BATCH NORMALIZATION EMBEDDINGS FOR DEEP DOMAIN GENERALIZATION

Anonymous authors
Paper under double-blind review

1 SUPPLEMENTARY MATERIAL

We provide supplementary material to validate our method. Sec. 1.1 provides an algorithmic overview of the proposed training policy; Sec. 1.3 shows additional results obtained with ResNet-18 (He et al. (2016)) on Office-31 (Saenko et al. (2010)) and Office-Caltech (Gong et al. (2012)); In Sec. 1.4 we conduct qualitative studies to verify our choices in terms of batch sizes and distance measures. Finally, we provide a qualitative visualization using t-sne (Maaten & Hinton (2008)) of the distribution of our batch normalization embeddings at different depths in the network.

N.B.: Blue references point to the original manuscript.

1.1 TRAINING POLICY

We here provide a formalization of the distance training policy described in Sec. 3.4.

Let $T = \{b_d\}_{d \in \mathcal{D}}$ be a training batch composed of K domain batches, each containing n samples from the corresponding domain d: $b_d = \{(x_d^i, y_d^i)\}_{i=1}^n$. In Alg. 1 we illustrate the training procedure for a single training batch T using the same notation as in the original manuscript.

Algorithm 1 Training Step for a batch T

1: **for** $b_d \in T$ **do** ▷ **for every domain batch**
2: Collect domain batch statistics $(\tilde{\mu}_d, \tilde{\sigma}_d)$ ▷ forward propagation
3: $\hat{\mu}_d^l \longleftarrow 0.99\hat{\mu}_d^l + 0.01\tilde{\mu}_d^l \quad \forall l \in \mathcal{B}$ ▷ update domain population mean
4: $(\hat{\sigma}_d^l)^2 \longleftarrow 0.99(\hat{\sigma}_d^l)^2 + 0.01(\tilde{\sigma}_d^l)^2 \quad \forall l \in \mathcal{B}$ ▷ update domain population variance
5: $e_d^l \longleftarrow (\hat{\mu}_d^l, (\hat{\sigma}_d^l)^2) \quad \forall l \in \mathcal{B}$ ▷ update domain layer embeddings
6: $e_d \longleftarrow [e_d^1, e_d^2, ..., e_d^L]$ ▷ update domain embedding
7: **for** $(x_t, y_t) \in T$ **do** ▷ **for every sample**
8: Collect instance statistics $(\mu_t, {\sigma_t}^2)$ ▷ forward propagation
9: $r_t^l \longleftarrow (\mu_t^l, {\sigma_t^l}^2) \quad \forall l \in \mathcal{B}$ ▷ define target layer embeddings
10: $r_t \longleftarrow [r_t^1, r_t^2, ..., r_t^L]$ ▷ define target embedding
11: $D_{\mathcal{L}}(e_d, r_t) = \sum_{l \in \mathcal{B}} \mathcal{W}(e_d^l, r_t^l) \quad \forall d \in D$ ▷ compute domain distances
12: $w_d^t = \frac{1}{D_{\mathcal{L}}(e_d, r_t)} \quad \forall d \in D$ ▷ compute domain similarities
13: $f_d^t \longleftarrow f(x_t|d) \quad \forall d \in D$ ▷ compute domain-specific predictions
14: $f(x_t) = \frac{\sum_{d \in \mathcal{D}} w_d^t f_d^t}{\sum_{d \in \mathcal{D}} w_d^t}$ ▷ compute final predictions
15: $L(\theta; T) = \sum_{(x_t, y_t) \in T} \mathbb{XE}(f(x_t), y_t)$ ▷ compute cross-entropy loss
16: $\theta \longleftarrow \theta - \eta \cdot L(\theta; T)$ ▷ update weights

During every training step, first, the domain batches are first propagated to update the corresponding domain embedding e_d *(l:2-6)*. Then, each individual sample x_t is propagated using instance normalization to collect its instance statistics $(\mu_t^l, \sigma 1_t^{l^2}) \quad \forall l \in \mathcal{B}$ *(l:8)*. Given the statistics we compute the target embedding r_t *(l:9-10)* and the domain similarities w_d^t *(l:12)*, as in Sec. 3.3. Each sample is propagated under K different domain assumptions (*i.e.,* through the corresponding domain-specific branches) *(l:13)*. The resulting domain-specific predictions are weighted according to Eq. 11 to compute the final prediction *(l:14)*. Finally, the cross-entropy loss between the final predictions $f(x_t)$ and the corresponding ground truths y_t is computed *(l:15)* and back-propagated to update the

Table 1: State-of-the-art comparison on PACS with AlexNet.

Method	Art	Cartoon	Photo	Sketch	Avg. DA	Avg.	$\Delta\%$
DICA - Muandet et al. (2013)	64.6	64.5	91.8	51.1	68.7	68.0	-1.02
D-MTAE - Ghifary et al. (2015)	60.3	58.7	91.1	47.9	68.7	64.5	-6.11
DSN - Bousmalis et al. (2016)	61.1	66.5	83.3	58.6	68.7	67.4	-1.89
TF-CNN - Li et al. (2017)	62.9	67.0	89.5	57.5	67.1	69.2	+3.13
CIDDG - Li et al. (2018c)	62.7	69.7	78.7	64.5	71.7	68.9	-3.91
Fusion - Mancini et al. (2018a)	64.1	66.8	90.2	60.1	67.1	70.3	+4.77
CrossGrad - Shankar et al. (2018)	64.1	66.8	90.2	60.1	68.7	70.3	+2.33
MetaReg - Balaji et al. (2018)	69.8	70.4	91.1	59.3	69.3	72.6	+4.76
MLDG - Li et al. (2018a)	66.2	66.9	88.0	59.0	67.2	70.0	+4.17
Epi-FCD - Li et al. (2019)	64.7	72.3	86.1	65.0	68.7	72.0	+4.80
JiGen - Carlucci et al. (2019)	67.6	71.7	89.0	65.2	71.5	73.4	+2.66
MASF - Dou et al. (2019)	**70.4**	**72.5**	**90.7**	**67.3**	71.7	**75.2**	+4.88
DeepAll	64.4	65.4	88.0	53.8	-	67.9	-
BNE (*Ours*)	66.7	65.7	89.5	66.8	67.9	72.2	**+6.33**

weights θ of the model *(l:16)*. Applying this procedure during training encourages the creation of a batch normalization latent space.

1.2 Training Settings

Coherently with other works, we evaluate both the AlexNet (Krizhevsky et al. (2012)) and the more recent ResNet-18 (He et al. (2016)) architecture. Before training each network, we initialize them with pre-trained weights on ImageNet and fine-tune the last fully-connected layer on the dataset of interest for 20 epochs. To train AlexNet (Krizhevsky et al. (2012)), we use SGD as optimizer with momentum 0.95 and L2 regularization on network weights with weight decay 5×10^{-5}. The initial learning rate is 10^{-3}, exponentially decayed with decay rate 0.95. ResNet-18 is trained with Adam (Kingma & Ba (2014)) and weight decay 10^{-6}. The initial learning rate is 10^{-4}. Coherently with previous works (Carlucci et al. (2017b;a); Mancini et al. (2018c)), we also compute gradients through the mean and standard deviation computation for the batch normalization layers. All the input images are normalized according to the statistics computed on ImageNet. At training time, data augmentation is performed by first resizing the input image to 256, then randomly cropping to 224×224 for ResNet-18 and 227×227 for AlexNet; finally, a random horizontal flip is performed. Every training batch is composed of 16 samples per domain for ResNet-18 and 6 for AlexNet.

All the models are implemented in Tensorflow 2.0 (Abadi et al. (2015)). We initialize both AlexNet and ResNet-18 using the publicly available Caffe weights pre-trained on ImageNet, after carefully converting them.[1]

1.3 Additional Results

Table 2: State-of-the-art comparison on Office-31 with ResNet-18.

Method	Amazon	Dslr	Webcam	Avg. DA	Avg.	$\Delta\%$
DeepAll	55.1	99.0	92.6	-	82.2	-
BNE (*Ours*)	**55.5**	**99.3**	**95.4**	82.2	**83.4**	+1.42

We here provide additional results with the ResNet-18 (He et al. (2016)) architecture for the dataset Office-31 (Saenko et al. (2010)) and with the AlexNet (Simon et al. (2016)) architecture for PACS (Li et al. (2017)). In the original manuscript, we already provide results with AlexNet and ResNet-18 respectively to compare against recently published works. Moreover, we expand the experimental setting with the addition of the dataset Office-Caltech (Gong et al. (2012)), for which we present results with both ResNet-18 and AlexNet.

[1]ResNet-18 and AlexNet ImageNet weights available at `https://github.com/heuritech/convnets-keras` and `https://github.com/cvjena/cnn-models`.

Table 3: State-of-the-art comparison on Office-Caltech with AlexNet.

Method	Amazon	Caltech	Dslr, Webcam	Amazon, Caltech	Avg. DA	Avg.	$\Delta\%$
UB - Khosla et al. (2012)	91.0	86.0	80.5	70.0	84.5	81.9	-3.08
DSN - Bousmalis et al. (2016)	-	-	85.8	81.2	84.5	-	-
MTAE - Ghifary et al. (2015)	93.1	86.2	85.3	80.5	84.5	86.3	+2.13
DGLRC - Ding & Fu (2017)	**94.2**	**87.6**	86.3	82.2	84.5	87.6	+3.67
MDA - Hu et al. (2019)	93.5	86.9	84.9	82.6	84.5	87.0	+2.96
CIDG - Li et al. (2018b)	93.2	85.1	83.7	65.91	84.5	82.0	-2.96
MCIT - Rahman et al. (2019)	93.3	86.3	85.2	82.7	84.5	86.9	+2.84
DeepAll	91.7	82.1	83.4	84.5	-	85.4	-
BNE (*Ours*)	93.7	85.9	**89.8**	**87.7**	85.4	**89.3**	**+4.57**

Table 4: State-of-the-art comparison on Office-Caltech with ResNet-18.

Method	Amazon	Caltech	Dslr, Webcam	Amazon, Caltech	Avg. DA	Avg.	$\Delta\%$
DeepAll	92.7	83.1	85.3	80.7	-	85.5	-
BNE (*Ours*)	**92.9**	**87.4**	**93.0**	**87.3**	85.5	**90.2**	+5.50

1.3.1 PACS

In Tab. 1, we extend the comparison on PACS considering AlexNet to compare against a vast literature of published works relying on this older architecture. Once again our proposal achieves absolute performance comparable to the state of the art even if starting from a weaker baseline. Indeed when comparing the relative gain in performance provided by our method ($\Delta\%$), we are clearly outperforming any previously published solutions with an increase of $+6.33\%$, while the second best obtains $+4.88\%$. Once again, when considering *Sketch* as unseen domain our method can boost the performance by a $+13\%$ absolute gain in accuracy over our baseline.

1.3.2 OFFICE-31

In Tab. 2, we extend the comparison on Office-31 considering ResNet-18 as it is a good example of a modern architecture with native batch normalization layers. The results confirms that our method is able to improve performances over DeepAll across all three tests.

1.3.3 OFFICE-CALTECH

Office-Caltech (Gong et al. (2012)) is a variant of Office-31 featuring one additional domain, derived from the Caltech-256 dataset (Griffin et al. (2007)). The dataset is composed of the 10 categories shared between Caltech-256 and the domains in Office-31. We introduce the results on the Office-Caltech dataset since it represents a challenging setting with a larger number of domains with respect to those analysed in the main paper; nominally 4 instead of 3. For Office-Caltech we follow the same evaluation procedure as for the other datasets, but we also test following a *leave-two-domain-out* procedure to compare against other published results. In Tab. 3 and Tab. 4 we show the results for the AlexNet and the ResNet-18 architecture respectively. The standard evaluation on this dataset enumerates the cases with a single target domain, either *Amazon* or *Caltech*, and a pair of targets: *Dslr-Webcam* and *Amazon-Caltech*. We use both AlexNet and ResNet-18 initialized with ImageNet weights to compare with published results and train it with our method for 100 epochs. Tab. 3 shows that our approach obtains the best average accuracy and the best gain with respect to the baseline. The gain provided by our method are especially evident in the challenging scenario when we consider 2 domains as targets, *e.g.,* $+6.4\%$ absolute accuracy in the *Dslr-Webcam* case. The same good property remains also when considering ResNet as architecture in Tab. 4, with a clear +5.5% gain over DeepAll.

1.4 ABLATION STUDY

We here provide additional ablation studies to better highlight different characteristics of our method with respect to the chosen batch size and the distance measures used in the latent space. All the following studies are performed with the modified version of AlexNet (Simon et al. (2016)) which includes batch normalization layers after every convolutional layer. The experiments are conducted on the PACS dataset (Li et al. (2017)).

1.4.1 BATCH SIZE

Table 5: Comparison of different batch sizes (per domain) on PACS with Alexnet.

Method	Batch Size	Art	Cartoon	Photo	Sketch	Average
DeepAll	-	64.4	65.4	88.0	53.8	67.9
BNE (*Ours*)	16	65.7	66.5	88.9	56.0	69.3
BNE (*Ours*)	32	67.9	65.7	89.4	62.3	71.3
BNE (*Ours*)	64	66.7	65.7	89.5	66.8	72.2

We study the impact of different batch sizes on the performance of our method in Tab. 5. As expected and already documented in several recent works leveraging batch normalization layers (Bjorck et al. (2018); Wu & He (2018)), the larger the batch size is the better the generalization capability is. In particular for our method the bigger is the batch size used at training time, the better are the approximation of the true population statistics (*i.e.,* , the better are the domain embeddings e_d). This translates in better final performance as detailed in Tab. 5 where we can observe an increment of $+2.9$ *Average* accuracy between using batch size 16 and 64.

1.4.2 DISTANCE

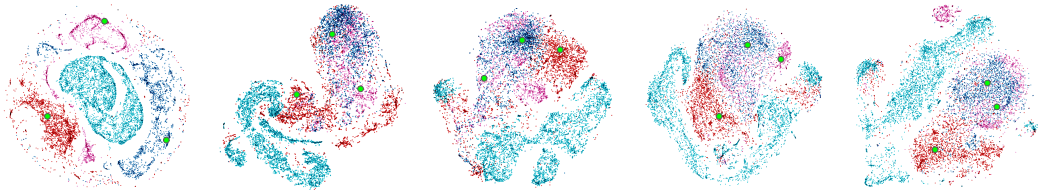
Table 6: Comparison of different distance measures with our method on PACS with Alexnet.

Method	Distance	Art	Cartoon	Photo	Sketch	Average
DeepAll	-	64.4	65.4	88.0	53.8	67.9
BNE (*Ours*)	Uniform	64.9	64.0	88.7	61.7	69.8
BNE (*Ours*)	Bhattacharyya	66.3	64.6	89.4	64.3	71.2
BNE (*Ours*)	Wasserstein	**66.7**	**65.7**	**89.5**	**66.8**	**72.2**

In Sec. 3.3 we explain that we use the Wasserstein distance to measure the distance between two multivariate gaussian distributions. Before picking this distance we have conducted a detailed study to select the best option for our task. We report the result of this study on Tab. 6. We considered three difference distance options: using an fixed value for the distance (*Uniform*), in this case all domains will be considered at the same distance from the test sample; the Bhattacharya distance and the Wasserstein distance. The Wasserstein distance shows to consistently improve over the baseline for domain generalization DeepAll, and over any other distance measure both on average and on any specific left-out domain. The significance of a specific distance measure is proven by the better results obtained by both the Bhattacharyya and the Wasserstein distance over the Uniform distance case, meaning that the potentiality of our method derives from the accurate distance measuring system rather than from the lightweight ensemble itself.

1.4.3 ACTIVATIONS DISTRIBUTION

In Fig. 1 we provide a visualization with different two-dimensional t-SNE (Maaten & Hinton (2008)) of the batch normalization embeddings of each sample in the PACS dataset. Different colors represent different domains: *Art Painting* (pink), *Cartoon* (red), *Photo* (blue), *Sketch* (cyan). Each figure illustrates the sample embeddings right before a specific batch normalization layer at different depths



(a) BN-1 (b) BN-2 (c) BN-3 (d) BN-4 (e) BN-5

Figure 1: T-SNEs of the distribution of samples from PACS Li et al. (2017) at different depths in our modified AlexNet Simon et al. (2016) architecture. Each point represents the instance statistics of a sample at a certain level in the network. The model is trained on the domains *Art Painting* (pink), *Cartoon* (red), *Photo* (blue); *Sketch* (cyan) is left unseen. Population statistics for seen domains are shown as green dots.

Table 7: Comparison of different variants of our method on PACS with Alexnet.

Method	DT	Warm-up	Art	Cartoon	Photo	Sketch	Avg.
DeepAll	-	-	64.4	65.4	88.0	53.8	67.9
(a) BNE	✗	✗	64.4	65.9	**89.6**	54.2	68.53
(b) BNE	✓	✗	63.7	**67.9**	84.6	66.6	70.7
(c) BNE	✓	✓	**66.7**	65.7	89.5	**66.8**	**72.2**

in the AlexNet architecture. For this test, the model is trained on *Art Painting* (pink), *Cartoon* (red), and *Photo* (blue), while *Sketch* is the unseen domain.

Before the first domain-specific normalization (*i.e.,* Fig. 1 (a)), features from different domains can be easily clustered together. After the first domain-specific normalization, deep features from *Art Painting* and *Photo* start to be similar. On the other side, features from *Cartoon* and from the unseen domain *Sketch* are always well-distinguishable from any other domain at all the considered layers. This match quite well the dissimilarity between domains in the PACS dataset as judged by a human.

The population statistics of the source domains always fall inside the corresponding cluster, proving how they represent a good descriptor of the corresponding domain.

1.4.4 Method Components

In the main paper we measured the contribution to the final performance of the different components of our methods. The proposed setting leveraged the PACS dataset and the ResNet-18 architecture. We here consider ablation experiments on the PACS dataset using the AlexNet architecture and report the results in Tab.3, comparing again with the DeepAll baseline. On row (a) we show the performance gained by using separate batchnorm statistics for the different train domains and using the projection and weighting strategy described in Sec. 3.3; row (b) extends the method above by using the *distance weighting at training time* (DT) as described in Sec. 3.4; finally, row (c) includes a warm-up phase in the training of the model to make population statistics converge to stable values before starting the distance training. By comparing the average accuracy (Avg.) across the four possible target sets, it is clear how every component contributes to an increase in performance with respect to the baseline.

1.5 Latent Space Validation

We now want to investigate how well we are able to collect domain specific attributes of samples by projecting them to the batchnorm latent space. We trained ResNet-18 until convergence without distance training and warm-up on the PACS dataset considering *Photo* or *Sketch* as unseen domains. Once trained, we forward every training sample through the network and compute its instance statistics to project it to the batchnorm latent space. After the projection we measure the distance from

Table 8: Analysis of the average similarity value as domain classification metrics with ResNet-18 on PACS without distance training. Classified domain is in bold.

(a) *Photo* unseen.

Source	Art	Cartoon	Sketch
Art	**8.24**	5.35	4.21
Cartoon	6.58	**7.02**	5.97
Sketch	3.94	4.56	**10.19**

(b) *Sketch* unseen.

Source	Art	Cartoon	Photo
Art	**1.18**	0.70	1.15
Cartoon	0.94	**1.02**	0.90
Photo	1.19	0.70	**1.25**

every domain embedding: if the closest domain matches the real domain, then the latent space effectively represents membership to a certain domain. In Tab. 8 we report the average value of the reciprocal of the distance for every training sample with respect to the centroid of the three training domains. The higher values on the diagonal confirm our intuition that the batchnorm latent space can be used to implicitly encode domain attributes.

1.6 Domain Discovery Net

Table 9: Comparison BNE and DNet with or without cross-entropy loss applied also on domain logits. We report the domain classification accuracy for different runs with different unseen domains, the average domain classification accuracy (Avg. Domain) and the average image classification accuracy (Avg. Class).

Method	XE	Art	Cartoon	Photo	Sketch	Avg. Domain	Avg. Class
BNE	✗	**71.2**	**82.2**	**75.0**	**60.1**	**72.1**	**83.1**
DNet	✗	33.3	33.3	33.3	33.3	33.3	79.1
BNE	✓	75.9	82.7	74.5	58.7	73.0	**80.8**
DNet	✓	**96.0**	**87.8**	**62.7**	**84.3**	**82.7**	74.9

In Sec 4.3, we compared the performance of BNE with DNet. For this purpose, we follow Mancini et al. (2018b) and implement a domain discovery network (DNet) that takes as input the activations after the first convolutional block and directly outputs the probability for the input sample to belong to each one of the training domains. This probability distribution is used to weigh the domain-specific predictions of our lightweight ensemble. Analogously to Mancini et al. (2018b) we implemented DNet as a lateral branch to our lightweight ensemble that is composed of a global pooling layer, followed by a ReLU non linearity, a fully-connected layer and a softmax activation. DNet is trained in an end-to-end fashion together with the main classifier. We considered two options: (i) training DNet applying only a cross-entropy loss on the image classification logits with respect to the input categories; (ii) training DNet directly supervising the classification of samples in the correct domain using domain labels.

In Tab. 9 we compare the domain classification accuracy (Avg. Domain) of BNE and that of DNet with or without applying a cross entropy loss over the domain labels across four tests considering different unseen domains on PACS. When cross-entropy is not applied on domain logits, BNE largely outperforms DNet. The 33% Avg. Domainfor DNet without cross-entropy on domain logits denotes that the discovery network learns to disregard the multidomain BN layer, always predicting the same domain class and thus leveraging only one branch of the multidomain BN layer. However, when cross-entropy is applied also on domain logits, the domain classification branch of DNet can adapt its parameters to predict well the domain classes (Avg. Domain). This, however, comes at the cost of a remarkable drop in image classification accuracy (Avg. Class) that can may be partially explained by DNet overfitting more to the training data and being less able to generalize to the test one. BNE instead provides a meaningful domain representation even without cross-entropy on domain logits, largely outperforming the image classification accuracy of DNet. Nevertheless, since our representation is not parametric, we cannot witness a visible increase in Avg. Domainwhen applying a cross-entropy loss also on the domain membership assigned through our representation.

The advantages of BNE over DNet are clear, since BNE leverages all the activations throughout the network to get an estimate of the domain membership, while DNet must rely only on the activations of the first layer due to the fixed input size of the domain classification branch. Moreover, our method allows a parameter-free domain representation, while DNet relies on a lateral branch to the main network. Finally, BNE allows to map samples in a latent space where distances from domain embeddings are computed, while DNet can only output the distance of the input sample from the training domains.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. 2015.

Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. In *Advances in Neural Information Processing Systems*, pp. 998–1008, 2018.

Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. In *Advances in Neural Information Processing Systems*, pp. 7694–7705, 2018.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *Advances in neural information processing systems*, pp. 343–351, 2016.

Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2229–2238, 2019.

Fabio Maria Carlucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulo. Autodial: Automatic domain alignment layers. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5077–5085. IEEE, 2017a.

Fabio Maria Carlucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulo. Just dial: Domain alignment layers for unsupervised domain adaptation. In *International Conference on Image Analysis and Processing*, pp. 357–369. Springer, 2017b.

Zhengming Ding and Yun Fu. Deep domain generalization with structured low-rank constraint. *IEEE Transactions on Image Processing*, 27(1):304–313, 2017.

Qi Dou, Daniel Coelho de Castro, Konstantinos Kamnitsas, and Ben Glocker. Domain generalization via model-agnostic learning of semantic features. In *Advances in Neural Information Processing Systems*, pp. 6447–6458, 2019.

Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *Proceedings of the IEEE international conference on computer vision*, pp. 2551–2559, 2015.

Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2066–2073. IEEE, 2012.

Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

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*, pp. 770–778, 2016.

Shoubo Hu, Kun Zhang, Zhitang Chen, and Laiwan Chan. Domain generalization via multidomain discriminant analysis. In *Uncertainty in artificial intelligence: proceedings of the... conference. Conference on Uncertainty in Artificial Intelligence*, volume 35. NIH Public Access, 2019.

Aditya Khosla, Tinghui Zhou, Tomasz Malisiewicz, Alexei A Efros, and Antonio Torralba. Undoing the damage of dataset bias. In *European Conference on Computer Vision*, pp. 158–171. Springer, 2012.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pp. 5542–5550, 2017.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018a.

Da Li, Jianshu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy M Hospedales. Episodic training for domain generalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1446–1455, 2019.

Ya Li, Mingming Gong, Xinmei Tian, Tongliang Liu, and Dacheng Tao. Domain generalization via conditional invariant representations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018b.

Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep domain generalization via conditional invariant adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 624–639, 2018c.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Massimiliano Mancini, Samuel Rota Bulò, Barbara Caputo, and Elisa Ricci. Best sources forward: domain generalization through source-specific nets. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 1353–1357. IEEE, 2018a.

Massimiliano Mancini, Samuel Rota Bulo, Barbara Caputo, and Elisa Ricci. Robust place categorization with deep domain generalization. *IEEE Robotics and Automation Letters*, 3(3):2093–2100, 2018b.

Massimiliano Mancini, Lorenzo Porzi, Samuel Rota Bulò, Barbara Caputo, and Elisa Ricci. Boosting domain adaptation by discovering latent domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3771–3780, 2018c.

Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pp. 10–18, 2013.

Mohammad Mahfujur Rahman, Clinton Fookes, Mahsa Baktashmotlagh, and Sridha Sridharan. Multi-component image translation for deep domain generalization. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 579–588. IEEE, 2019.

Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pp. 213–226. Springer, 2010.

Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. *arXiv preprint arXiv:1804.10745*, 2018.

Marcel Simon, Erik Rodner, and Joachim Denzler. Imagenet pre-trained models with batch normalization. *arXiv preprint arXiv:1612.01452*, 2016.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.