Pattern Recognition 135 (2023) 109115

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Batch normalization embeddings for deep domain generalization

Mattia Segu[a,*], Alessio Tonioni[a], Federico Tombari[a]

Google, Brandschenkestrasse 110, Zurich, 8002, Switzerland

ARTICLE INFO

Article history:
Received 13 April 2021
Revised 17 August 2022
Accepted 16 October 2022
Available online 23 October 2022

Keywords:
Domain generalization
Domain representation learning
Learning from multiple sources

ABSTRACT

Domain generalization aims at training machine learning models to perform robustly across different and unseen domains. Several methods train models from multiple datasets to extract domain-invariant features, hoping to generalize to unseen domains. Instead, first we explicitly train domain-dependent representations leveraging ad-hoc batch normalization layers to collect independent domain's statistics. Then, we propose to use these statistics to map domains in a shared latent space, where membership to a domain is measured by means of a distance function. At test time, we project samples from an unknown domain into the same space and infer properties of their domain as a linear combination of the known ones. We apply the same mapping strategy at training and test time, learning both a latent representation and a powerful but lightweight ensemble model. We show a significant increase in classification accuracy over current state-of-the-art techniques on popular domain generalization benchmarks: PACS, Office-31 and Office-Caltech.

1. Introduction

Machine learning models trained on a certain data distribution often fail to generalize to samples from different distributions. This phenomenon is commonly referred to in literature as *domain shift between training and testing data* [1,2], and is one of the biggest limitations of data driven algorithms. Assuming the availability of few annotated samples from the test domain, the problem can be mitigated by fine-tuning the model with explicit supervision [3] or with domain adaptation techniques [4]. Unfortunately, this assumption does not always hold in practice as it is often unfeasible in real scenarios to collect samples for any possible environment.

Domain generalization refers to algorithms to solve the *domain shift* problem by learning models robust to unseen domains. Several works leverage different domains at training time to learn a domain-invariant feature extractor [5–9]. Other works focus on optimizing the model parameters to obtain consistent performance across domains via ad-hoc training policies [10–13], while a different line of work requires modifications to the model architecture to achieve domain invariance [14–17].

While these methods try to extract domain-invariant features, we go in the opposite direction and explicitly leverage domain-specific representations by collecting domain-dependent batch normalization (BN) statistics for each of the domains available at training time. By doing so, we train a lightweight ensemble of domain-specific models sharing all parameters except for BN statistics. Peculiarly to our proposal, we use the accumulated statistics to map each domain as a point in a *latent space of domains*. We will refer to this mapping as the Batch Normalization Embedding (BNE) of a domain. Fig. 1 sketches a visualization of such space for the case of three domains available at training time (*e.g.* Photo, Art Painting and Cartoon). At convergence, each training domain is mapped to a single point in the domain space. Then, at test time unseen samples from unknown domains can be mapped to the same space by means of their instance normalization statistics. By measuring the distances between the instance normalization statistics of the test sample (black dot) and the accumulated population statistics of each domain (colored dots), we can infer properties of the unknown test domain. Specifically, we leverage the reciprocal of such distances at test time to weigh the domain-specific predictions of our lightweight ensemble and accurately classify an unseen sample from an unknown domain. The same combination of domain-specific models can be used at training time on samples from the known domains to force the ensemble to learn a meaningful latent space and logits that can be linearly combined according to the proposed weighting strategy.

To sum up the contributions of our work: (i) we propose to accumulate domain-specific batch normalization statistics accumulated on convolutional layers to map image samples into a latent space where membership to a domain can be measured according to a distance from domain BNEs; (ii) we propose to use this con-

* Corresponding author.
E-mail address: segum@ethz.ch (M. Segu).

https://doi.org/10.1016/j.patcog.2022.109115
0031-3203/

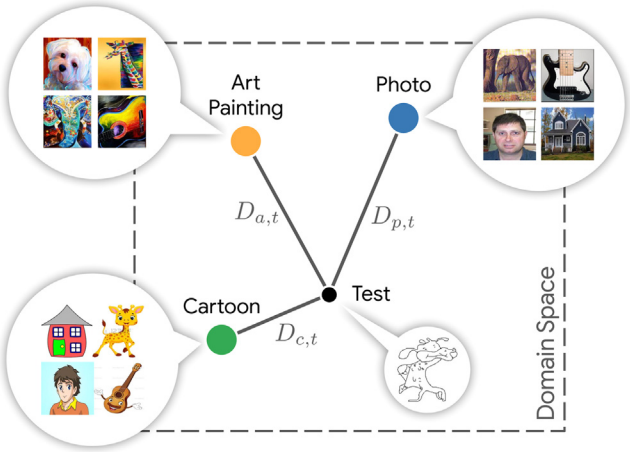


Fig. 1. Illustration of our method on the PACS dataset when the domains *Art Painting, Photo* and *Cartoon* are available at training time. Batch normalization layers implicitly learn a *domain space* onto which known (training) domains are mapped via their population statistics. Unknown test samples can be projected onto the same domain space via their instance statistics, and located with respect to the known domains via the corresponding distances $D_{a,t}$, $D_{p,t}$, and $D_{c,t}$.

cept to learn a lightweight ensemble model that shares all parameters excepts the normalization statistics and can generalize better to unseen domains; (iii) compared to previous work, we do not discard domain-specific attributes but exploit them to learn a domain latent space and map unknown domains with respect to known ones; (iv) our method can be applied to any modern Convolutional Neural Network (CNN) that relies on batch normalization layers, and scales gracefully to the number of domains available at training time.

2. Related work

Domain Generalization. Most domain generalization works attempt to expose the model to domain shift at training time to generalize to unseen domains. Invariance can be encouraged at multiple levels:

Feature-level, denotes methods deriving domain-invariant features by minimizing a discrepancy between multiple training domains. Ghifary *et al.* [6] brought domain generalization to the attention of the deep learning community by training multi-task autoencoders to transform images from one source domain into different ones, thereby learning invariant features. Analogously, Li *et al.* [9] extended adversarial autoencoders by minimizing the Maximum Mean Discrepancy measure to align the distributions of the source domains to an arbitrary prior distribution via adversarial feature learning. Conditional Invariant Adversarial Networks [18] have been proposed to learn domain-invariant representations, whereas Deep Separation Networks [19] extract image representations partitioned into two sub-spaces: one unique to each domain and one shared. Differently, Motiian *et al.* [8] propose to learn a discriminative embedding subspace via a Siamese architecture [7]. Episodic training [13] was proposed to train a generic model while exposing it to domain shift. In each episode, a feature extractor is trained with a badly tuned classifier (or vice-versa) to obtain robust features. Recently, [20] proposed a method to simultaneously discover latent domains by clustering features together and minimizing feature discrepancy between them. For all these methods, the limited variety of domains to which the model can be exposed at training time can limit the magnitude of the shift to which the model learns invariance.

Data-level, denotes methods attempting to reduce the domain shift by augmenting the cardinality and variety of the training samples. Methods based on domain-guided perturbations of input samples [11] or on adversarial examples [12] have been proposed with the purpose of training a model to be robust to distribution shift. Domain randomization was adopted [10,21] to solve the analogous problem of transferring a model from synthetic to real data by extending synthetic data with random renderings. By performing data augmentation those methods force the feature extractor to learn domain-invariant features, while we argue that discarding domain-specific information might be detrimental for performance.

Model-based, denotes methods relying on ad-hoc architectures to tackle the domain generalization problem. [15] introduced a low-rank parameterized CNN model, a dynamically parameterized neural network that generalizes the shallow binary undo bias method [14]. Mancini *et al.* [17] train multiple domain-specific classifiers and estimate the probabilities that a target sample belongs to each source domain to fuse the classifiers' predictions. A recent work [22] proposes an alternative approach to tackle domain generalization by teaching a model to simultaneously solve jigsaw puzzles and perform well on a task of interest. Most of these methods require changes to state-of-the-art architectures, resulting in an increased number of parameters or complexity of the network.

Meta-learning, denotes methods relying on special training policies to train models robust to domain shift. [23] extends to domain generalization the widely used model agnostic meta learning method [24]. A gradient-based meta-training procedure [25] was introduced to expose the optimization to domain shift while regularizing the semantic structure of the feature space. [26] proposes a training heuristic to iteratively discard the dominant features activated by training data and learn more general representations. These methods simulate unseen domains by splitting the training data in meta-training and meta-test sets, therefore are inherently bounded by the variety of samples available at training time.

Batch Normalization for distribution alignment. The use of separate batch normalization statistics to align a training distribution to a test one has been firstly introduced for domain adaptation [27,28]. The same domain-dependent batchnorm layer has been adapted to the multi-domain scenario [29,30] and exploited in a graph-based method [31] that leverages domain meta-data to better align unknown domains to the known ones. All these works, however, require some representation of the target domain to perform the alignment during training, using either samples or meta-data describing the target domain. Our approach instead does not rely on any external source of information regarding the target domain. Domain-specific normalization layers have only recently been proposed for domain generalization in [32], where a cluster of networks is trained to learn an optimal mixture of instance and batch normalization.

3. Method

The core idea of our method is to exploit domain-specific batch normalization statistics to map known and unknown domains in a shared latent space, where domain membership of samples can be measured according to their distance from the domain embeddings of the known domains.

3.1. Problem formulation

Let $\mathcal{X}$ and $\mathcal{Y}$ denote the input (*e.g.* images) and the output (*e.g.* object categories) spaces of a model. Let $\mathcal{D} = \{d_i\}_{i=1}^{K}$ denote the set of the K source domains available at training time. Each domain d_i can be described by an unknown conditional probability distribution $p^y_{x,d_i} = p(y|x,i)$ over the space $\mathcal{X} \times \mathcal{Y}$. The aim of a machine learning model is to learn the probability distribution $p^y_x = p(y|x)$

of the training set by training models to learn a mapping $\mathcal{X} \rightarrow \mathcal{Y}$. We propose to use a lightweight ensemble of models to learn a mapping $(\mathcal{X}, \mathcal{D}) \rightarrow \mathcal{Y}$ that leverages the domain label to model a set of conditional distributions $\{p^y_{x,d_i}\}_{i=1}^K$, each conditioned on the domain membership. Let t be a generic target domain available only at testing time and following the unknown probability distribution $p^y_{x,t}$ over the same space. Since it is not possible to learn the target distribution $p^y_{x,t}$ during training, our goal is to accurately estimate it as a mixture (*i.e.* linear combination) of the learned source distributions p^y_{x,d_i}.

For each source domain $d \in \mathcal{D}$, a training set $\mathcal{S}_d = \{(x_{1_d}, y_{1_d}), \ldots, (x_{n_d}, y_{n_d})\}$ containing n_d labelled samples is provided. The test set $\mathcal{T} = \{x_{1_t}, \ldots, x_{n_t}\}$ is composed of m_t unlabelled samples collected from the unknown marginal distribution p^x_t of the target domain t. As opposed to the domain adaptation setting, we assume that target samples are not available at training time, and that each of them might belong to a different unseen domain.

3.2. Multi-Source domain alignment layer

Neural networks are particularly prone to capture dataset bias in their internal representations [33], making internal features distributions highly domain-dependent. To capture and alleviate the distribution shift that is inherent in the multi-source setting, we draw inspiration from [27,29,30,32] and adopt batch normalization layers [34] to normalize the activations of each domain to the same reference distribution via *domain-specific normalization statistics*.

At inference time, the activations of a certain domain d are normalized by matching their first and second order moments, nominally (μ_d, σ^2_d), to those of a reference Gaussian with zero mean and unitary variance:

$$BN(z; d) = \frac{z - \mu_d}{\sqrt{\sigma^2_d + \epsilon}}, \tag{1}$$

where z is an input activation extracted from the marginal distribution q^z_d of the activations from the domain d; $\mu_d = \mathbb{E}_{z \sim q^z_d}[z]$ and $\sigma^2_d = Var_{z \sim q^z_d}[z]$ are the population statistics for the domain d, and $\epsilon > 0$ is a small constant to avoid numerical instability. At training time, the layer collects and applies domain-specific batch statistics $(\tilde{\mu}_d, \tilde{\sigma}^2_d)$, while updating the corresponding moving averages to approximate the domain population statistics.

At inference time, if the domain label d of a test sample is unknown or it does not belong to $\mathcal{D}$, we can still rely on normalization by *instance statistics, i.e.* the degenerate case of batch statistics with batch size equal to 1.

Fig. 2 (a) depicts the functioning of a multi-source domain generalization layer. Our method builds on the observation that for convolutional layers instance statistics and batch statistics are approximations of the same underlying distribution with different degrees of noise. Since the population statistics are a temporal integration of the batch statistics, the validity of this statement extends to the comparison with them. For example, statistics for a single channel in the case of a batch normalization layer applied on a 2D feature map of size $H \times W$, for a generic batch size B (*batch statistics*) and for $B = 1$ (*instance statistics*), are computed as:

$$\tilde{\mu} = \frac{1}{B \cdot H \cdot W} \sum_{b,h,w} z_{b,h,w} \overset{(B=1)}{=} \frac{1}{H \cdot W} \sum_{h,w} z_{h,w} \tag{2}$$

$$\tilde{\sigma}^2 = \frac{1}{B \cdot H \cdot W} \sum_{b,h,w} (z_{b,h,w} - \tilde{\mu})^2 \overset{(B=1)}{=} \frac{1}{H \cdot W} \sum_{h,w} (z_{h,w} - \tilde{\mu})^2, \tag{3}$$

where $\tilde{\mu}$ and $\tilde{\sigma}^2$ are respectively the batch mean and variance and $z_{b,h,w}$ is the value of a single element of the feature map. If we consider $z_{b,h,w}$ to be described by a normally distributed random variable $Z \sim \mathcal{N}(\mu, \sigma^2)$, then the instance and batch statistics are an estimate of the parameters of the same gaussian computed over a different number of samples, $H \cdot W$ and $B \cdot H \cdot W$ respectively. In the next section, we explain how we exploit this property to map source domains and unseen samples from unknown domain into the same latent space.

3.3. Domain localization in the batchnorm latent space

The domain alignment layers described in Sec. 3.2 allow to learn the multiple source distributions $\{p^y_{x,d}\}_{d \in \mathcal{D}}$ distinctly. By leveraging them, we can learn a lightweight ensemble of domain-specific models, where every network shares all the weights except for the normalization statistics. Since such a lightweight ensemble nicely embodies the multiple source distributions, we propose to reduce the domain shift on the unknown target domain by interpolating across these distributions to estimate the unknown distribution $p^y_{x,t}$. The resulting target distribution is a weighted mixture of the distributions in the ensemble, for which the weights choice depends on the *similarity* of a test sample to each source domain.

We denote with a $l \in \mathcal{B} = \{1, 2, \ldots, L\}$ in superscript notation the different batch normalization layers in the model. For each of them we can define a latent space $\mathcal{L}^l$ spanned by the activation statistics at the $l-th$ layer of the model. In this space, we observe that single samples x are mapped via their instance statics $(\tilde{\mu}, \tilde{\sigma}^2)$, whereas the population statistics accumulated for each domain (μ_d, σ^2_d) are used to represent domain centroids. Fig. 2-(b) shows a visualization of the latent space $\mathcal{L}^1$ for the PACS dataset [15], composed of 4 domains, and 3 of which (*e.g. Art Painting, Cartoon, Photo*) assumed available at training time. Population (big green dots) and instance (small dots) statistics are respectively used to project domains and individual samples. We rely on t-SNE to visualize instance and population statistics of source and target samples, and we observe how the latent space that we propose allows a spontaneous and stark division between domain clusters. Considering all latent spaces at different layers, we define a batch normalization embedding (BNE) for a certain domain d as the stacking of the population statistics computed at every layer:

$$e_d = [e^1_d, e^2_d, \ldots, e^L_d] = [(\mu^1_d, {\sigma^1_d}^2), (\mu^2_d, {\sigma^2_d}^2), \ldots, (\mu^L_d, {\sigma^L_d}^2)]. \tag{4}$$

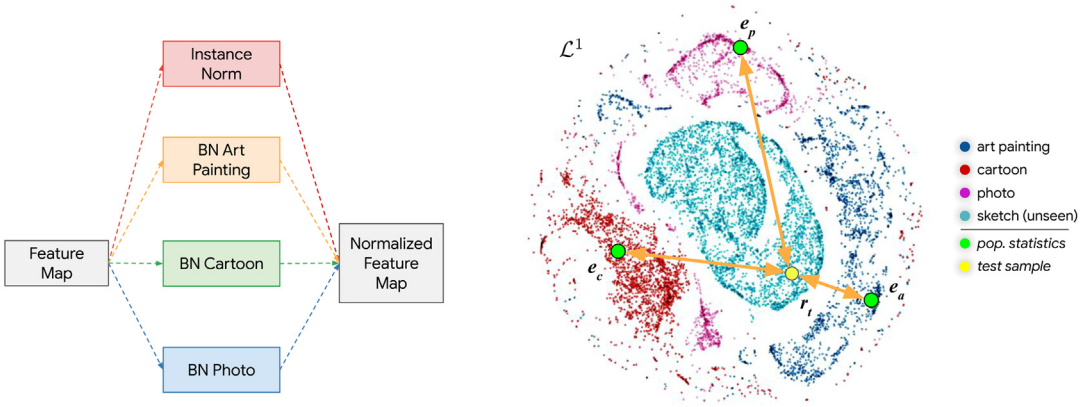
For a target sample x_t from an unknown domain t, we can derive a projection to the same space by forward propagating it through the network and computing its instance statistics. The latent embedding r_t of x_t is defined as the stacked vector of its instance statistics at different batch normalization layers in the network:

$$r_t = [r^1_t, r^2_t, \ldots, r^L_t] = [(\mu^1_t, {\sigma^1_t}^2), (\mu^2_t, {\sigma^2_t}^2), \ldots, (\mu^L_t, {\sigma^L_t}^2)]. \tag{5}$$

Each r^l_t represents the instance statistics collected at a certain layer l during forward propagation and can be used to map the sample x_t in the latent space $\mathcal{L}^l$ of layer l. Once the BNE for the test sample is available, it is possible to measure the *similarity* of a target sample x_t to one of the known domains d as the inverse of the distance between r_t and e_d. By extension, this allows a soft 1-Nearest Neighbour domain classification of any test sample.

To compute a distance between two points in $\mathcal{L}^l$, we consider the means and variances of the corresponding batch normalization layer as the parameters of a multivariate Gaussian distribution. We can hence adopt a distance on the space of probability measures, *i.e.* a symmetric and positive definite function that satisfies the triangle inequality. We select the *Wasserstein distance* for the special case of two multivariate gaussian distributions, but we report a comparison to alternative distances in Tab. 13. Let $p \sim \mathcal{N}(\mu_p, C_p)$ and $q \sim \mathcal{N}(\mu_q, C_q)$ be two normal distributions on R^n, with expected value μ_p and $\mu_q \in R^n$ respectively and C_p, $C_q \in R^{n \times n}$ covariance matrices. Denoting with $||\cdot||_2$ the Euclidean norm on R^n, the *2-Wasserstein distance* is:

$$\mathcal{W}(p, q) = \phi((\mu_p, C_p), (\mu_q, C_q)) \tag{6}$$



(a) Multi-Source Domain Alignment Layer. (b) t-SNE plot of instance and population statistics for seen and unseen domains.

Fig. 2. Our method on PACS ([15]) with *Sketch* unseen. A Multi-Source Domain Alignment Layer (a) collects domain-specific population statistics and compute instance statistics for test samples. (b) depicts the learned domain space $\mathcal{L}^1$ by means of a t-SNE plot of instance and population statistics for a BNE model. The Wasserstein distance (orange arrows) between a test embedding r_t and each domain embedding e_d weighs the predictions of domain-specific classifiers.

$$= ||\mu_p - \mu_q||_2^2 + Tr(C_p + C_q - 2(C_q^{\frac{1}{2}} C_p C_q^{\frac{1}{2}})^{\frac{1}{2}}),$$

Tr being the trace of the matrix. We rely on Eq. 6 to measure the distance between a test sample x_t and the domain d by summing over the batch normalization layers $l \in \mathcal{B}$ the distance between the activation embeddings r_t^l and e_d^l:

$$D_{\mathcal{L}}(e_d, r_t) = \sum_{l \in \mathcal{B}} \mathcal{W}(e_d^l, r_t^l) \tag{7}$$
$$= \sum_{l \in \mathcal{B}} \phi((\mu_d^l, Diag(\sigma_d^{l\,2})), (\mu_{x_t}^l, Diag(\sigma_{x_t}^{l\,2}))).$$

Eq. 2 and Eq. 3 show that instance and batch statistics differ only for the number of samples over which they are estimated, making the comparison meaningful. The *similarity* of a test sample x_t to the domain d is defined as the reciprocal of the distance from that domain and denoted as w_d^t. Once the similarities to each source domain are computed, we can use them to recover the unknown target distribution $p_{x,t}^y$ as a mixture (*i.e.* a linear combination) of the learned source distributions $p_{x,d}^y$ weighted by the corresponding domain similarity:

$$p_{x,t}^y = \frac{\sum_{d \in \mathcal{D}} w_d^t p_{x,d}^y}{\sum_{d \in \mathcal{D}} w_d^t}. \tag{8}$$

We denote with $f(\cdot)$ the result of a forward pass in a neural network. We get the final prediction of our lightweight ensemble model $f(x_t)$ as a linear combination of the domain dependant models $f(x_t|d)$:

$$f(x_t) = \frac{\sum_{d \in \mathcal{D}} w_d^t f(x_t|d)}{\sum_{d \in \mathcal{D}} w_d^t}, \tag{9}$$

Fig. 2-(b) shows a visualization of our localization technique superimposed over the t-SNE plot of instance and population statistics. By measuring the distance of a test samples (yellow dot) from the training domain embeddings (big green dots), we obtain ad-hoc mixture coefficients for each test sample.

Our formulation allows to navigate in the latent space of the batchnorm statistics. Specifically, if a test sample belongs to one of the source domains, our method assigns a high weight to the prediction of the corresponding domain-specific model. On the other hand, if the test sample does not belong to any of the source domains, the final prediction will be expressed as a linear combination of the domain-dependant models embodied in our lightweight ensemble.

3.4. Training policy

To encourage a well-defined latent space for every batch normalization layer, we replicate at training time the distance weighting procedure described in Eq. 9 to compute predictions on samples from known domains. Each training batch is composed of K domain batches with an equal number of samples. During every training step, (i) the domain batches are first propagated to update the corresponding domain population statistics $(\mu_d, \sigma_d^2,)$. Then, (ii) all individual samples are propagated assuming an unknown domain to collect their instance statistics and compute the domain similarities w_d^t, as in Sec. 3.3. Finally, (iii) each sample is propagated under K different domain assumptions (*i.e.* through the corresponding domain-specific branches) and the resulting domain-specific predictions are weighted according to Eq. 9. Applying this procedure during training promotes the creation of a well-defined latent space.

Since we initialize our model with weights pre-trained on ImageNet [35], each domain-specific batch normalization branch needs to be specialized before starting the distance training (DT) procedure described above, otherwise convergence problems might occur. We thus *warm-up* domain-specific batch normalization statistics by pre-training the model on the whole dataset following the standard procedure, except for the accumulation and application of domain-specific batch normalization statistics.

4. Experiments

4.1. Experimental settings

By means of a synecdoche, we name our method after BNE, its main component.

Datasets. We evaluate our method on three domain generalization benchmarks:

PACS [15] features 4 domains (*Art Painting, Cartoon, Photo, Sketch*) with a significant domain shift. Each domain includes samples from 7 different categories, for a total of 9991 samples. Some examples are shown in Fig. 2.

Office-31 [36] was originally introduced for domain adaptation and has been subsequently used for domain generalization. The dataset is composed of 3 different sources and 31 categories, representing images captured with a Webcam and a dSLR camera or collected from the Amazon website.

Office-Caltech [37] is a variant of Office-31 featuring one additional domain, derived from the Caltech-256 dataset [38]. The dataset is composed of the 10 categories shared between Caltech-256 and the domains in Office-31.

Training Settings. Coherently with other works, we evaluate both the AlexNet [39] and the more recent ResNet-18 [40] 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 [39], 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 [41] and weight decay 10^{-6}. The initial learning rate is 10^{-4}. Coherently with previous works ([27,29]), 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 256×256, 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 ([42]). We initialize both AlexNet and ResNet-18 using the publicly available Caffe weights pre-trained on ImageNet.[1]

Evaluation Protocol. Coherently with other works, we evaluate both the AlexNet [39] and the ResNet-18 [40] architectures. For the experiments on PACS and Office-31 we follow the standard *leave-one-domain-out* evaluation procedure, where the model is trained on all domains but one, and tested on the left-out one. For Office-Caltech we do the same but also test following a *leave-two-domain-out* procedure. Since the original version of AlexNet does not include batch normalization layers, we adopt a variant with batch normalization applied on the activations of each convolutional layer [43]. Since the goal of domain generalization is to leverage multiple sources to learn models that are robust on any target domain, the natural deep-learning baseline to compare against consists in training directly on the merged set of source domains. We will refer to it as (DeepAll). We compare our method against this strong baseline and several deep-learning based methods for domain generalization. Since different methods rely on different initializations of network weights, which result in different baselines, we compare with methods providing their own baseline and report for every competitor: the performance on each unseen domain, the average baseline performance (Avg. DA), the average performance of the method itself (Avg.) and the relative gain (Δ%).

4.2. Domain generalization for classification

We here compare BNE against several methods for object classification on commonly used benchmarks.

PACS. We first benchmark our method on the PACS dataset [15], which presents a challenging domain generalization setting for object recognition. Every test uses 3 domains as training set and one as unknown test set; for each of this leave-one-out configurations, we train a model from the same initialization for 60 epochs. We first test our method using the ResNet-18 architecture and report the results in Tab. 1. Overall, our proposal obtains the best absolute accuracy on 1 out of 4 target sets, with the second best average accuracy (Avg.) of 83.1% and a relative gain (Δ%) of +5.86 making it the second most effective algorithm on this dataset. Since all the networks are initialized with weights trained on ImageNet, they are implicitly biased towards the *Photo* domain, as testified by the higher accuracy on it when treated as test set. *Sketch*, instead, is arguably the more challenging domain, as testified by the lower accuracy achieved by all methods. It is in this scenario that our method is able to provide the bigger gain (+9.6% absolute gain in accuracy over our baseline). The best perfomance on this datasets is obtained by DSON [32], which proposes to learn an ad-hoc mixture of instance and batch normalization statics to improve generalization. This characteristic makes it a perfect candidate to be extended with the domain embedding strategy that we propose in Sec. 3.3. An in-depth discussion on this topic is provided in Sec. 4.3.5.

In Tab. 2, 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 (Δ%), 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 our method can boost the performance by a +13% absolute gain in accuracy over our baseline.

Office-31. We evaluate our method on Office-31 [36] and follow the *leave-one-domain-out* protocol. To compare with published results, we use AlexNet initialized with ImageNet weights and train it with our method for 100 epochs with learning rate 10^{-4}. Tab. 3 shows that our approach obtains the best absolute accuracy in two out of three test scenarios and a relative gain comparable or better than the alternatives. The *Amazon* target domain proves to be the most challenging setting, as the images are acquired in ideal conditions (*i.e.* white backgrounds, studio lighting...) that are fairly different from the ImageNet domain. In this challening generalization scenario, BNE absolute accuracy outperforms the baseline by an impressive +11.2% and +2.3% with respect to the closest competitor. In Tab. 4, we extend the comparison on Office-31 considering ResNet-18, 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.

Office-Caltech. We evaluate our method also on Office-Caltech [37] and follow the standard evaluation procedure for this dataset, enumerating cases with a single target domain, either *Amazon* or *Caltech*, and scenarios with pairs of target domains: *Dslr-Webcam* and *Amazon-Caltech*. We use AlexNet initialized with ImageNet weights to compare with published results, and train BNE for 100 epochs. Tab. 5 shows that our approach achieves the best average accuracy and the best gain with respect to the baseline. The performance boost delivered with our method is especially evident in the challenging scenario where 2 domains are treated as targets, *e.g.* +6.4% absolute accuracy on *Dslr-Webcam*. In Tab. 6 we show additional results for Office-Caltech using ResNet-18. The same good property observed using Alexnet is confirmed also when considering ResNet as architecture in Tab. 6, with a clear +5.5% gain over DeepAll.

4.3. Ablation study

We here provide additional ablation studies to better highlight the properties of our method. In Sec. 4.3.1, we measure the impact on performance of the different components of our method. We validate BNE against other popular normalization strategies in Sec. 4.3.2. In Sec. 4.3.3, we validate quantitatively our latent space

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

Table 1
Comparison on PACS with ResNet-18. Methods with * do not use domain labels.

Method	Art	Cartoon	Photo	Sketch	Avg. DA	Avg.	Δ%
CrossGrad - [11]	78.7	73.3	94.0	65.1	79.1	77.8	-1.64
MetaReg - [44]	79.9	75.1	95.2	69.5	79.9	81.7	+2.25
MLDG - [23]	79.5	77.3	94.3	71.5	79.1	80.7	+2.02
Epi-FCR - [13]	82.1	77.0	93.9	73.0	79.1	81.5	+3.03
JiGen* - [22]	79.4	75.3	**96.0**	71.4	79.1	80.5	+1.77
MASF - [25]	80.3	77.1	94.99	71.69	79.2	81.0	+1.01
D-SAM - [45]	77.3	72.4	95.3	77.8	79.5	80.7	+1.47
MMLD* - [20]	81.3	77.2	96.1	72.3	78.7	81.8	+3.93
DSON - [32]	**84.7**	77.6	95.9	**82.2**	78.9	**85.1**	**+7.85**
DeepAll	75.8	73	94.4	70.9	-	78.5	-
BNE (*Ours*)	78.8	**78.9**	94.8	79.7	78.5	83.1	+5.86

Table 2
Comparison on PACS with AlexNet.

Method	Art	Cartoon	Photo	Sketch	Avg. DA	Avg.	Δ%
DICA - [5]	64.6	64.5	91.8	51.1	68.7	68.0	-1.02
D-MTAE - [6]	60.3	58.7	91.1	47.9	68.7	64.5	-6.11
DSN - [19]	61.1	66.5	83.3	58.6	68.7	67.4	-1.89
TF-CNN - [15]	62.9	67.0	89.5	57.5	67.1	69.2	+3.13
CIDDG - [18]	62.7	69.7	78.7	64.5	71.7	68.9	-3.91
Fusion - [17]	64.1	66.8	90.2	60.1	67.1	70.3	+4.77
CrossGrad - [11]	64.1	66.8	90.2	60.1	68.7	70.3	+2.33
MetaReg - [44]	69.8	70.4	91.1	59.3	69.3	72.6	+4.76
MLDG - [23]	66.2	66.9	88.0	59.0	67.2	70.0	+4.17
Epi-FCD - [13]	64.7	72.3	86.1	65.0	68.7	72.0	+4.80
JiGen - [22]	67.6	71.7	89.0	65.2	71.5	73.4	+2.66
MASF - [25]	**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**

Table 3
Comparison on Office-31 with AlexNet.

Method	Amazon	Dslr	Webcam	Avg. DA	Avg.	Δ%
UB - [14]	42.4	98.5	93.4	74.2	78.1	+5.26
DSN - [19]	44.0	99.0	94.5	74.2	79.2	+6.74
MTAE - [6]	43.7	99.0	94.2	74.2	79.0	+6.47
DGLRC - [16]	45.4	**99.4**	**95.3**	74.2	80.0	+7.82
MCIT - [46]	51.7	97.9	94.0	74.2	81.2	**+9.43**
DeepAll	43.8	94.1	88.4	-	75.4	-
BNE (*Ours*)	**54.0**	**99.4**	92.3	75.4	**81.9**	+8.62

Table 4
Comparison on Office-31 with ResNet-18.

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

proposal. Further, we extensively compare in Sec. 4.3.4 the performance of BNE against the variant DNet that leverages a domain discovery network. Finally, in Sec. 4.3.5, we conduct a qualitative analysis to verify our choices in terms of batch sizes and the distance measures used in the latent space. The experiments are conducted on the PACS dataset [15].

4.3.1. Method components

To measure the impact on performance of the different components of our method, we run ablation experiments on the PACS dataset using the ResNet-18 backbone, and report the results in Tab. 7. We compare again with the DeepAll baseline and against a *DiscoveryNet* (DNet, row (d)), a variant of BNE inspired by [30]. While in BNE we propose to assign domain membership by looking at the distance between the latent batch normalization embeddings, with DNet domain membership is learned through a domain classification network. On row (a) we show the performance gain attributable to the usage of separate batchnorm statistics for each training domain, while using at inference time the projection and weighting strategy described in Sec. 3.3; row (b) extends the this approach by leveraging *distance weighting* (DT) *at training time*, as described in Sec. 3.4; finally, row (c) includes a warm-up phase in the initial training phase to help population statistics to converge to stable values before starting distance training. By comparing the

Table 5
Comparison on Office-Caltech with AlexNet.

Method	Amazon	Caltech	Dslr, Webcam	Amazon, Caltech	Avg. DA	Avg.	Δ%
UB - [14]	91.0	86.0	80.5	70.0	84.5	81.9	-3.08
DSN - [19]	-	-	85.8	81.2	84.5	-	-
MTAE - [6]	93.1	86.2	85.3	80.5	84.5	86.3	+2.13
DGLRC - [16]	**94.2**	**87.6**	86.3	82.2	84.5	87.6	+3.67
MDA - [47]	93.5	86.9	84.9	82.6	84.5	87.0	+2.96
CIDG - [48]	93.2	85.1	83.7	65.91	84.5	82.0	-2.96
MCIT - [46]	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 6
Comparison on Office-Caltech with ResNet-18.

Method	Amazon	Caltech	Dslr, Webcam	Amazon, Caltech	Avg. DA	Avg.	Δ%
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

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

Method	DT	Warm-up	Art	Cartoon	Photo	Sketch	Avg.	Δ%
DeepAll	-	-	75.8	73.0	94.4	70.9	78.5	-
(a) BNE	✗	✗	74.7	71.7	93.0	74.8	78.6	+0.03
(b) BNE	√	✗	**79.8**	76.0	92.5	72.5	80.2	+2.13
(c) BNE	√	√	78.8	**78.9**	**94.8**	**79.7**	**83.1**	**+5.86**
(d) DNet	√	√	77.3	73.8	94.2	71.2	79.1	+0.08

Table 8
Comparison of different variants of our method on PACS with Alexnet.

Method	DT	Warm-up	Art	Cartoon	Photo	Sketch	Avg.	Δ%
DeepAll	-	-	64.4	65.4	88.0	53.8	67.9	-
(a) BNE	✗	✗	64.4	65.9	**89.6**	54.2	68.5	+0.92
(b) BNE	√	✗	63.7	**67.9**	84.6	66.6	70.7	+4.12
(c) BNE	√	√	**66.7**	65.7	89.5	**66.8**	**72.2**	+6.33

Table 9
Comparison of different normalization strategies on PACS with ResNet-18.

Method	Art	Cartoon	Photo	Sketch	Average
(a) InstanceNorm	62.6	72.7	79.7	71.7	71.7
(b) BatchNorm	75.8	73.0	94.4	70.9	78.5
(c) Freeze BatchNorm	75.0	76.8	92.6	73.9	79.6
(d) BNE (*Ours*)	**78.8**	**78.9**	**94.8**	**79.7**	**83.1**

Table 10
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**

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. By comparing line (c) to (d), we can notice how our proposal is more effective than the variant DNet inspired by the domain mapping strategy from [30], while also requiring less parameters. More details on DNet and extensive comparisons are reported in Sec. 4.3.4.

We also conduct ablation experiments regarding the model components on the PACS dataset using the AlexNet architecture. We report the results in Tab. 8, comparing again all variants of our method 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.

4.3.2. Normalization strategies

BNE can be interpreted as a peculiar normalization technique to achieve better generalization. We hence provide a quantitative comparison of our method against other popular normalization strategies: (a) InstanceNorm [28]; (b) BatchNorm [34]; (c) Freeze BatchNorm [34] (*i.e.*, keeping the population statistics as the one computed after the imagenet pre-training); (d) BNE *(Ours)*.

Results for this comparison are shown in Tab. 9. Freezing batch normalization statistics (c) to those accumulated on ImageNet provides better generalization than fine-tuning population statistics on the training datasets (b), which might lead to overfitting and is equivalent to the baseline DeepAll. This is coherent with what highlighted in [32], Among the analysed normalization techniques, InstanceNorm (a) achieves the poorest results. Our proposal (d) instead, by combining instance and batch normalization properties in a principled way, achieves the best results. BNE outperforms by a large margin all other normalization strategies, both overall (+3.5% over Freeze BatchNorm) and on any specific domain.

4.3.3. 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 every domain embedding: if the closest domain matches the real one, then the latent space effectively represents membership to a certain domain.

In Tab. 10 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.

Furthermore, we investigate the relationship between measured distances and the prediction accuracy on the same ResNet-18 trained on PACS without DT. For each test sample we measure the prediction accuracy obtained using only the predictions from either the closest domain branch, the second closest or the third (i.e.,

Table 11
Analysis of the classification accuracy considering predictions from the closest, second-closest (second) and third-closest (third) domain branches to target sample. We also report the average accuracy (Avg.) over all leave-one-domain-out tests.

Method	Art Painting	Cartoon	Photo	Sketch	Avg.
Closest	**68.02**	**63.10**	**92.75**	**71.09**	**72.20**
Second	65.14	60.75	86.83	57.11	64.58
Third	47.95	60.66	67.31	57.90	58.08

the farthest away). We run the test for all 4 possible unseen domains following the leave-one-domain-out protocol and report in Tab. 11 the average accuracy. The results show a clear correlation between distances and accuracy, as trusting the "closest" domain branch clearly results in a higher accuracy than the others.

4.3.4. Domain discovery net

In Sec. 4.3.1, we compared the performance of BNE with DNet. For this purpose, we follow [30] 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 [30] 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 also the classification of samples in the correct domain using domain labels.

In Tab. 12 we compare the domain classification accuracy (Avg. Domain) of BNE and 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. Domain for 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), 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. Domain when applying a cross-entropy loss also on the domain membership assigned through our representation.

Table 12
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

Table 13
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**

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.

4.3.5. Choosing a distance metric

A crucial component of our method is the distance function, used to locate each test sample with respect to the known domains. As mentioned in Sec. 3.3, we picked the Wasserstein distance for this task after a detailed preliminary study among different options, reported in Tab. 13. We considered three different distances: using a fixed value for the distance (*Uniform*), equivalent to averaging predictions of domain specific branches; using the Bhattacharya distance; using the Wasserstein distance. The basic Uniform distance setting is similar to [32], with the main difference being the normalization layer used: a domain-specific batch-norm in our case, a learned mixture of instance and batch normalizations for [32]. Their approach thus also require to learn additional parameters. In our experiments, the Wasserstein distance proves to be a more principled choice, consistently delivering the best performance both on average and on any specific left-out domain. Measuring the similarity of all samples to one of the training domain by means of a distance function is always more effective than blindly averaging predictions, therefore the effectiveness of our method derives from the accurate sample-wise domain attribution rather than from the chosen normalization layer. This observation leaves open the opportunity of combining our distance weighting scheme with other recently proposed normalization methods, *e.g.* [32].

4.3.6. Influence of the backbone architecture

Our experimental results show that applying BNE to AlexNet and ResNet results in comparable relative improvements with respect to the baseline performance on a given dataset. Given the large difference in parameters between AlexNet (62M) and ResNet (11M) and the difference in design choices (AlexNet does not rely on residual connections), we expect our method to scale to CNNs that differ in scale and design. Moreover, AlexNet was originally proposed without batch normalization layers. We show that plugging in batch normalization layers allows to apply our method and leads to satisfying results. As a result, our method is applicable to any CNN, as long as the batch size is large enough.

5. Conclusions

Our method allows to navigate the latent space of batch normalization statistics, describing unknown domains as a combination of known ones. We rely on domain-specific normalization layers to disentangle independent representations for each training domain, and then use such embeddings to localize unseen samples from unknown domains. Our method outperforms

many alternatives on a number of domain generalization benchmarks ([15,36,37]), underlining the advantage of maintaining specific domain representations over forcing invariant representations. We believe that our work highlights interesting under-explored properties of batch normalization layers. Further, our formulation could be extended to domain adaptation by injecting unlabelled samples from the target domain during training. If few target samples are simultaneously available, they could be used to retrieve a less biased estimate of the unseen domain statistics.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] M. Sugiyama, A.J. Storkey, Mixture regression for covariate shift, in: Advances in Neural Information Processing Systems, 2007, pp. 1337–1344.
[2] Y. Luo, L. Zheng, T. Guan, J. Yu, Y. Yang, Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2507–2516.
[3] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks? in: Advances in neural information processing systems, 2014, pp. 3320–3328.
[4] M. Wang, W. Deng, Deep visual domain adaptation: a survey, Neurocomputing 312 (2018) 135–153.
[5] K. Muandet, D. Balduzzi, B. Schölkopf, Domain generalization via invariant feature representation, in: International Conference on Machine Learning, 2013, pp. 10–18.
[6] M. Ghifary, W. Bastiaan Kleijn, M. Zhang, D. Balduzzi, Domain generalization for object recognition with multi-task autoencoders, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 2551–2559.
[7] G. Koch, R. Zemel, R. Salakhutdinov, Siamese neural networks for one-shot image recognition, ICML deep learning workshop, volume 2, Lille, 2015.
[8] S. Motiian, M. Piccirilli, D.A. Adjeroh, G. Doretto, Unified deep supervised domain adaptation and generalization, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 5715–5725.
[9] H. Li, S. Jialin Pan, S. Wang, A.C. Kot, Domain generalization with adversarial feature learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 5400–5409.
[10] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, P. Abbeel, Domain randomization for transferring deep neural networks from simulation to the real world, in: 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS), IEEE, 2017, pp. 23–30.
[11] S. Shankar, V. Piratla, S. Chakrabarti, S. Chaudhuri, P. Jyothi, S. Sarawagi, Generalizing across domains via cross-gradient training, arXiv preprint arXiv:1804.10745 (2018).
[12] R. Volpi, H. Namkoong, O. Sener, J.C. Duchi, V. Murino, S. Savarese, Generalizing to unseen domains via adversarial data augmentation, in: Advances in Neural Information Processing Systems, 2018, pp. 5334–5344.
[13] D. Li, J. Zhang, Y. Yang, C. Liu, Y.-Z. Song, T.M. Hospedales, Episodic training for domain generalization, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 1446–1455.
[14] A. Khosla, T. Zhou, T. Malisiewicz, A.A. Efros, A. Torralba, Undoing the damage of dataset bias, in: European Conference on Computer Vision, Springer, 2012, pp. 158–171.
[15] D. Li, Y. Yang, Y.-Z. Song, T.M. Hospedales, Deeper, broader and artier domain generalization, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 5542–5550.
[16] Z. Ding, Y. Fu, Deep domain generalization with structured low-rank constraint, IEEE Trans. Image Process. 27 (1) (2017) 304–313.
[17] M. Mancini, S.R. Bulò, B. Caputo, E. Ricci, Best sources forward: domain generalization through source-specific nets, in: 2018 25th IEEE International Conference on Image Processing (ICIP), IEEE, 2018, pp. 1353–1357.
[18] Y. Li, X. Tian, M. Gong, Y. Liu, T. Liu, K. Zhang, D. Tao, Deep domain generalization via conditional invariant adversarial networks, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 624–639.
[19] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, D. Erhan, Domain separation networks, in: Advances in neural information processing systems, 2016, pp. 343–351.
[20] T. Matsuura, T. Harada, Domain generalization using a mixture of multiple latent domains, in: AAAI, 2020, pp. 11749–11756.
[21] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, D. Scaramuzza, Deep drone racing: from simulation to reality with domain randomization, IEEE Trans. Rob. (2019).
[22] F.M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, T. Tommasi, Domain generalization by solving jigsaw puzzles, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2229–2238.
[23] D. Li, Y. Yang, Y.-Z. Song, T.M. Hospedales, Learning to generalize: Meta-learning for domain generalization, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[24] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 1126–1135.
[25] Q. Dou, D.C. de Castro, K. Kamnitsas, B. Glocker, Domain generalization via model-agnostic learning of semantic features, in: Advances in Neural Information Processing Systems, 2019, pp. 6447–6458.
[26] Z. Huang, H. Wang, E.P. Xing, D. Huang, Self-challenging improves cross-domain generalization, arXiv preprint arXiv:2007.02454 (2020).
[27] F.M. Carlucci, L. Porzi, B. Caputo, E. Ricci, S.R. Bulo, Just dial: Domain alignment layers for unsupervised domain adaptation, in: International Conference on Image Analysis and Processing, Springer, 2017, pp. 357–369.
[28] Y. Li, N. Wang, J. Shi, X. Hou, J. Liu, Adaptive batch normalization for practical domain adaptation, Pattern Recognit 80 (2018) 109–117.
[29] M. Mancini, L. Porzi, S. Rota Bulò, B. Caputo, E. Ricci, Boosting domain adaptation by discovering latent domains, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3771–3780.
[30] M. Mancini, S.R. Bulo, B. Caputo, E. Ricci, Robust place categorization with deep domain generalization, IEEE Rob. Autom. Lett. 3 (3) (2018) 2093–2100.
[31] M. Mancini, S.R. Bulo, B. Caputo, E. Ricci, Adagraph: Unifying predictive and continuous domain adaptation through graphs, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 6568–6577.
[32] S. Seo, Y. Suh, D. Kim, J. Han, B. Han, Learning to optimize domain specific normalization for domain generalization, arXiv preprint arXiv:1907.04275 (2019).
[33] Y. Li, N. Wang, J. Shi, J. Liu, X. Hou, Revisiting batch normalization for practical domain adaptation, arXiv preprint arXiv:1603.04779 (2016).
[34] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, 2015, pp. 448–456.
[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.
[36] K. Saenko, B. Kulis, M. Fritz, T. Darrell, Adapting visual category models to new domains, in: European conference on computer vision, Springer, 2010, pp. 213–226.
[37] B. Gong, Y. Shi, F. Sha, K. Grauman, Geodesic flow kernel for unsupervised domain adaptation, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2012, pp. 2066–2073.
[38] G. Griffin, A. Holub, P. Perona, Caltech-256 object category dataset(2007).
[39] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.
[40] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[41] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).
[42] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, et al., Tensorflow: Large-scale machine learning on heterogeneous distributed systems (2015).
[43] M. Simon, E. Rodner, J. Denzler, Imagenet pre-trained models with batch normalization, arXiv preprint arXiv:1612.01452 (2016).
[44] Y. Balaji, S. Sankaranarayanan, R. Chellappa, Metareg: Towards domain generalization using meta-regularization, in: Advances in Neural Information Processing Systems, 2018, pp. 998–1008.
[45] A. D'Innocente, B. Caputo, Domain generalization with domain-specific aggregation modules, in: German Conference on Pattern Recognition, Springer, 2018, pp. 187–198.
[46] M.M. Rahman, C. Fookes, M. Baktashmotlagh, S. Sridharan, Multi-component image translation for deep domain generalization, in: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2019, pp. 579–588.
[47] S. Hu, K. Zhang, Z. Chen, L. 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.
[48] Y. Li, M. Gong, X. Tian, T. Liu, D. Tao, Domain generalization via conditional invariant representations, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

Mattia Segú received his MSc degree in Robotics, Systems and Control from ETH Zurich in 2021. In 2020 he was an intern at Google Zurich. He is now a PhD student in the Computer Vision Lab at ETH Zurich. His research interests include domain generalization, continual learning and uncertainty estimation.

Alessio Tonioni received his PhD degree in Computer Science and Engineering from University of Bologna in 2019. Currently, he is a research scientist at Google Zurich. His research interest concerns machine learning for depth estimation, domain adaptation and generalization. He has authored more than 15 papers on these subjects.

Federico Tombari is a Research Scientist and Manager at Google Zurich and a Lecturer at the Technical University of Munich. He has 200+ peer-reviewed publications in the field of 3D computer vision and machine learning. He was the recipient, among others, of two Google Faculty Research Awards and one Amazon Research Award.