

[10] B. Sun, J. Feng, and K. Saenko. Return of frustratingly easy domain adaptation. In *AAAI*, 2016. 2