Unsupervised Domain Generalization by Learning a Bridge Across Domains

Sivan Harary*, Eli Schwartz*, Assaf Arbelle*, Peter Staar1, Shady Abu-Hussein1,2, Elad Amrani1,3, Roei Herzig1,2, Amit Alfassy1,3, Raja Giryes2, Hilde Kuehne6,7, Dina Katabi5, Kate Saenko4,7, Rogerio Feris7, Leonid Karlinsky*

1IBM Research, 2Tel-Aviv University, 3Technion, 4Boston University, 5MIT, 6Goethe University, 7MIT-IBM Watson AI Lab

Abstract

The ability to generalize learned representations across significantly different visual domains, such as between real photos, clipart, paintings, and sketches, is a fundamental capacity of the human visual system. In this paper, different from most cross-domain works that utilize some (or full) source domain supervision, we approach a relatively new and very practical Unsupervised Domain Generalization (UDG) setup of having no training supervision in neither source nor target domains. Our approach is based on self-supervised learning of a Bridge Across Domains (BrAD) - an auxiliary bridge domain accompanied by a set of semantics preserving visual (image-to-image) mappings to BrAD from each of the training domains. The BrAD and mappings to it are learned jointly (end-to-end) with a contrastive self-supervised representation model that semantically aligns each of the domains to its BrAD-projection, and hence implicitly drives all the domains (seen or unseen) to semantically align to each other. In this work, we show how using an edge-regularized BrAD our approach achieves significant gains across multiple benchmarks and a range of tasks, including UDG, Few-shot UDA, and unsupervised generalization across multi-domain datasets (including generalization to unseen domains and classes).

1. Introduction

When observing some technical manual with schematic drawings of equipment for the first time, people do not need any supervision to associate these schematic drawings to real complex objects. This demonstrates the importance and efficiency of one of the basic human capacities - ability to learn with little to no supervision across multiple visual domains, as well as to effectively generalize to new domains and even new object classes without additional supervision.

Figure 1. Originally, same domain instances are closer to each other than to instances of the same class in other domains. Naive application of popular self-supervised learning techniques tends to separate domains before classes (Section 4). Our approach is to learn a BrAD - an auxiliary bridge domain (e.g. of edge-like images) that helps aligning instances of the same class across domains. Arrows indicate forces in feature space applied by our training losses (Section 3). green = attraction; red = repulsion.

Recent literature, extensively discussed in Section 2, is rich in works on learning to semantically generalize across domains without supervision in the target domain(s). The target domains can be observed as a collection of unlabeled images in Unsupervised Domain Adaptation (UDA), or even completely unseen during training in Domain Generalization (DG). For both UDA and DG, success in generalization would mean that the desired downstream tasks (classification, detection, etc.) would successfully transfer to a new seen or unseen domain. However, in most UDA and DG works an abundant source domain supervision for
the intended downstream task is assumed. But do we always have that in real-life use-cases? In many situations, such as in the technical manuals example above, we need our systems to generalize not only to new domains, but also to completely new kinds of objects for which we might have very little data (images and/or labels) - not sufficient to train the standard UDA or DG methods. Few recent Unsupervised DG (UDG) and Few-Shot UDA (FUDA) works, realized the importance of this issue, and restrict the source domain(s) supervision to few or even zero labeled examples. In this paper we target the least restrictive (in terms of labeling requirements) UDG setting, where we do not require any source domain supervision at training, and which also implicitly supports generalization to completely new unseen visual domains with new unseen classes.

We all draw an edge-like image when asked to draw something quickly. Object edges seem to be our shared universal visual representation for all the domains we observe. This drove the basic intuition behind our Bridge Across Domains approach: teach the machine to represent all the visual domains (in feature space) in the same way as it represents this seemingly universally shared visual domain of edge-like images. Our approach (Figure 1) is based on our proposed concept of a learnable Bridge Across Domains (BrAD) - an auxiliary visual ‘bridge’ domain to which it is relatively easy to visually map (in an image-to-image sense) all the domains of interest. The BrAD is used only during contrastive self-supervised training of our model for semantically aligning the representations (features) of each of the training domains to the ones for the shared BrAD. By transitivity of this semantic alignment in feature space, the learned model representations for all the domains are all BrAD-aligned and hence implicitly aligned to each other. This makes the learned mappings to the BrAD unnecessary for inference, making the trained model generalizable even to new unseen domains (for which we do not have BrAD mappings) at test time. Moreover, not depending on BrAD-mapping for inference allows exploiting additional non-BrAD-specific features (e.g. color) which are also learned by our representation model encoder (Sec. 3).

We show that even a simple heuristic implementation of the BrAD idea, mapping images to their edge maps, already gives a nice improvement over strong self-supervised baselines not utilizing BrAD. We further show that with learnable BrAD, our method demonstrates good gains across various datasets and tasks: UDG, FUDA, and a proposed task of generalization to different domains and classes.

To summarize our contributions are as follows: (i) we propose a new concept of a learnable BrAD - an auxiliary visual ‘bridge’ domain which is relatively easily mapable from domains of interest, seen or unseen, and can drive the learned representation features to be semantically aligned (generalize) across domains; (ii) we show how using the concept of BrAD combined with self-supervised contrastive learning and some additional ideas allows to train an effective (and efficient) model for different kinds of source-labels-limited cross-domain tasks, including UDG, FUDA, and even generalization across multi-domain benchmarks without supervision; (iii) we demonstrate significant gains of up to 14% over UDG and up to 13.3% over FUDA respective state-of-the-art (SOTA) on several benchmarks, as well as showing significant advantages in transferring the learned representations without any additional fine-tuning to new unseen domains and object categories.

2. Related Work

Unsupervised Domain Adaptation (UDA). UDA [17] refers to transferring knowledge from a labeled source domain to an unlabeled target domain. Most UDA methods employ feature distribution alignment using: maximum discrepancy of domain distributions [26, 36, 38, 49, 53, 69, 70], adversarial domain classifier [14, 15, 22, 37, 51, 56, 63], and entropy optimization [27, 37, 47, 48]. GAN based image translation is used in [2, 22, 30, 40, 50, 52]. In [35, 59, 64] source domain data is replaced with a model pre-trained on the source. [39] uses low-level edge features to enforce consistency in UDA for monocular depth estimation. In Few-shot UDA (FUDA) [27, 65], only a few examples per class are labeled in the source domain, while the rest are unlabeled. In our work we take another step further and preform domain adaptation without seeing any labeled examples during training.

Domain Generalization (DG). DG addresses transferring knowledge to domains unseen during training. Most DG works perform supervised training on (a set of) labeled source domains. Methods for supervised DG include: distribution matching across domains [32, 34], adaptive weighting of different source domain losses [46], enforcing low rank of the latent representations [33], and random source domains mixing [60]. Unsupervised DG (UDG) is a new task of training with unlabeled source domains introduced by [68]. The unlabeled source domains images are used for self-supervised pretraining followed by fitting a classifier to the learned representations via labeling a small portion of the source images. We show strong advantages of our method for the UDG setting, even when the unsupervised pretraining features are used directly via kNN without any further classifier fitting.

Self-Supervised Learning (SSL). SSL refers to learning strong semantic feature representations from unlabeled data. Historically, many pre-text tasks were proposed for SSL [13, 16, 28, 42, 43, 67]. Recently, contrastive learning [5–10, 13, 18, 20, 29, 55, 58, 61, 66] has shown great promise and SOTA results. Contrastive methods usually optimize instance discrimination by making two augmentations of the same image closer in feature space than their distance to a set of negative anchors. In our experiments we show an ad-
Finally, a domain discriminator losses I I domains), in a way that for a class mapping C BrAD training architecture. extending the MocoV2 approach \([9]\) to incorporate BrAD domains, even if they are not seen during training. The semantic alignment property will also generalize to other likely be satisfied. Moreover, our overall goal is that this C from domain jointly with SSL on both domains. Re-

Figure 2. BrAD training architecture. \(I_n^1, I_n^2\) are random augmentations of an image \(I_n \in D_n\). \(\Psi_n(I_n^1), \Psi_n(I_n^2)\) are their respective mappings to the bridge domain, \(\Omega\), using a (learned) domain specific image-to-image mapping \(\Psi_n\). The color of the arrows indicates the image that flows through the model: \(I_n^1\) (blue), \(I_n^2\) (pink), \(\Psi_n(I_n^1)\) (green) and \(\Psi_n(I_n^2)\) (red). The negative keys \(K\) of the contrastive losses \(L_{nce}\) come from a domain specific queue. We apply the \(L_\Omega\) regularization, distilling from an edge-mapping \(E\) (can be even a simple Canny, see Sec. 4.4), forcing the bridge domain images to be similar to edge maps, which are (intuitively) less sensitive to domain shift. Finally, a domain discriminator \(A\) and an adversarial loss \(L_{adv}\) improve the domain invariance of the \(\Omega\) projected images representations.

Self-Supervised Learning for UDA and DG. Some UDA and DG methods employ SSL losses in their pipeline. SSL task of solving jigsaw puzzle was leveraged in \([42]\) to aid UDA and DG. \([54]\) combine supervised learning on the source domain jointly with SSL on both domains. Recently \([27, 65]\) proposed a cross-domain SSL approach for Few-shot UDA (FUDA), and \([68]\) proposed an SSL method for UDG. As mentioned, we show strong advantages of our method for both FUDA and UDG tasks.

3. Method

Let \(D = \{D_n\}_{n=1}^N\) be a set of \(N\) domains used for training (e.g., a pair of source and target domains in FUDA \([27, 65]\), or a set of source domains in UDG \([68]\)). Each domain \(D_n\) is represented by a set of unlabeled images \(\{I_n^j\}\), for clarity we will omit \(j\) and denote by \(I_n\) ‘an image’ from domain \(D_n\). Our goal is to train a backbone model \(B\) (e.g., CNN) that projects any image \(I_n \in D_n\) into a \(d\)-dimensional representation space \(\mathcal{F} \subset \mathbb{R}^d\) (shared for all domains), in a way that for a class mapping \(C\) (unknown at training) and any \(I_m \in D_m\) and \(I_r \in D_m\), s.t. \(C(I_m) = C(I_r)\), the semantic alignment property will also generalize to other domains, even if they are not seen during training.

We train \(B\) using contrastive self-supervised learning extending the MocoV2 approach \([9]\) to incorporate BrAD learning and other ideas explained below. Specifically, our training architecture (Figure 2) is comprised of the following components: (1) The backbone \(B: I \to \mathbb{R}^d\) (e.g., ResNet50 with GAP and \(d = 2048\)) - it is the only thing kept after training, the rest of the components are used for training only and later discarded; (2) The projection head \(P: \mathbb{R}^d \to \mathbb{R}^p\) where \(p < d\) (e.g., two-layer MLP with \(p = 128\)) and with \(L_2\) normalized on top; (3) Separate negatives queue \(Q_n\) for each of the domains \(D_n \in D\) - as separating between domains is much easier than separating between classes, we observed that having a single queue (as in \([9]\)) for all the domains hurts performance (as explained in Sec. 4.4); (4) A set of image-to-image models \(\Psi_n : D_n \to \Omega\), for mapping each domain \(D_n \in D\) to the shared (across all seen and unseen domains) auxiliary BrAD domain \(\Omega\) - the \(\Psi_n\) are regularized to produce edge-like images which comprise \(\Omega\), in Sec. 4 we explore and compare several options for \(\Psi_n\); (5) Domain discriminator \(A: \mathbb{R}^d \to \{1, \ldots, N\}\), which is an adversarial domain classifier applied only to the \(\Omega\) images representations, i.e., to \(B(\Psi_n(I_n))\), and trying to predict the original domain index \(n\) of any image \(I_n \in D_n\) projected to \(\Omega\). Intuitively, learning to produce representations that confuse \(A\) better aligns the projections of all the different original domains inside \(\Omega\). (6) The momentum models \(B^m\) and \(P^m\) (as in \([9]\)) - EMA-only updated copies of \(B\) and \(P\) respectively.

The training proceeds in batches of images randomly sampled from all the training domains \(D\) jointly. For clarity we will describe the training flow for a single input image \(I_n \in D_n \in D\). Having \(I_n^1\) and \(I_n^2\) be two augmentations of \(I_n\), we first define the following contrastive loss:

\[
L_{cont}(I_n) = \frac{1}{2} \left( L_{nce}(P(B(I_n^1)), P^m(B^m(\Psi_n(I_n^1)))) + L_{nce}(P(B(\Psi_n(I_n^2))), P^m(B^m(I_n^2))) \right)
\]

where \(L_{nce}(q, k_+, k_-)\) is the standard InfoNCE loss \([19, 20]\) with the query \(q\), the positive key \(k_+\) that attracts \(q\), and
the set of negative keys \( k^- \) that repulse \( q \). Our InfoNCE uses cosine similarity to compare queries and keys. Since in both \( \mathcal{L}_{nce} \) summands of Eq. (1), the positive keys \( k^+ \) are always encoded via the momentum models \( \mathcal{B}^m \) and \( \mathcal{P}^m \) (not producing gradients), we need both these \( \mathcal{L}_{nce} \) in order to train \( \mathcal{B} \) and \( \mathcal{P} \) to represent both the original training domains images \( I_n \in \mathcal{D}_n \) and their BrAD-mappings \( \Psi_n(I_n) \in \Omega \). Note that the first \( \mathcal{L}_{nce} \) of Eq. (1) teaches \( \mathcal{B} \) to extract \( \Omega \)-relevant features directly from each \( \mathcal{D}_n \), which means we can discard the BrAD-mapping models \( \Psi_n \) after training and apply \( \mathcal{B} \) even to unseen domains for which we do not have a learned BrAD-mapping. After each batch is processed, the batch images ‘momentum’ representations are (circularly) queued in accordance to their source domains:

\[
Q_n \leftarrow Q_n \cup \{ \mathcal{P}^m(\mathcal{B}^m(\Psi_n(I_n^{a2}))), \mathcal{P}^m(\mathcal{B}^m(I_n^{a2})) \}
\]  

(2)

Maintaining our queues in this way enables \( \mathcal{D}_n \) images from future training batches to contrast (in \( \mathcal{F} \)) not only against \( \Omega \) projections of other \( \mathcal{D}_n \) images, but also against other images from \( \mathcal{D}_n \) - thus enabling our representation model \( \mathcal{B} \) to complement its set of \( \Omega \)-specific features with some \( \mathcal{D}_n \)-specific ones (e.g. color features). Additionally, we use the following adversarial loss:

\[
\mathcal{L}_{adv}(I_n) = CE(\mathcal{A}(\mathcal{B}(\Psi_n(I_n^{a1}))), n)
\]  

(3)

where \( CE \) is the standard cross-entropy loss and \( n \in \{1, \ldots, N\} \) is the correct domain index for the image \( I_n \). We employ the standard (‘two-optimizers’ in PyTorch) adversarial training scheme for \( \mathcal{L}_{adv} \). In each training batch, the domain discriminator \( \mathcal{A} \) is minimizing \( \mathcal{L}_{adv} \), while blocking \( \mathcal{B} \) and \( \Psi_n \) gradients, whereas \( \mathcal{B} \) and \( \Psi_n \) minimize the negative loss: \(-\mathcal{L}_{adv}\), while blocking the \( \mathcal{A} \) gradients. Note that we employ \( \mathcal{L}_{adv} \) only for the \( \Omega \) projections of the original domains, thus not requiring direct alignment between the domains of \( \mathcal{D} \). Moreover, to reduce ‘competition’ between \( \mathcal{L}_{adv} \) and \( \mathcal{L}_{cont} \), we use the domain discriminator \( \mathcal{A} \) directly on \( \mathcal{B} \)-generated representations (the final features) and not on the projection head \( \mathcal{P} \)-generated representations (temporary features used for efficiency in \( \mathcal{L}_{cont} \)). Lastly, we define the BrAD loss that regularizes the \( \Psi_n \) models to produce edge-like images comprising the shared auxiliary BrAD domain \( \Omega \), which we show to be very effective for different tasks in Sec. 4:

\[
\mathcal{L}_{\Omega}(I_n) = \|\Psi_n(I_n^{a1}) - \mathcal{E}(I_n^{a1})\|^2
\]  

(4)

where \( \mathcal{E} \) is some edge model, which could be heuristic, such as Canny edge detector [4], or pre-trained, such as HED [62], we explore and compare variants in Sec. 4.4. Finally, our full loss for image \( I_n \) is therefore:

\[
\mathcal{L}_f(I_n) = \alpha_1 \cdot \mathcal{L}_{cont}(I_n) + \alpha_2 \cdot \mathcal{L}_{\Omega}(I_n) - \alpha_3 \cdot \mathcal{L}_{adv}(I_n)
\]  

(5)

where the sign in front of \( \mathcal{L}_{adv} \) becomes positive when computing gradients for training the adversarial domain discriminator \( \mathcal{A} \).

**Implementation details.** Our code\(^1\) is in PyTorch [12] and is based on the code of [9]. We set \( \alpha_1, \alpha_2, \alpha_3 = 1 \) in our experiments. The backbone \( \mathcal{B} \) was ResNet-18 [21] for UDG experiments (same as in [68]), and ResNet-50 in FUDA and cross-benchmark generalization experiments (same as in [27]). We used batches of size 256, SGD with momentum 0.9, cosine LR-schedule (from LR 0.03 to 0.002) and trained for 250 epochs for FUDA and 1000 epochs for UDG (same as [68]). We set \( |Q_n| = \min(64K, 2 \cdot |\mathcal{D}_n|) \) and stored only a single pair of (momentum) representations generated from each domain image \( I_n \) and its \( \Omega \) projection (by \( \Psi_n \)). Furthermore, we found it slightly beneficial to exclude the cached version of \( q \) from its \( k^- \) negative keys set when computing the \( \mathcal{L}_{nce}(q, k^+, k^-) \) losses. For \( \mathcal{A} \) we used a 3-layer MLP (1024, 512, 256) with LeakyReLU, followed by a linear domain classifier. For the BrAD-mapping models \( \Psi_n \) architecture, we used the architecture of HED [62] in its PyTorch implementation [41].

### 4. Results

As our BrAD approach is completely unsupervised during training, we used none or limited-supervision cross-domain tasks, specifically Unsupervised Domain Generalization (UDG) [68] and the Few-shotUDA (FUDA) [27, 65], for evaluating its performance and comparing to other self-supervised or source-labels-limited cross-domain methods. Additionally, we evaluate how BrAD and other self-supervised approaches generalize to unseen domains and unseen classes after being trained on a large-scale unlabeled cross-domain data, such as DomainNet [44].

**Datasets.** DomainNet [44] is the largest, most diverse and recent cross-domain benchmark to-date. It is comprised of 6 domains: Real, Painting, Sketch, Clipart, Infograph and Quickdraw, with 345 object classes, 48K - 173K images per domain, and average of 269 images per class. PACS [31] is a standard domain generalization benchmark. It is comprised of 4 domains: Photo, Art, Cartoon and Sketch, with 7 object classes, 2.5K images per domain, and average of 357 images per class. VisDA [45] is a simulation-to-real dataset with 12 classes. The simulation domain is generated via repeated (80-480 times) 3D rendering of instances of 3D object models, 50-150 models per class. It is therefore comprised of only \( \sim1.5K \) distinct object instances.

**Office-Home** [57] is a relatively small dataset consisting of 4 domains: Art, Clipart, Product and Real, with 65 classes, and an average of only 60 images per class.

#### 4.1. Unsupervised Domain Generalization

The UDG task is defined as: (i) unsupervised training on a set of source domains; (ii) using only a small labeled

\(^1\)Available at [https://github.com/leokarlin/BrAD](https://github.com/leokarlin/BrAD)
subset of source domain images to fit a linear classifier on top of the (frozen) features produced by the unsupervised model; and (iii) evaluating the resulting classifier performance on a set of target domains, unseen during training. In our UDG experiments we accurately followed the protocol of the UDG state-of-the-art (SOTA) method DIUL [68], including same backbone arch., same num. epochs, and same subset of classes used for training and testing. Same as [68], we evaluated on DomainNet [44] (Tab. 1) and PACS [31] (Tab. 2). In DomainNet, we train on Clipart, Infograph and Quickdraw and test on unseen Painting, Real and Sketch, and vice versa. For PACS we do a leave-one-domain-out test using the other three as source (repeating this for all domains). Unlike DIUL [68], who used additional full model fine-tuning when amount of source labels was 10% of the source data size, our self-supervised model was never fine-tuned with the source labels (in all cases). Moreover, we also provide kNN results for our method, where we use our resulting features directly without any additional training.

As can be seen from Tab. 1 and Tab. 2, BrAD demonstrates significant gains (both in linear cls. and in kNN modes) not only over [68], but also over a variety of SOTA self-supervised pre-training baselines (probed with the classifier in exactly the same manner as [68] and as described above). This illustrates an important point, the BrAD idea seems to be effective in improving the generalization of self-supervised pre-training to unseen target domains, which, according to these results, seems quite difficult for the current self-supervised SOTA methods.

### 4.2. Few-shot Unsupervised Domain Adaptation

We used the largest and most recent cross-domain dataset, DomainNet [44], to evaluate our BrAD approach FUDA [27, 65] performance. Same as in [65] (and as is common UDA practice), for this evaluation we used only 4 domains: Clipart, Real, Painting and Sketch, and the source-target directions listed in Tab. 3. We follow the

<table>
<thead>
<tr>
<th>Source domains</th>
<th>Target domain</th>
<th>Label Fraction 1%</th>
<th>Label Fraction 5%</th>
<th>Label Fraction 10%</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERM</td>
<td>6.54</td>
<td>10.21</td>
<td>15.10</td>
<td></td>
</tr>
<tr>
<td>BYOL [18]</td>
<td>6.21</td>
<td>9.60</td>
<td>14.55</td>
<td></td>
</tr>
<tr>
<td>MoCo V2 [9, 20]</td>
<td>18.85</td>
<td>28.13</td>
<td>32.46</td>
<td></td>
</tr>
<tr>
<td>AdCo [23]</td>
<td>16.16</td>
<td>30.77</td>
<td>32.25</td>
<td></td>
</tr>
<tr>
<td>SimCLR V2 [8]</td>
<td>23.51</td>
<td>34.03</td>
<td>37.11</td>
<td></td>
</tr>
<tr>
<td>DIUL [68]</td>
<td>18.53</td>
<td>39.32</td>
<td>35.15</td>
<td></td>
</tr>
<tr>
<td>Ours (kNN)</td>
<td>40.65</td>
<td>55.75</td>
<td>60.78</td>
<td></td>
</tr>
<tr>
<td>Ours (linear cls.)</td>
<td>47.26</td>
<td>64.01</td>
<td>68.27</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Accuracy (%) results for UDG on DomainNet. All baseline results are taken from [68]. All methods use ResNet18 backbone and are unsupervisedly pretrained for 1000 epochs before training on the labeled (source only) data. All baselines use a linear classifier (for ours we also include a kNN result that does not utilize any supervised training). ERM indicates the randomly initialized ResNet18. Overall and Avg. indicate the overall test data accuracy and the mean of the per-domain accuracies, respectively. They are different since the test sets of different domains are not of the same size. The reported results are an average over 3 runs. **bold** = best, **blue** = second best.
### 4.3. Generalization to unseen domains and classes

One of the interesting research questions to consider w.r.t. self-supervised learning - is how well it can generalize to unseen domains and classes. Assume one had access to a large collection of diverse unlabeled multi-domain data (e.g. unlabeled DomainNet [44]) needed for training a contrastive self-supervised model. However, also assume that the trained model would need to be used for inference in a few-shot scenario (i.e. with very little data) in a new unseen domain and for a new unseen set of classes.

To test how leading self-supervised methods [5, 6, 9, 10, 66] deal with the proposed scenario (generalization to a mix of seen and unseen domains, as well as mostly unseen classes) and compare their performance to our approach, we conducted the following cross-dataset FUDA generalization experiment, whose results are provided in Tab. 4. We train all methods using their official codes and recommended hyper-parameter and backbone settings on the entire data of Clipart, Real, Painting and Sketch domains from DomainNet. Same ResNet50 backbone is used for all methods (including ours) except Dino [6] that uses the stronger ViT backbone for which it was optimized. We then evaluated the resulting models using a kNN classifier and the FUDA setting detailed in Sec. 4.2, on the OfficeHome [57], PACS [31], and VisDA [45] cross-domain datasets. In all cases, the 1-shot and the 3-shot source domain examples per class were sampled randomly and were kept the same for all methods. This experiment (sampling of shots) was repeated 5 times and in Tab. 4 we report the averages. Similar to UDG experiments in Sec. 4.1, these results indicate again the difficulty for cross-domain generalization inherent to the popular self-supervised learning approaches, and the advantages of BrAD for improving this generalization.

### 4.4. Ablation Studies

In Tab. 5 we evaluate the contribution of the different components of our approach using the FUDA task [27, 65] on the DomainNet dataset [44]. The experimental setting is described in Sec. 4.2 above. Specifically, we show how the average performance of the resulting model in 1-shot comprehensive, we include results for our method in both our intended mode and a pairwise mode. In our intended mode (‘ours’ in Tab. 3) we train a single model on all the domains jointly. In the pairwise mode (‘ours pairwise’ in Tab. 3), we train 7 separate models, one for each source-target domain pair. Besides showing a competitive advantage of our method in all modes, results in Tab. 3 indicate that multi-domain training has clear advantages in efficiency (single model vs. 7 models), ease of use (no need to know the query domain), and performance (about 10% better).

---

Table 2. Accuracy (%) results for the UDG on PACS. For each target domain all other 3 are used as source domains for training. For other details about the number of runs, meaning of column titles and etc., please see Tab. 1 caption. All baseline results are taken from [68].bold = best results, blue = second best.

FUDA protocol defined in [65], where source domain has a single (1-shot) or three (3-shot) labeled images per-class and the rest of its images are provided as unlabeled. We used the same classes and the exact indices of the labeled samples for each case as provided by [65] for repeatability. Our results and comparison to other methods are summarized in Tab. 3. According to the protocol of [65], all compared methods models are initialized with ImageNet pretraining and operate in transductive setting. All the methods besides ours use the respective 1 or 3 samples per class during their training. In our case, we used those samples only during the inference - either as the search space in the kNN, or for training the linear classifier (on these few labeled source images only). All the methods besides ours are designed for working separately on each pair of source and target domains. Therefore, for the comparison to be comprehensive, we include results for our method in both our intended mode and a pairwise mode. In our intended mode (‘ours’ in Tab. 3) we train a single model on all the domains jointly. In the pairwise mode (‘ours pairwise’ in Tab. 3), we train 7 separate models, one for each source-target domain pair. Besides showing a competitive advantage of our method in all modes, results in Tab. 3 indicate that multi-domain training has clear advantages in efficiency (single model vs. 7 models), ease of use (no need to know the query domain), and performance (about 10% better).

---

2 According to the released official code of [65], transductive = utilize the entire domains data (including unlabeled test data) in their training. Transductive setting adds about 3 – 4% to the performance (Sec. 4.4).

3 One of the known limitations of self-supervised contrastive learning methods (including ours) is the need to observe relatively large amount of different instances per class (naturally without class labels).
Table 3. 1-shot/3-shot accuracy (%) results for FUDA task [65] on DomainNet. All baseline results except CDS [27] are taken from [65]. The CDS results were kindly provided by its authors and are higher than those reported in [65]. bold = best results, blue = second best.

<table>
<thead>
<tr>
<th>DD</th>
<th>MQ</th>
<th>BrAD</th>
<th>Transductive</th>
<th>ImageNet pretrained</th>
<th>1-shot</th>
<th>3-shots</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>16.9</td>
<td>21.9</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>16.5</td>
<td>22.0</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>21.0</td>
<td>22.3</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>20.6</td>
<td>27.9</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>31.5</td>
<td>39.8</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>29.8</td>
<td>38.0</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>34.3</td>
<td>42.1</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>37.8</td>
<td>47.1</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>44.1</td>
<td>55.2</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>48.6</td>
<td>58.3</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>47.4</td>
<td>59.3</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>38.7</td>
<td>51.1</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>41.4</td>
<td>52.7</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>41.6</td>
<td>51.9</td>
</tr>
</tbody>
</table>

Table 5. BrAD ablation study using the FUDA task on DomainNet dataset with km classifier. bold = best, blue = second best. The highlighted row is our full method in Sec. 4.2 settings. The “learned*” is our full method without using HED pretrained weights.

Table 4. 1-shot / 3-shot accuracy (%) cross dataset results. The models are trained from scratch on DomainNet and tested on the OfficeHome, PACS and VisDA. We report an average over 5 runs randomizing the shots. bold = best, blue = second best.
and replaced the BSDS500 pretrained HED model used as $\mathcal{E}$ in the $L_\Omega$ loss (Eq. (4)) with a simple blurred Canny edge map (more details in the Supplementary). This way *not having* any BSDS500 pretrained weights anywhere in our system. As can be seen from the corresponding “learned*” row in Table 5, there is almost no change in the end result (-1.2/+1.0) indicating our approach can work just as well without BSDS500 pretraining of HED. Please refer to the Supplementary for more details. Finally, we verified that the strong gains we observed due to the introduction of BrAD models $\Psi_n$ do not disappear with ImageNet pretraining and transductive setting. As can be seen in the corresponding rows of Tab. 5, both Canny and frozen HED BrAD variants maintain moderate gains of up to 2.9% (over no-BrAD mode), while our full method with the learned $\Psi_n$ BrAD maintains large gains (+9.9/+7.2) as expected.

5. Conclusions and Limitations

In this paper, we have proposed a novel self-supervised cross-domain learning method based on semantically aligning (in feature space) all the domains to a common BrAD domain - a learned auxiliary bridge domain accompanied with relatively easy to learn image-to-image mappings to it. We have explored a special case of the edge-regularized BrAD - specifically driving BrAD to be a domain of edge-map-like images. In this implementation, we have shown significant advantages of our proposed approach for the important limited-source-labels tasks such as FUDA and UDG, as well as for a proposed task of generalization between cross-domain benchmarks to potentially unseen domains and classes. We observed a significant improvement over previous unsupervised and partially supervised methods for these tasks. Future work may also include exploration of the edge-like transforms used here as potentially useful augmentations for contrastive SSL in general.

Limitations of the current paper include: (i) intentional focusing only on edge-like bridge domains, which is one of the simplest BrAD one may construct. Naturally this bares limitations, e.g., lowering the relative importance of representing non-edge related features such as color. Thus, exploring other non-edge bridge domains is an important topic for future work; (ii) our current approach is built on top of a very useful, yet a single SSL method, namely, MoCo [9]. A direct extension could be employing our approach on top of a vision transformer backbone using the SSL method of [6], or more broadly, making it applicable to any SSL method; (iii) finally, our approach being completely unsupervised in pre-training, lacks control over which semantic classes are formed in the learned representation space, which is a common problem shared with most current SSL techniques that might lead to missing the classes that are under-represented in terms of different instance count in the unlabeled data. Addressing this in a follow-up work may include adding such control via some form of zero-shot or few-shot priming, or by training with coarse labels [3].

Acknowledgments This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-19-C-1001. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA. Raja Giryes was supported by ERC-StG grant no. 757497 (SPADE). Dina Katabi was supported by funding from the MIT-IBM Watson AI lab.
References


