Deeper, Broader and Artier Domain Generalization

Da Li Yongxin Yang Yi-Zhe Song Timothy M. Hospedales
Queen Mary University of London University of Edinburgh
{da.li, yongxin.yang, yizhe.song}@qmul.ac.uk, t.hospedales@ed.ac.uk

Abstract

The problem of domain generalization is to learn from multiple training domains, and extract a domain-agnostic model that can then be applied to an unseen domain. Domain generalization (DG) has a clear motivation in contexts where there are target domains with distinct characteristics, yet sparse data for training. For example recognition in sketch images, which are distinctly more abstract and rarer than photos. Nevertheless, DG methods have primarily been evaluated on photo-only benchmarks focusing on alleviating the dataset bias where both problems of domain distinctiveness and data sparsity can be minimal. We argue that these benchmarks are overly straightforward, and show that simple deep learning baselines perform surprisingly well on them.

In this paper, we make two main contributions: Firstly, we build upon the favorable domain shift-robust properties of deep learning methods, and develop a low-rank parameterized CNN model for end-to-end DG learning. Secondly, we develop a DG benchmark dataset covering photo, sketch, cartoon and painting domains. This is both more practically relevant, and harder (bigger domain shift) than existing benchmarks. The results show that our method outperforms existing DG alternatives, and our dataset provides a more significant DG challenge to drive future research.

1. Introduction

Learning models that can bridge train-test domain-shift is a topical issue in computer vision and beyond. In vision this has been motivated recently by the observation of significant bias across popular datasets [27], and the poor performance of state-of-the-art models when applied across datasets. Existing approaches can broadly be categorized into domain *adaptation* (DA) methods, that use (un)labeled target data to adapt source model(s) to a specific target domain [23]; and domain *generalization* (DG) approaches, that learn a domain agnostic model from multiple sources that can be applied to any target domain [12, 10]. While DA has been more commonly studied, DG is the more valuable yet challenging setting, as it does not require acquisition of a large target domain set for off-line analysis to drive adaptation. Such data may not even exist if the target domain is sparse. Instead it aims to produce a more human-like model, where there is a deeper semantic sharing across different domains – a dog is a dog no matter if it is depicted in the form of a photo, cartoon, painting, or indeed, a sketch.

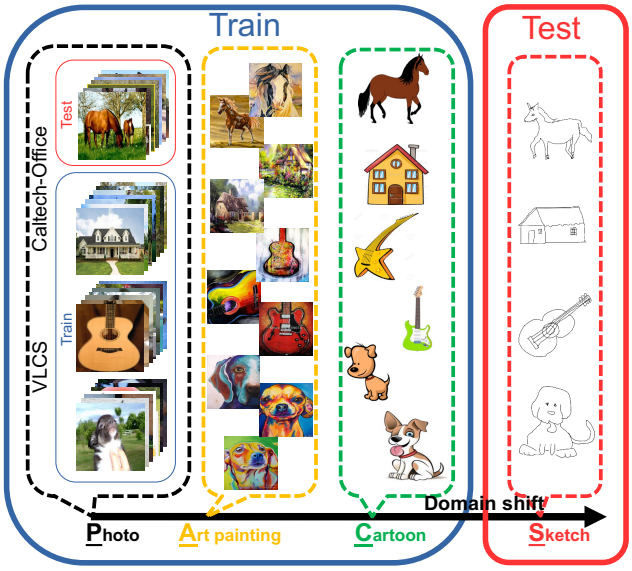


Figure 1: Contrast between prior Caltech Office and VLCS datasets versus our new PACS dataset. The domain generalization task is to recognize categories in an unseen testing domain. PACS provides more diverse domains with bigger more challenging domain-shifts between them.

The most popular existing DA/DG benchmarks define domains as photos of objects spanning different camera types [23], or datasets collected with different composition biases [27]. While these benchmarks provide a good start, we argue that they are neither well motivated nor hard enough to drive the field. *Motivation*: The constituent domains/datasets in existing benchmarks are based upon conventional photos, albeit with different camera types or composition bias. However there exist enough photos, that

one could in principle collect enough target domain-specific data to train a good model, or enough diverse data to cover all domains and minimize bias (thus negating the need for DA). A more compelling motivation is domains where the total available images is fundamentally constrained, such as for particular styles of art [5, 29], and sketches [33, 6, 34, 26]. Compared to photos, there may simply not be enough examples of a given art style to train a good model, even if we are willing to spend the effort. *Difficulty:* The camera type and bias differences between domains in existing benchmarks are already partially bridged by contemporary Deep features [4, 32], thus questioning the need for DA or DG methods. In this paper, we show that multi-domain deep learning provides a very simple but highly effective approach to DG that outperforms existing purpose-designed methods.

To address these limitations, we provide a harder and better motivated benchmark dataset PACS, consisting of images from photo (P), art painting (A), cartoon (C), and sketch (S) domains. This benchmark carries two important advancements over prior examples: (i) it extends the previously photo-only setting in DA/DG research, and uniquely includes domains that are maximally distinct from each other, spanning a wide spectrum of visual abstraction, from photos that are the least abstract to human sketches which are the most abstract; (ii) it is more reflective of a real-world task where a target domain (such as sketch) is intrinsically sparse, and so DG from a more abundant domain (such as photos) is really necessary. As illustrated qualitatively in Fig. 1, the benchmark is harder, as the domains are visually more distinct than in prior datasets. We explore these differences quantitatively in Sec. 4.2.

There have been a variety of prior approaches to DG based on SVM [12, 30], subspace learning [19], metric learning [7], and autoencoders [10]. Despite their differences, most of these have looked at fixed shallow features. In this paper, we address the question of how end-to-end learning of deep features impacts the DG setting. Our deep learning approach trains on multiple source domains, and extracts *both* domain agnostic features (e.g., convolutional kernels), and classifier (e.g., final FC layer) for transfer to a new target domain. This approach can be seen as a deep multi-class generalization of the shallow binary Undo Bias method [12], which takes the form of a dynamically parameterized deep neural network [25]. However, the resulting number of parameters grows linearly with the number of source domains (of which ultimately, we expect many for DG), increasing overfitting risk. To address this we develop a low-rank parameterized neural network which reduces the number of parameters. Furthermore the low-rank approach provides an additional route to knowledge sharing besides through explicit parameterization. In particular it has the further benefit of automatically modeling how related the different domains are (e.g., perhaps sketch is similar to cartoon; and cartoon is similar to painting), and also how the degree of sharing should vary at each layer of the CNN.

To summarize our contributions: Firstly, we highlight the weaknesses of existing methods (they lose to a simple deep learning baseline) and datasets (their domain shift is small). Second, we introduce a new, better motivated, and more challenging DG benchmark. Finally, we develop a novel DG method based on low-rank parameterized CNNs that shows favorable performance compared to prior work.

2. Related work

Domain Generalization Despite different methodological tools (SVM, subspace learning, autoencoders, etc), existing methods approach DG based on a few different intuitions. One is to project the data to a new domain invariant representation where the differences between training domains is minimized [19, 10], with the intuition that such a space will also be good for an unseen testing domain. Another intuition is to predict which known domain a testing sample seems most relevant to, and use that classifier [30]. Finally, there is the idea of generating a domain agnostic classifier, for example by asserting that each training domain's classifier is the sum of a domain-specific and domain-agnostic weight vector [12]. The resulting domain-agnostic weight vector can then be extracted and applied to held out domains. Our approach lies in this latter category. However, prior work in this area has dealt with shallow, linear models only. We show how to extend this intuition to end-to-end learning in CNNs, while limiting the resulting parameter growth, and making the sharing structure richer than an unweighted sum.

There has been more extensive work on CNN models for domain *adaptation*, with methods developed for encouraging CNN layers to learn transferable features [9, 17]. However, these studies have typically not addressed our domain *generalization* setting. Moreover, as analysis has shown that the transferability of different layers in CNNs varies significantly [32], these studies have had carefully hand designed the CNN sharing structure to address their particular DA problems. In our benchmark, this is harder, as the gaps between our more diverse domains are unknown and likely to be more variable. However, our low-rank modeling approach provides the benefit of automatically estimating both the per-domain and per-layer sharing strength.

Domain Generalization is also related to *learning to learn*. Learning to learn methods aim to learn not just specific concepts or skills, but learning algorithms or problem agnostic biases that improve generalization [20, 22, 8]. Similarly DG is to extract common knowledge from source domains that applies to unseen target domains. Thus our method can be seen as a simple learning to learn method for

the DG setting. Different from few-shot learning [8, 22], DG is a zero-shot problem as performance is immediately evaluated on the target domain with no further learning.

Neural Network Methods Our DG method is related to parameterized neural networks [1, 25], in that the parameters are set based on external metadata. In our case, based on a description of the current domain, rather than an instance [1], or additional sensor [25]. It is also related to low-rank neural network models, typically used to compress [13] and speed up [16] CNNs, and have very recently been explored for cross-category CNN knowledge transfer [31]. In our case we exploit this idea both for compression – but across rather than within domains [13], as well as for cross-domain (rather than cross-category [31]) knowledge sharing. Different domains can share parameters via common latent factors. [2] also addresses the DG setting, but learns shared parameters based on image reconstruction, whereas ours is learned via paramaterizing each domain's CNN. As a parameterized neural network, our approach also differs from all those other low-rank methods [13, 16, 31], which have a fixed parameterization.

2.1. Benchmarks and Datasets

DG Benchmarks The most popular DG benchmarks are: 'Office' [23] (containing Amazon/Webcam/DSLR images), later extended to include a fourth Caltech 101 domain [11] (OfficeCaltech) and Pascal 2007, LabelMe, Caltech, SUN09 (VLCS) [27, 12]. The domains within Office relate to different camera types, and the others are created by the biases of different data collection procedures [27]. Despite the famous analysis of dataset bias [27] that motivated the creation of the VLCS benchmark, it was later shown that the domain shift is much smaller with recent deep features [4]. Thus recent DG studies have used deep features [10], to obtain better results. Nevertheless, we show that a very simple baseline of fine-tuning deep features on multiple source domains performs comparably or better than prior DG methods. This motivates our design of a CNN-based DG method, as well as our new dataset (Fig 1) which has greater domain shift than the prior benchmarks. Our dataset draws on non-photorealistic and abstract visual domains which provide a better motivated example of the sort of relatively sparse data domain where DG would be of practical value.

Non-photorealistic Image Analysis Non-photorealistic image analysis is a growing subfield of computer vision that extends the conventional photo-only setting of vision research to include other visual depictions (often more abstract) such as paintings and sketches. Typical tasks include instance-level matching between sketch-photo [33, 24], and art-photo domains [3], and transferring of object recognizers trained on photos to detect objects in art [5, 29]. Most prior work focuses on two domains (such as photo and painting [5, 29], or photo and sketch [33, 24]). Studies have investigated simple 'blind' transfer between domains [5], learning cross-domain projections [33, 3], or engineering structured models for matching [29]. Thus, in contrast to our DG setting, prior non-photorealistic analyses fall into either cross-domain instance matching, or domain adaptation settings. To create our benchmark, we aggregate multiple domains including paintings, cartoons and sketches, and define a comprehensive domain-generalization benchmark covering a wide spectrum of visual abstraction based upon these. Thus in contrast to prior DG benchmarks, our domain-shifts are bigger and more challenging.

3. Methodology

Assume we observe S domains, and the ith domain contains N_i labeled instances $\{(x_j^{(i)}, y_j^{(i)})\}_{j=1}^{N_i}$ where $x_j^{(i)}$ is the input data (e.g., an image) for which we assume they are of the same size among all domains (e.g., all images are cropped into the same size), and $y_j^{(i)} \in \{1 \dots C\}$ is the class label. We assume the label space is consistent across domains. The objective of DG is to learn a domain agnostic model which can be applied to unseen domains in the future. In contrast to domain adaptation, we can not access the labeled or unlabeled examples from those domains to which the model is eventually applied. So the model is supposed to extract the domain agnostic knowledge within the observed domains. In the training stage, we will minimize the empirical error for all observed domains,

$$\underset{\Theta_1,\Theta_2,\dots,\Theta_S}{\operatorname{argmin}} \frac{1}{S}\sum_{i=1}^{S}\frac{1}{N_i}\sum_{j=1}^{N_i}\ell(\hat{y}_j^{(i)}, y_j^{(i)}) \tag{1}$$

where ℓ is the loss function that measures the error between the predicted label $\hat{y}$ and the true label y, and prediction is carried out by a function $\hat{y}_j^{(i)} = f(x_j^{(i)}|\Theta_i)$ parameterized by Θ_i. A straightforward approach to finding a domain agnostic model is to assume $\Theta_* = \Theta_1 = \Theta_2 = \cdots = \Theta_S$, i.e., there exists a universal model Θ_*. Doing so we literally ignore the domain difference. Alternatively, Undo-Bias [12] considers linear models, and assumes that the parameter (a D-dimensional vector when $x \in \mathbb{R}^D$) for the ith domain is in the form $\Theta^{(i)} = \Theta^{(0)} + \Delta^{(i)}$, where $\Theta^{(0)}$ can be seen as a domain agnostic model that benefits all domains, and $\Delta^{(i)}$ is a domain specific bias term. Conceptually, $\Theta^{(0)}$ can also serve as the classifier for any unseen domains. [12] showed that (for linear models) $\Theta^{(0)}$ is better than the universal model Θ_* trained by $\underset{\Theta_*}{\operatorname{argmin}} \frac{1}{S}\sum_{i=1}^{S}\frac{1}{N_i}\sum_{j=1}^{N_i}\ell(\Theta_*^T x_j^{(i)}, y_j^{(i)})$ in terms of testing performance on unseen domains. However we show that for deep networks, a universal model $f(x|\Theta_*)$ is a strong baseline that requires improved methodology to beat.

3.1. Parameterized Neural Network for DG

To extend the idea of Undo-Bias [12] into the neural network context, it is more convenient to think $\Theta^{(i)}$ is *generated* from a function $g(z^{(i)}|\Theta)$ parameterized by Θ. Here $z^{(i)}$ is a binary vector encoding of the ith domain with two properties: (i) it is of length $S+1$ where S is the number of observed domains; (ii) it always has only two units activated (being one): the ith unit active for the ith domain and the last unit active for all domains. Formally, the objective function becomes,

$$\underset{\Theta}{\operatorname{argmin}} \frac{1}{S}\sum_{i=1}^{S}\frac{1}{N_i}\sum_{j=1}^{N_i}\ell(\hat{y}_j^{(i)}, y_j^{(i)}) \tag{2}$$

where $\hat{y}_j^{(i)} = f(x_j^{(i)}|\Theta_i) = f(x_j^{(i)}|g(z^{(i)}|\Theta))$.

To reproduce Undo-Bias [12], we can stack all parameters in a column-wise fashion to form Θ, i.e., $\Theta = [\Delta^{(1)}, \Delta^{(2)}, \ldots, \Delta^{(S)}, \Theta^{(0)}]$, and choose the $g(\cdot)$ function to be linear mapping: $g(z^{(i)}|\Theta) = \Theta z^{(i)}$.

From linear to multi-linear The method as described so far generates the model parameter in the form of *vector* thus it is only suitable for single-out setting (univariate regression or binary classification). To generate higher order parameters, we use a multi-linear model, where Θ is (3rd order or higher) tensor. E.g., to generate a weighting matrix for a fully-connected layer in neural network, we can use

$$W_{\text{FC}}^{(i)} = g(z^{(i)}|\mathcal{W}) = \mathcal{W} \times_3 z^{(i)} \tag{3}$$

Here $\times_3$ is the inner product between tensor and vector along tensor's 3rd axis. For example if W is the weight matrix of size $H \times C$ (i.e., the number of input neurons is H and the number of output neurons is C) then $\mathcal{W}$ is a $H \times C \times (S+1)$ tensor.

If we need to generate the parameter for a convolutional layer of size $D_1 \times D_2 \times F_1 \times F_2$ (Height $\times$ Width $\times$ Depth $\times$ Filter Number), then we use:

$$\mathcal{W}_{\text{CONV}}^{(i)} = g(z^{(i)}|\mathcal{W}) = \mathcal{W} \times_5 z^{(i)} \tag{4}$$

where $\mathcal{W}$ is a 5th order tensor of size $D_1 \times D_2 \times F_1 \times F_2 \times (S+1)$.

Domain generalization Using one such parameter generating function per layer, we can dynamically generate the weights at *every* layer of a CNN based on the encoded vector of every domain. In this approach, knowledge sharing is realized through the last (bias) bit in the encoding of z. I.e., every weight tensor for a given domain is the sum of a domain specific tensor and a (shared) domain agnostic tensor. For generalization to an unseen domain, we apply the one-hot, bias-only, vector $z_* = [0, 0, \ldots, 0, 1]$ to synthesize a domain agnostic CNN.

3.2. Low rank parameterized CNNs

The method as described so far has two limitations: (i) the required parameters to learn now grow linearly in the number of domains (which we eventually hope to be large to achieve good DG), and (ii) the sharing structure is very prescribed: every parameter is an equally weighted sum of its domain agnostic and domain-specific bias partners.

To alleviate these two issues, we place a structural constraint on $\mathcal{W}$. Motivated by the well-known Tucker decomposition [28], we assume that the M-order tensor $\mathcal{W}$ is synthesized as:

$$\mathcal{W} = \mathcal{G} \times_1 U_1 \cdots \times_M U_M \tag{5}$$

where $\mathcal{G}$ is a $K_1 \times \ldots K_M$ sized low-rank core tensor, and U_m are $K_m \times D_m$ matrices (note that $D_M = S+1$). By controlling the ranks $K_1 \ldots K_M$ we can effectively reduce the number of parameters to learn. By learning $\{\mathcal{G}, U_1 \ldots U_M\}$ instead of $\mathcal{W}$, the number of parameters is reduced from $(D_1 \times \cdots \times D_{M-1} \times (S+1))$ to $(K_1 \times \ldots K_M) + \sum_{m=1}^{M-1} D_m \times K_m + K_M \times (S+1)$. Besides, U_M produces a K_M-dimensional dense vector that guides how to linearly combine the shared factors, which is much more informative than the original case of equally weighted sum.

Given a tensor $\mathcal{W}$ the Tucker problem can be solved via high-order singular value decomposition (HO-SVD) [15].

$$\mathcal{G} = \mathcal{W} \times_1 U_1^T \cdots \times_M U_M^T \tag{6}$$

where U_n is the U matrix from the SVD of the the mode-n flattening of $\mathcal{W}$. However, note that aside from (optionally) performing this once for initialization, we do *not* perform this costly HO-SVD operation during learning.

Inference and Learning To make predictions for a particular domain, we synthesize a concrete CNN by multiplying out the parameters $\{\mathcal{G}, U_1, \ldots, U_M\}$ after that doing an inner product with the corresponding domain's z. This CNN can then be used to classify an input instance x. Since our method does not introduce any non-differentiable functions, we can use standard back-propagation to learn $\{\mathcal{G}, U_1, \ldots, U_M\}$ for every layer.

For our model there are hyperparameters – Tucker rank $[K_1 \ldots K_M]$ – that can potentially be set at each layer. We sidestep the need to set all of these, by using the strategy of decomposing the stack of (ImageNet pre-trained) single domain models plus one agnostic domain model through Tucker decomposition, and then applying a reconstruction error threshold of $\epsilon = 10\%$ for the HO-SVD in Eq 6. This effectively determines all rank values via one 'sharing strength' hyperparameter ϵ.

4. Experiments

4.1. New Domain Generalization Dataset: PACS

Our PACS DG dataset is created by intersecting the classes found in Caltech256 (Photo), Sketchy (Photo, Sketch) [24], TU-Berlin (Sketch) [6] and Google Images (Art painting, Cartoon, Photo). Our dataset and code, together with latest results using alternative state-of-the-art base networks, can be found at: http://sketchx.eecs.qmul.ac.uk/.

PACS: Our new benchmark includes 4 domains (Photo, Sketch, Cartoon, Painting), and 7 common categories 'dog', 'elephant', 'giraffe'. 'guitar', 'horse', 'house', 'person'. The total number of images is 9991.

4.2. Characterizing Benchmarks' Domain Shifts

We first perform a preliminary analysis to contrast the domain shift within our PACS dataset to that of prior popular datasets such as VLCS. We make this contrast from both a feature space and a classifier performance perspective.

Feature Space Analysis Given the DG setting of training on source domains and applying to held out test domain(s), we measure the shift between source and target domains based on the Kullback-Leibler divergence as: $D_{shift}(D^s, D^t) = \frac{1}{m \times n} \sum_i^n \sum_j^m \lambda_i KLD(D_i^s || D_j^t)$, where n and m are the number of source and target domains, and λ_i weights the i th source domain, to account for data imbalance. To encode each domain as a probability, we calculate the mean DECAF_7 representation over instances and then apply softmax normalization.

Classifier Performance Analysis We also compare the datasets by the margin between multiclass classification accuracy of within-domain learning, and a simple cross-domain baseline of training a CNN on all the source domains before testing on the held out target domain (as we shall see later, this baseline is very competitive). Assuming within-domain learning performance is an upper bound, then this difference indicates the space which a DG method has to make a contribution, and hence roughly reflects size of the domain-shift/difficulty of the DG task.

Results Fig. 2(a) shows the average domain-shift in terms of KLD across all choices of held out domain in our new PACS benchmark, compared with the VLCS benchmark [27]. Clearly the domain shift is significantly higher in our new benchmark, as is visually intuitive from the illustrative examples in Fig. 1. To provide a qualitative summarization, we also show the distribution of features in our PACS compared to VLCS in Fig. 2(b,c) as visualized by a 2 dimensional t-SNE [18] plot, where the features are categorized and colored by their associated domain. From this result, we can see that the VLCS data are generally hard to separate by domain, while our PACS data are much more separated by domain. This illustrates the greater degree of shift between the domains in PACS over VLCS.

We next explore the domain shifts from a model-, rather than feature-centric perspective. Fig. 3a summarizes the within-domain and across-domain performance for each domain within PACS and VLCS benchmarks. The average drop in performance due to cross-domain transfer is 20.2% for PACS versus 10.0% for VLCS. This shows that the scope for contribution of DG/DA in our PACS is double that of VLCS, and illustrates the greater relevance and challenge of the PACS benchmark.

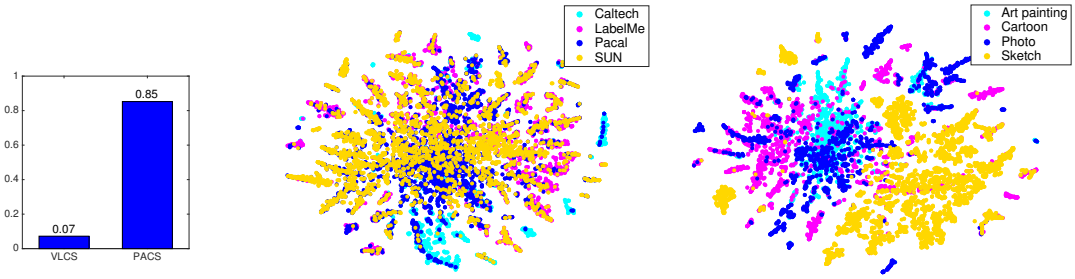
4.3. Domain Generalization Experiments

4.3.1 Datasets and Settings

We evaluate our proposed method on two datasets: VLCS, and our proposed PACS dataset. **VLCS** [27] aggregates photos from Caltech, LabelMe, Pascal VOC 2007 and SUN09. It provides a 5-way multiclass benchmark on the five common classes: 'bird','car','chair','dog' and 'person'. Our **PACS** (described in Sec. 4.1) with 7 classes from Photo, Sketch, Cartoon, Painting domains. All results are evaluated by multi-class accuracy, following [10]. We explore features including **Classic** SIFT features (for direct comparison with earlier work), **DECAF** pre-extracted deep features following [10], and **E2E** end-to-end CNN learning.

Settings: For our method in E2E configuration, we use the ImageNet pre-trained AlexNet CNN, fine-tuned with multi-domain learning on the training domains. On VLCS, we follow the train-test split strategy from [10]. Our initial learning rate is 5e-5 and batch size is 64 for each training domain. We use the best performed model on validation to do the test after tuning the model for $25k$ iterations. On PACS, we split the images from training domains to 9 (train) : 1 (val) and test on the whole held-out domain. Recall that our model uses a 2-hot encoding of z to parameterize the CNN. The domain-specific vs agnostic 'prior' can be set by varying the ratio ρ of the elements in the 2-hot coding. For training we use $\rho = 0.3$, so $z = \{[0, 0, 0.3, 1], [0, 0.3, 0, 1], ...\}$. For DG testing we use $z = [0, 0, 0, 1]$.

Baselines: We evaluate our contributions by comparison with number of alternatives including variants designed to reveal insights, and state of the art competitors:
Ours-MLP: Our DG method applied to a 1 hidden layer multi-layer perception. For use with pre-extracted features. **Ours-Full:** Our full low-rank parameterized CNN trained end-to-end on images. **SVM:** Linear SVM, applied on the aggregation of data from all source domains. **Deep-All:** Pretrained Alexnet CNN [14], fine-tuned on the aggregation of all source domains. **Undo-Bias:** Modifies traditional SVM to include a domain-specific and global weight vector which can be extracted for DG [12]. The original

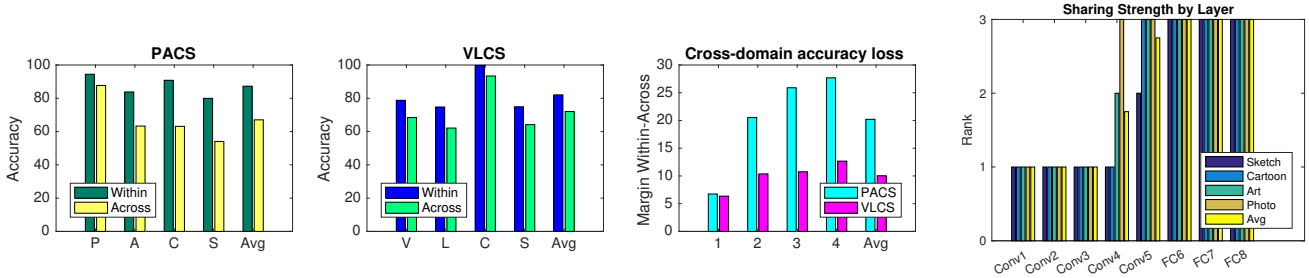


(a) Average KLD between domains.

(b) The feature distribution of VLCS

(c) The feature distribution of PACS

Figure 2: Evaluation of domain shift in different domain generalization benchmarks.



(a) Measuring domain-shift by within versus across domain accuracy. Left: Our PACS, Middle: VLCS. Right: Distribution of margins between within and across domain accuracy.

(b) Per-layer rank (inverse sharing strength) after learning for different held out domains.

Figure 3: Cross-domain similarity (a) and learned sharing strength by layer (b).

Undo-Bias is a binary classifier (BC). We also implement a multi-class (MC) generalization. **uDICA:** A kernel based method learning a subspace to minimize the dissimilarity between domains [19][1]. **UML**: Structural metric learning algorithm learn a low-bias distance metric for classification tasks [7]. **LRE-SVM:** Exploits latent domains, and a nuclear-norm based regularizer on the likelihood matrix of exemplar-SVM [30]. **1HNN**: 1 hidden layer neural network. **MTAE-1HNN**: 1HNN with multi-task auto encoder [10]. **D-MTAE-1HNN**: 1HNN with de-noising multi-task auto encoder [10]. **DSN**: The domain separation network learns specific and shared models for the source and target domains [2]. We re-purpose the original DSN from the domain adaptation to the DG task. Note that DSN is already shown to outperform the related [9].

4.3.2 VLCS Benchmark

Classic Benchmark - Binary Classification with Shallow Features Since our approach to extracting a domain invariant model is related to the intuition in Undo Bias [12], we first evaluate our methodology by performing a direct comparison against Undo Bias. We use the same 5376 dimensional VLCS SIFT-BOW features[2] from [12], and compare Our-MLP using one RELU hidden layer with 4096 neurons. For direct comparison, we apply Our-MLP in a 1-vs-All manner as per Undo-Bias. The results in Table 1 show that without exploiting the benefit of end-to-end learning, our approach still performs favorably compared to Undo Bias. This is due to (i) our low-rank modeling of domain-specific and domain-agnostic knowledge, and (ii) the generalization of doing so in a multi-layer network.

Multi-class recognition with Deep Learning In this experiment we continue to analyze the VLCS benchmark, but from a multiclass classification perspective. We compare existing DG methods (Undo-Bias [12], UML [7], LRE-SVM [30], uDICA [19], MTAE+1HNN [10], D-MTAE+1HNN [10]) against baselines (1HNN, SVM, Deep) and our methods Ours-MLP/Ours-Full. For the other methods besides Deep-All and Ours-Full, we follow [10] and use pre-extracted $DECAF_6$ features[3] [4]. For Deep and Ours-Full, we fine-tune the CNN on the source domains.

From the results in Table 2, we make the following ob-

[1] Like [10], we found sDICA to be worse than uDICA, so excluded it.

[2] http://undoingbias.csail.mit.edu/

[3] http://www.cs.dartmouth.edu/~chenfang/proj_page/FXR_iccv13/index.php

Unseen domain	Bird		Car		Chair		Dog		Person	
	Undo bias	Ours-MLP	Undo bias	Ours-MLP	Undo bias	Ours-MLP	Undo bias	Ours-MLP	Undo bias	Ours-MLP
Caltech	**12.08**	10.89	**63.80**	61.29	7.54	**11.26**	**5.24**	3.90	**50.81**	48.48
LabelMe	**33.08**	28.35	69.22	**74.07**	**5.34**	3.68	1.66	**2.06**	64.85	**67.00**
Pascal	**15.42**	13.63	37.49	**42.81**	30.05	**32.71**	14.97	**15.93**	58.47	**63.61**
Sun	0.59	**2.01**	70.62	**71.32**	37.44	**37.50**	1.12	**1.89**	42.20	**42.71**
Mean AP %	**15.29**	13.72	60.28	**62.37**	20.09	**21.29**	5.75	**5.94**	54.08	**55.45**

Table 1: Comparison against Undo-Bias [12] on the VLCS benchmark using classic SIFT-BOW features, and our shallow model Ours-MLP. Average precision (%) and mean average precision (%) of binary 1-v-all classification in unseen domains.

Unseen domain	Image ↦ Deep Feature ↦ Classifier									Image ↦ E2E	
	SVM	1HNN	Undo-Bias[12]	uDICA[19]	UML[7]	LRE-SVM[30]	MTAE+1HNN[10]	D-MTAE+1HNN[10]	Ours-MLP	Deep-All	Ours-Full
Caltech	77.67	86.67	87.50	61.70	91.13	88.11	90.71	89.05	92.43	93.40	**93.63**
LabelMe	52.49	58.20	58.09	46.67	58.50	59.74	59.24	60.13	58.74	62.11	**63.49**
Pascal	58.86	59.10	54.29	44.41	56.26	60.58	61.09	63.90	65.58	68.41	**69.99**
Sun	49.09	57.86	54.21	38.56	58.49	54.88	60.20	61.33	61.85	**64.16**	61.32
Ave.%	59.93	65.46	63.52	47.83	65.85	65.83	67.81	68.60	69.65	72.02	**72.11**

Table 2: Comparison of features and state of the art on the VLCS benchmark. Multi-class accuracy (%).

servations: (i) Given the fixed DECAF_6 feature, most prior DG methods improve on vanilla SVM, and D-MTAE [10] is the best of these. (ii) Ours-MLP outperforms 1HNN, which uses the same type of architecture and the same feature. This margin is due to our low-rank domain-generalization approach. (iii) The very simple baseline of fine-tuning a deep model on the aggregation of source domains (Deep-All) performs surprisingly well and actually outperforms all the prior DG methods. (iii) Ours-Full outperforms Deep-All slightly. This small margin is understandable. Our model does have more parameters to learn than Deep-All, despite the low rank; and the cost of doing this is not justified by the relatively small domain gap between the VLCS datasets.

4.3.3 Our PACS benchmark

We compare baselines (SVM, 1HNN) and prior methods (LRE-SVM [30], D-MTAE+1HNN [10], uDICA [19]) using DECAF_7 features against Deep-ALL, DSN [2] and Ours-Full using end-to-end learning. From the results in Table 3 we make the observations: (i) uDICA and D-MTAE-1HNN are the best prior DG models, and DSN is also effective despite being designed for DA. While uDICA scores well overall, this is mostly due to very high performance on the photo domain. This is understandable as in that condition DICA uses unaltered DECAF_7 features tuned for photo recognition. It is also the least useful direction for DG, as photos are already abundant. (ii) As for the VLCS benchmark, Deep-ALL again performs well. (iii) However Ours-Full performs best overall by combining the robustness of a CNN architecture with an explicit DG mechanism.

Ablation Study: To investigate the contributions of each components in our framework, we compare the following variants: *Tuning-Last*: Trains on all sources followed by direct application to the target. But fine-tunes the final FC layer only. *2HE-Last*: Fine-tunes the final FC layer, and uses our tensor weight generation (Eq. 3) based on 2-hot encoding for multidomain learning, before transferring the shared model component to the target. But without low rank factorisation. *2HE+Decomp-Last*: Uses 2-hot encoding based weight synthesis, and low-rank decomposition of the final layer (Eq. 3). *Ours-Full*: Uses 2-hot encoding and low-rank modeling on every layer in the CNN.

From the results, we can see that each component helps: (i) 2HE-Last outperforms Tuning-Last, demonstrating the ability of our tensor weight generator to synthesize domain agnostic models for a multiclass classifier. (ii) 2HE+Decomp-Last outperforms 2HE-Last, demonstrating the value of our low-rank tensor modeling of the weight generator parameters. (iii) Ours-Full outperforms 2HE+Decomp-Last, demonstrating the value of performing these DG strategies at every layer of the network.

4.4. Further Analysis

Learned Layer-wise Sharing Strength An interesting property of our approach is that, unlike some other deep learning methods [9, 17] it does not require manual specification of the cross-domain sharing structure at each layer of the CNN; and unlike Undo Bias [12] it can choose how to share more flexibly through the rank choice at each layer. We can observe the estimated sharing structure at each layer by performing Tucker decomposition to factorize the tuned model under a specified reconstruction error threshold ($\epsilon = 0.001$). The resulting domain-rank at each layer reveals the sharing strength. The rank per-layer for each held-out domain in PACS is shown in Fig. 3b. Here there are three training domains, so the maximum rank is 3 and the minimum rank is 1. Intuitively, the results show heavily shared Conv1-Conv3 layers, and low-sharing in FC6-FC8

Figure 4: Visualization of the preferred images of output neurons 'horse', 'giraffe' and 'house' in the domains of the PACS dataset. Left: real images. Middle: synthesized images for PACS domains. Right: synthesized images for agnostic domain.

Unseen domain	Image ↦ Deep Feature ↦ Classifier						Image ↦ E2E		
	SVM	1HNN	uDICA [19]	LRE-SVM [30]	D-MTAE+1HNN [10]	Ours-MLP	Deep-All	DSN [2]	Ours-Full
Art painting	55.39	59.10	**64.57**	59.74	60.27	61.40	63.30	61.13	62.86
Cartoon	52.86	57.89	64.54	52.81	58.65	57.16	63.13	66.54	**66.97**
Photo	82.83	89.86	**91.78**	85.53	91.12	89.68	87.70	83.25	89.50
Sketch	43.89	50.31	51.12	37.89	47.86	50.38	54.07	**58.58**	57.51
Ave.%	58.74	64.29	68.00	58.99	64.48	64.65	67.05	67.37	**69.21**

Table 3: Evaluation % of classification on PACS. Multi-class accuracy (%).

Unseen domain	Ablation Study			
	Tuning-Last	2HE-Last	2HE+Decom-Last	Ours-Full
Art painting	59.79	59.20	62.71	62.86
Cartoon	56.22	55.50	52.69	66.97
Photo	86.79	87.33	88.84	89.50
Sketch	46.41	48.45	52.16	57.51
Ave.%	62.30	62.62	64.10	69.21

Table 4: Ablation study. Multi-class accuracy (%).

layers. The middle layers Conv4 and Conv5 have different sharing strength according to which domains provide the source set. For example, in Conv 5, when Sketch is unseen, the other domains are relatively similar so can have greater sharing, compared to when Sketch is included as a seen domain. This is intuitive as Sketch is the most different from the other three domains. This flexible ability to determine sharing strength is a key property of our model.

Visualization To visualize the preferences of our multi-domain network, we apply the DGN-AM [21] method to synthesize the preferred input images for our model when parameterized (via the domain descriptor z) to one specific domain versus the abstract domain-agnostic factor. This visualization is imperfect because [21] is trained using a photo-domain, and most of our domains are non-photographic art. Nevertheless, from Fig. 4 the synthesis for Photo domain seem to be the most concrete, while the Sketch/Cartoon/Painting domains are more abstract.

5. Conclusion

We presented a new dataset and deep learning-based method for domain generalization. Our PACS (Photo-Art-Cartoon-Sketch) dataset is aligned with a practical application of domain generalization, and we showed it has more challenging domain shift than prior datasets, making it suitable to drive the field in future. Our new domain generalization method integrates the idea of learning a domain-agnostic classifier with a robust deep learning approach for end-to-end learning of domain generalization. The result performs comparably or better than prior approaches.

References

[1] L. Bertinetto, J. F. Henriques, J. Valmadre, P. H. S. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In *NIPS*, 2016. 3

[2] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. In *NIPS*, 2016. 3, 6, 7, 8

[3] E. J. Crowley, O. M. Parkhi, and A. Zisserman. Face painting: querying art with photos. In *BMVC*, 2015. 3

[4] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014. 2, 3, 6

[5] A. Z. E. J. Crowley. The art of detection. In *ECCV Workshop on Computer Vision for Art Analysis*, 2016. 2, 3

[6] M. Eitz, J. Hays, and M. Alexa. How do humans sketch objects? *TOG*, 2012. 2, 5

[7] C. Fang, Y. Xu, and D. N. Rockmore. Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In *ICCV*, 2013. 2, 6, 7

[8] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. 2, 3

[9] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015. 2, 6, 7

[10] M. Ghifary, W. Bastiaan Kleijn, M. Zhang, and D. Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *ICCV*, 2015. 1, 2, 3, 5, 6, 7, 8

[11] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, 2012. 3

[12] A. Khosla, T. Zhou, T. Malisiewicz, A. Efros, and A. Torralba. Undoing the damage of dataset bias. In *ECCV*, 2012. 1, 2, 3, 4, 5, 6, 7

[13] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. In *ICLR*, 2016. 3

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 5

[15] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIMAX*, 2000. 4

[16] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. In *ICLR*, 2015. 3

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

[18] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *JMLR*, 2008. 5

[19] K. Muandet, D. Balduzzi, and B. Scholkopf. Domain generalization via invariant feature representation. In *ICML*, 2013. 2, 6, 7, 8

[20] T. Munkhdalai and H. Yu. Meta networks. In *ICML*, 2017. 2

[21] A. M. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *NIPS*, 2016. 8

[22] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017. 2, 3

[23] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *ECCV*, 2010. 1, 3

[24] P. Sangkloy, N. Burnell, C. Ham, and J. Hays. The sketchy database: learning to retrieve badly drawn bunnies. *TOG*, 2016. 3, 5

[25] O. Sigaud, C. Masson, D. Filliat, and F. Stulp. Gated networks: an inventory. *arXiv*, 2015. 2, 3

[26] J. Song, Y. Qian, Y.-Z. Song, T. Xiang, and T. Hospedales. Deep spatial-semantic attention for fine-grained sketch-based image retrieval. In *ICCV*, 2017. 2

[27] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *CVPR*, 2011. 1, 3, 5

[28] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 1966. 4

[29] Q. Wu, H. Cai, and P. Hall. Learning graphs to model visual objects across different depictive styles. In *ECCV*. 2014. 2, 3

[30] Z. Xu, W. Li, L. Niu, and D. Xu. Exploiting low-rank structure from latent domains for domain generalization. In *ECCV*, 2014. 2, 6, 7, 8

[31] Y. Yang and T. M. Hospedales. Deep multi-task representation learning: A tensor factorisation approach. In *ICLR*, 2017. 3

[32] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014. 2

[33] Q. Yu, F. Liu, Y.-Z. Song, T. Xiang, T. M. Hospedales, and C. C. Loy. Sketch me that shoe. In *CVPR*, 2016. 2, 3

[34] Q. Yu, Y. Yang, F. Liu, Y.-Z. Song, T. Xiang, and T. M. Hospedales. Sketch-a-net: A deep neural network that beats humans. In *IJCV*, 2017. 2