

A Style and Semantic Memory Mechanism for Domain Generalization*

Yang Chen[†], Yu Wang[‡], Yingwei Pan[‡], Ting Yao[‡], Xinmei Tian[†], and Tao Mei[‡]

[†] University of Science and Technology of China, Hefei, China

[‡] JD AI Research, Beijing, China

cheny01@mail.ustc.edu.cn, {feather1014, panyw.ustc, tingyao.ustc}@gmail.com, xinmei@ustc.edu.cn, tmei@jd.com

Abstract

Mainstream state-of-the-art domain generalization algorithms tend to prioritize the assumption on semantic invariance across domains. Meanwhile, the inherent intra-domain style invariance is usually underappreciated and put on the shelf. In this paper, we reveal that leveraging intra-domain style invariance is also of pivotal importance in improving the efficiency of domain generalization. We verify that it is critical for the network to be informative on what domain features are invariant and shared among instances, so that the network sharpens its understanding and improves its semantic discriminative ability. Correspondingly, we also propose a novel “jury” mechanism, which is particularly effective in learning useful semantic feature commonalities among domains. Our complete model called STEAM can be interpreted as a novel probabilistic graphical model, for which the implementation requires convenient constructions of two kinds of memory banks: semantic feature bank and style feature bank. Empirical results show that our proposed framework surpasses the state-of-the-art methods by clear margins.

1. Introduction

Machine learning models are usually deployed in scenarios where the test data are unknown beforehand. This phenomenon can lead to dangerous consequences especially when the predictions are used for life-threatening occasions such as medical diagnosis. The prediction might be seriously erroneous due to the distributional gap between training data and test data. It is therefore critical for machine learning algorithms to maintain safe and reliable predictions that generalize well across domains. The goal of domain generalization (DG) approaches [20, 21, 41, 55] is to solve this issue by leveraging labeled data from multiple training domains. However, we observed that mainstream state-of-the-art DG algorithms tend to only prioritize the semantic

invariance assumption across domains, while the style invariance within each domain is usually ignored. In this paper, we reveal that intra-domain style invariance is also of pivotal importance to improve DG approaches. Particularly, we propose a novel model to incorporate both intra-domain style invariance and inter-domain semantic invariance for DG tasks. The proposed framework is called **STEAM**, which relies on a **STyle** and **sEmAntic Memory** mechanism to practically implement our proposed assumptions.

In contrast to existing DG works [7, 53, 55], STEAM further benefits from the hypothesis that instances from the same domain should share style information. The motivation is that the simple constraint helps efficiently disentangle the style feature, and therefore eases the search for true semantic feature with a reduced degree of freedom. To reach this goal, we resort to the recently prevailing self-supervised learning paradigm that makes our assumptions practically accessible. Our first objective is to achieve intra-domain style invariance by conveniently resorting to contrastive loss. Given the invariance assumption, style features corresponding to each domain are discovered, and the network can further learn semantic features along the directions mostly orthogonal to the domain styles. Since the semantic feature is considered as the true causal factor affecting the instance category, STEAM effectively helps reduce overfitting to the domain styles through the above mechanisms. Most importantly, we also force the network to respect the conventional semantic invariance among domains. Specifically, we require that each pair of samples from the same class to compute a similarity score with all the semantic features in “memory”. These two samples need to reach a consensus on every such similarity score when sweeping through all the stored semantic features. We name this procedure “jury” mechanism, and we elaborate the mathematical net effect of such mechanism in the paper.

To summarize, our contributions in this paper include: 1. We explore the assumption of instance level intra-domain style invariance and justify the empirical advantage by incorporating such assumptions into the DG framework. 2. We propose a “jury” mechanism, which efficiently learns

*This work was performed at JD AI Research.

domain-agnostic semantic features beneficial for classification tasks. Such mechanism is capable of efficiently preventing semantic features from overfitting to domain styles. 3. We observe that the proposed STEAM not only generalizes well on DG benchmarks, but also can be conveniently modified for domain adaptation (DA) problems.

2. Related Work

Domain Adaptation (DA) algorithms aim to exploit both annotated training data in the source domain and unlabeled samples in the target domain. Mainstream DA approaches [5, 11, 32, 31, 50] usually penalize distributional misalignment between source and target data via, e.g., maximum mean discrepancy loss [23, 25] or adversarial loss [3, 24, 38]. Given the success of DA algorithms, multi-source domain adaptation (MSDA) methods [33, 48, 52] consider scenarios where multiple sources are available for training to better improve generalization.

Domain Generalization (DG) algorithms also assume access to multiple labeled training source domains. However, target data is unavailable during training for DG approaches, leading to a more challenging yet a more practical setting than DA problems. Many early DG approaches [20, 21, 28] borrowed the idea of distribution alignment from DA to reduce the distributional gap between multiple training sources. Some recent DG methods consider generating extra synthetic images given the multiple source domains, so that test data distributed closely to the training data are “in-distribution” with the training data [1, 41, 42, 53]. Some methods decompose the network parameters into domain-specific and domain-invariant parts during training, while only the domain-invariant parameters are used for predictions at test time. For instance, [18] develops a low-rank parameterized CNN model where each layer of the network is decomposed into “common” and “specific” components. In [34], only the last layer of the network is decomposed to serve the goal of DG. Several normalization and meta-learning strategies are also considered for domain generalization such as [19, 39, 55].

Contrastive Learning has shown impressive performance in the context of self-supervised learning [6, 10, 15, 27, 46, 47, 51]. Representative contrastive learning [15] proposed efficient memory bank constructions so that the historical features are conveniently stored and reused even if batchsize is small. We feel inspired from these approaches [15] where the memory bank help retain temporal consistency between features, and we introduce such a memory mechanism to best facilitate our motivations. However, our approach is neither targeting a pre-train task nor an unsupervised learning problem, in contrast to [10, 15, 47]. Rather, we look into DG problems and we aim to learn the desired feature invariances respecting our unique hypothesis. We notice a related contrastive learning method in [27], that

enforces a particular prediction regularizer across augmentations to improve in-class consistency. We argue that the work in [27] is an unsupervised algorithm that completely distinguishes its nature from our DG task, while the loss proposed here also leads to an entirely different interpretation and application.

We observe that existing DG methods along the decomposition path either: hinge on inter-domain semantic invariance features assumption, so that the network becomes agnostic to styles; or they rely on style decomposition approaches to synthesize more data. In contrast, our proposed method enjoys two exclusive novel assumptions: 1. We are the first to impose instance level intra-domain style invariance during the training. Having these domain-specific style features at hand, we effectively reduce the degree of freedom of the problem in further learning the useful semantic features. 2. We design a novel “jury” mechanism that is different from any existing DG method, which achieves significant improvement in domain generalization.

3. Method

In the regime of out of distribution (OOD) detection, it has been shown that data distribution is heavily affected by population-level background statistics [12, 29, 35, 40]. Owing to this issue, OOD inputs can rather be classified into in-distribution classes with high confidence, given the presence of dominant background noise. Recent observations [12, 29] have shown that deep generative models can even assign a higher likelihood to OOD inputs. One reason is that simple parameterization on marginal input distribution can be significantly confounded and dominated by background statistics, and does not learn much useful parameterization on the variation of *semantic features*. Accordingly, work such as [35] aims to mute the effect of these background statistics via the likelihood ratio method, to achieve better OOD detection based on cleaner semantic features. Although the literature on OOD detection investigates completely orthogonal topics and directions against the DG community, we feel intrigued and motivated to reformulate domain generalization problems with novel background noise priors. The ultimate goal is to relieve the network optimization on the background statistics, so that the network steers away from its overfitting to these domain specific background noise, while focusing on learning true semantic distributional commonalities among domains.

We therefore do not seek to directly close the distribution gap between data distributions across domains (that GAN and MMD methods normally do). We impose prior knowledge that *instances from the same domain share hidden invariant background statistics*. Under this constraint, the network reduces uncertainty and redundancy when searching for semantic features that might be glimmering in comparison to the dominant background style statistics.

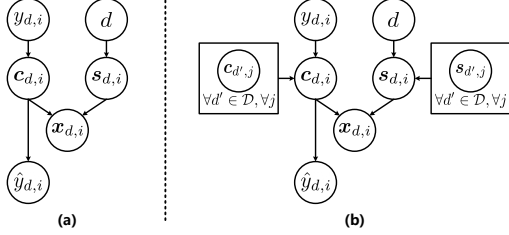


Figure 1. Probabilistic graphical model comparisons for (a) conventional causal model for DG (b) causal model for STEAM, based on memory bank constructions. For STEAM, all the training samples have an impact on deciding the semantic and style feature for each instance. $\hat{y}_{d,i}$ is the class prediction given feature $c_{d,i}$.

3.1. Problem Formulation

For DG problems, we consider D source domains $\mathcal{D} = \{\mathcal{D}_d\}_{d=1}^D$, with each d -th domain \mathcal{D}_d includes N_d training pairs $\{(\mathbf{x}_{d,i}, y_{d,i})\}_{i=1}^{N_d}$, where $\mathbf{x}_{d,i}$ is the i -th sample in \mathcal{D}_d , and $y_{d,i} \in \{1, 2, \dots, n_c\}$ is the label of $\mathbf{x}_{d,i}$. n_c is the number of classes shared across domains. The goal of DG approach is to learn a model from multiple labeled source domains that generalizes well to an unseen target domain \mathcal{D}_T .

We define notations frequently used throughout the paper. We assume each training sample is projected to feature embedding via CNN encoders [16, 22]. Specifically, the image $\mathbf{x}_{d,i}$ firstly is input into feature extractor: $\mathbf{z}_{d,i} = E_f(\mathbf{x}_{d,i}, \theta_{e,f})$, where function E_f extracts image feature $\mathbf{z}_{d,i}$ out of $\mathbf{x}_{d,i}$ by a CNN parameterization $\theta_{e,f}$. A semantic encoder E_c then reads in $\mathbf{z}_{d,i}$ and produces semantic feature $\mathbf{c}_{d,i} = E_c(\mathbf{z}_{d,i}, \theta_{e,c})$. In parallel, a style encoder simultaneously inputs the $\mathbf{z}_{d,i}$ feature and maps it into style representation via $\mathbf{s}_{d,i} = E_s(\mathbf{z}_{d,i}, \theta_{e,s})$. We define a classifier $C(\mathbf{c}_{d,i}, \phi)$, where function C is parameterized by ϕ and used to classify semantic features among n_c possible classes. Here, the above encoder parameters $\Theta_e = \{\theta_{e,f}, \theta_{e,c}, \theta_{e,s}\}$ and classifier parameters ϕ are CNN parameters learned during training. Subscript e associated with each notation, e.g., $\theta_{e,c}$, is intended to be reminiscent of “encoder”.

We name the proposed method “Style and Semantic Memory Mechanism (STEAM)” for Domain Generalization. The overall STEAM model can be interpreted as a probabilistic graphical model in Fig. 1(b). Each semantic feature $\mathbf{c}_{d,i}$ and style feature $\mathbf{s}_{d,i}$ both depend on statistics (rectangles) from every other instances available in memory (given unlimited memory, the whole dataset then). Our motivation is that human intrinsically define classes by contrasting. Take for instance, Labrador and Husky are both defined as dogs, apparently because they share higher similarity than with any other species. In comparison, conventional DG methods as shown as in Fig. 1(a) only models each instance’s semantic feature $\mathbf{c}_{d,i}$ independently from other $\mathbf{c}_{d',j}$. Note STEAM hinges on enormous historical

training data. It is therefore critical that our loss function breaks free from the limitation of batchsize. A simple solution would be to directly store the learned features out of the encoder into a static memory bank for later usage [47]. Unfortunately, similar to the observation in [15], we find features stored in this way cannot retain any temporal representation consistency due to the rapidly update of encoder via backpropagation. We thus simultaneously maintain an alternative memory encoder: $E_m = \{E_{m,f}, E_{m,c}, E_{m,s}\}$ with parameters $\Theta_m = \{\theta_{m,f}, \theta_{m,c}, \theta_{m,s}\}$ that is able to slowly release the historical feature representations into the memory bank in an momentum updated way.

Specifically, we make sure the constructed memory encoder shall involve identical architecture and functioning components mirroring everything in E_c, E_f, E_s above: We have memory feature encoder: $E_{m,f}(\cdot, \theta_{m,f})$, memory style feature encoder $E_{m,s}(\cdot, \theta_{m,s})$, and memory semantic feature encoder $E_{m,c}(\cdot, \theta_{m,c})$. The only difference is that the parameters $\Theta_e = \{\theta_{e,f}, \theta_{e,c}, \theta_{e,s}\}$ are updated via backpropagation, whereas the parameters of memory encoder $\Theta_m = \{\theta_{m,f}, \theta_{m,c}, \theta_{m,s}\}$ are only momentum updated according to the changes of Θ_e . Subscript m is intended to be reminiscent of “memory”. We elaborate the usage of memory encoders in Section 3.2, and Section 3.3.

Generally, a bird’s-eye view of our whole framework is illustrated in Fig. 2: We maintain an encoder to extract features of input images, and use a parallel memory encoder to generate and release memory features in the memory bank. Within both encoder and memory encoder structure, we both include a style encoder component and semantic encoder component. We explain the usage of these components in the following sections.

3.2. Intra-domain Invariance on Style Features

Our first objective is to impose cross-instance intra-domain style invariance via a memory bank construction. The plan is to maintain D domain specific style banks that gradually become agnostic to semantics as the training progresses, while each style bank retains intra-domain invariant and inter-domain contrastive style features at instance level.

We compute the style feature of each $\mathbf{x}_{d,i}$ via memory encoder: $\bar{\mathbf{s}}_{d,i} = E_{m,s}(E_{m,f}(\mathbf{x}_{d,i}, \theta_{m,f}), \theta_{m,s})$, and we sequentially push each arriving $\bar{\mathbf{s}}_{d,i}$ into the “domain style bank” \mathbf{V}_d^s . We define D number of parallel style memory banks $\mathbf{V}^s = (\mathbf{V}_1^s, \dots, \mathbf{V}_d^s, \dots, \mathbf{V}_D^s, d \in [1, D])$ and dynamically update each \mathbf{V}_d^s bank like queues: We push each newly arriving style representation $\bar{\mathbf{s}}_{d,i}$ into the queue tail of \mathbf{V}_d^s , and remove the oldest style feature from the queue head as in [15]. Assuming the memory size of each bank \mathbf{V}_d^s is a constant B . The entries stored in \mathbf{V}_d^s are denoted with brand new subscripts as: $[\mathbf{v}_{d,1}^s, \dots, \mathbf{v}_{d,j}^s, \dots, \mathbf{v}_{d,B}^s], j \in [1, B]$, where each $\mathbf{v}_{d,j}^s$ dynamically stores the style feature vector $\bar{\mathbf{s}}_{d,\cdot}$ at the j position of the \mathbf{V}_d^s bank.

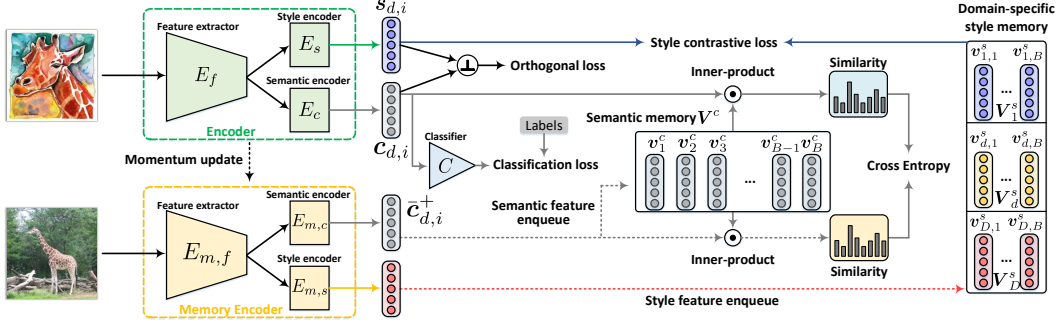


Figure 2. The framework of STEAM. We train an encoder for style and semantic feature extraction. We maintain a memory encoder to obtain D number of parallel style memory banks and one semantic memory bank. We use contrastive loss based on style banks to achieve intra-domain style invariance. We construct a memory semantic feature bank (“jury”) to achieve inter-domain semantic invariance.

Recall that we desire to figure out the shared invariant style features embedded behind each domain. To achieve this goal, we require every style feature out of the style encoder $s_{d,i} = E_s(E_f(x_{d,i}, \theta_{e,f}), \theta_{e,s})$ to have high similarity score with any stored style features from the same domain in memory V_d^s , whereas similarity score between style feature $s_{d,i}$ and all features from other domain banks $V_{d'}^s, d' \neq d$ remains low. We find contrastive loss [15, 43] a natural fit to fulfill this goal:

$$\mathcal{L}_s = -\frac{1}{Z_s} \sum_{d,i,j} \log \frac{\exp(\langle s_{d,i}, v_{d,j}^s \rangle / \tau)}{\exp(\langle s_{d,i}, v_{d,j}^s \rangle / \tau) + \sum_{d' \neq d} \sum_{\ell=1}^B \exp(\langle s_{d,i}, v_{d',\ell}^s \rangle / \tau)}, \quad (1)$$

where $Z_s = B \cdot \sum_d N_d$ normalizes sample number, τ is a temperature parameter and $\langle x_1, x_2 \rangle = x_1^T x_2 / \|x_1\| \|x_2\|$ denote the cosine similarity between x_1 and x_2 .

It is transparent that Eq. (1) is essentially a softmax function aiming to distinguish each $(s_{d,i}, v_{d,j}^s)$ intra-domain pair from the total $B \times (D-1)$ number of $(s_{d,i}, v_{d',\ell}^s), d' \neq d$ inter-domain pairs. As the training proceeds, the instance level style feature $s_{d,i}$ compares with all the members in the domain style bank $v_{d,j}^s, j \in [1, B]$, in order to reach consensus on intra-domain style invariance of domain d . In other words, Eq. (1) penalizes feature misalignment between $s_{d,i}$ and $v_{d',\ell}^s, d' \neq d$, whereas the style features from distinct domains are pushed away, i.e., any increase in inner product $\langle s_{d,i}, v_{d',\ell}^s \rangle, d' \neq d$ increases loss Eq. (1). These D number of domain specific style banks therefore gradually become agnostic to semantics during the training, as each bank V_d^s retains intra-domain invariant and inter-domain contrastive style features at instance level.

3.3. Inter-domain Invariance on Semantic Features

We have tentative style features at hand. We now turn to our second objective: to learn inter-domain semantic invariant feature via another memory bank. Bear in mind that these semantic features are considered the true causal variables that determine the semantics of samples independent of domains. We define $x_{d,i}^+$ to be a “variant” of each training input $x_{d,i}$. We call $x_{d,i}^+$ a “variant” of $x_{d,i}$, because it is considered as semantically identical to $x_{d,i}$. The sample

$x_{d,i}^+$ is randomly chosen from all possible D domains that either is from the same class of $x_{d,i}$, or simply is selected from the augmentation pool (RandAug [13]) of image $x_{d,i}$.

Feature $c_{d,i}$ is considered the true causal variable that determines the semantics of training samples. We desire that a semantic feature $c_{d,i} = E_c(E_f(x_{d,i}, \theta_{e,f}), \theta_{e,c})$ to return a high similarity score with the semantic feature of any possible $x_{d,i}^+$, whereas $c_{d,i}$ remains distinct to samples from other classes. We certainly cannot build n_c classes number of parallel semantic banks analogously to what we did for style banks, because there might be millions of classes. We correspondingly come up with a novel “jury” mechanism that relaxes our objective.

Construction of Semantic Jury Memory Bank. The semantic “jury” bank’s construction relies on memory encoders: We sequentially push any arriving semantic memory feature $\bar{c}_{d,i}^+ = E_{m,c}(E_{m,f}(x_{d,i}^+, \theta_{m,f}), \theta_{m,c})$ into the single semantic feature memory bank V^c regardless of whose “variant” $\bar{c}_{d,i}^+$ is and which domain d is. We update the entries in V^c , again, like maintaining a queue structure: entries stored in V^c are denoted as: $V^c = [v_1^c, \dots, v_j^c, \dots, v_B^c], j \in [1, B]$. Note features $c_{d,i}$ and $\bar{c}_{d,i}^+$ are semantically identical, as they represent the different samples from the same class. But how far are $c_{d,i}$ and $\bar{c}_{d,i}^+$ in the semantic embedding space?

We leave the judgment to the “jury”. All the queuing semantic features in the bank have the right to bid, contribute, and weigh their contribution into the similarity measurement between $\bar{c}_{d,i}^+$ and $c_{d,i}$. Mathematically, we denote the probability $\mathbf{p}(x_{d,i}; \theta_{e,c}, \theta_{e,f}, V^c) = [p_1^e, \dots, p_j^e, \dots, p_B^e], j \in [1, B]$ as similarity score between $c_{d,i}$ with respect to all stored semantic features in semantic memory V^c , where each probability entry p_j^e is defined as:

$$p_j^e = \frac{\exp(\langle c_{d,i}, v_j^e \rangle / \tau)}{\sum_{v^c \in V^c} \exp(\langle c_{d,i}, v^c \rangle / \tau)}. \quad (2)$$

Similarly, we also compute $\mathbf{p}(x_{d,i}^+; \theta_{m,c}, \theta_{m,f}, V^c) = [p_1^m, \dots, p_j^m, \dots, p_B^m]$, which is the similarity scores between $\bar{c}_{d,i}^+$ with respect to each of the member in the V^c :

$$p_j^m = \frac{\exp(\langle \bar{c}_{d,i}^+, v_j^c \rangle / \tau)}{\sum_{v^c \in V^c} \exp(\langle \bar{c}_{d,i}^+, v^c \rangle / \tau)}. \quad (3)$$

The motivation here is that, the ‘‘jury’’ V^c would cross check every ‘‘jury’’ member’s similarity score with both $\bar{c}_{d,i}^+$ and $c_{d,i}$. The bank V^c summaries these two distribution respectively into $p(\mathbf{x}_{d,i}; \theta_{e,c}, \theta_{e,f}, V^c)$ and $p(\mathbf{x}_{d,i}^+; \theta_{m,c}, \theta_{m,f}, V^c)$ as in Eq. (2) and Eq. (3). The assumption here is that, if $\mathbf{x}_{d,i}^+$ and $\mathbf{x}_{d,i}$ truly share invariant semantic features, then their similarity score across the entire semantic feature bank V^c shall be as close as possible, too. We therefore penalize cross entropy between $p(\mathbf{x}_{d,i}^+; \theta_{m,c}, \theta_{m,f}, V^c)$ and $p(\mathbf{x}_{d,i}; \theta_{e,c}, \theta_{e,f}, V^c)$:

$$\mathcal{L}_c = -\frac{1}{Z_c} \sum_{d,i} p(\mathbf{x}_{d,i}^+; \theta_{m,c}, \theta_{m,f}, V^c) \log p(\mathbf{x}_{d,i}; \theta_{e,c}, \theta_{e,f}, V^c), \quad (4)$$

where $Z_c = \sum_d N_d$ normalizes over samples. Eq. (4) penalizes misalignment between probability $p(\mathbf{x}_{d,i}; \theta_{e,c}, \theta_{e,f}, V^c)$ and $p(\mathbf{x}_{d,i}^+; \theta_{m,c}, \theta_{m,f}, V^c)$ via similarity cross-checks with all the features stored in the V^c . As a result, features in the current V^c jointly vote on distributional similarity between $\mathbf{x}_{d,i}^+$ and $\mathbf{x}_{d,i}$. This strategy strongly contrasts with conventional contrastive learning, where for MoCo like mechanisms, only a single positive key is considered for each query, whereas negative keys in the memory bank are only considered as negative and contrastive to the positives.

But why not directly penalize the inner product $\langle c_{d,i}, \bar{c}_{d,i}^+ \rangle$ or the likes which seems a more obvious option? If one takes a closer look at Eq. (4), the net effect of the ‘‘jury’’ mechanism is that, any feature member, say v_j^c in the bank V^c having a relatively high similarity score with both $c_{d,i}$ and $\bar{c}_{d,i}^+$ would vote for agreement on this similarity, and then v_j^c becomes confident to contribute itself to jointly supervise the optimization direction to further improve semantic invariance among $c_{d,i}$, $\bar{c}_{d,i}^+$ and v_j^c . The strength of this vote depends on v_j^c ’s similarity score with each $c_{d,i}$ and $\bar{c}_{d,i}^+$. This makes one reminiscent of popular multi-view contrastive learning strategy [8, 43], where introducing more views for each instance is always beneficial for learning. Conversely, all feature members in the jury showing low similarity scores with both $c_{d,i}$ and $\bar{c}_{d,i}^+$ would automatically tend to contrast itself away from both $c_{d,i}$ and $\bar{c}_{d,i}^+$. The loss therefore inherently summarizes the compounding effect of all features in the bank via such joint similarity cross-check mechanism, in comparison to any conventional contrastive learning approaches that banks are only considered ‘‘negative’’. We notice a related work [27], that also enforces invariant prediction regularizer across augmentations. We argue that the work in [27] is a completely unsupervised algorithm that distinguishes itself from our DG task. Notably, the proposed bank definition here out of the specific DG invariance hypothesis is also orthogonal to [27], and the loss Eq. (4) proposed here leads to entirely different interpretation and implementation.

To make sure the features are indeed semantically dis-

criminative, we apply supervised classification loss on $\mathbf{x}_{d,i}$:

$$\mathcal{L}_{cls} = -\frac{1}{N} \sum_{i=1}^N y_{d,i} \log C(c_{d,i}, \phi), \quad (5)$$

where classifier $C(c_{d,i}, \phi)$ predicts the class of sample $\mathbf{x}_{d,i}$ by only using its semantic feature $c_{d,i}$.

3.4. Decoupling Semantics from Styles

Last but not least, current loss on semantic features $c_{d,i}$ is completely disconnected from its style features $s_{d,i}$, and cannot benefit from the style invariance assumption at all. A simple solution to fix this is to enforce orthogonality [4] between $c_{d,i}$ and $s_{d,i}$. Let H_c and H_s be matrices whose rows are the semantic representations $c_{d,i}$ and style representations $s_{d,i}$ from $\mathbf{x}_{d,i}$. We constrain the semantic feature to go towards the orthogonal direction against style features, such that the semantic encoder $E_c(\cdot, \theta_{e,c})$ and classifier $C(c_{d,i}, \phi)$ could mostly avoid overfitting to the style features. This is formalized into the squared Frobenius norm:

$$\mathcal{L}_o = \|H_c^T H_s\|_F^2. \quad (6)$$

3.5. Overall Loss Function

Taking into account all the discussions above, the eventual loss function is:

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_s + \mathcal{L}_c + \mathcal{L}_o, \quad (7)$$

where all present losses are equally weighted. The overall loss is backpropagated through the network in a batchwise manner. The parameters of encoder $\Theta_e = \{\theta_{e,f}, \theta_{e,c}, \theta_{e,s}\}$ and classifier ϕ are updated via backpropagation, whereas the parameters of memory encoder $\Theta_m = \{\theta_{m,f}, \theta_{m,c}, \theta_{m,s}\}$ are momentum updated as:

$$\Theta_m = \alpha \times \Theta_m + (1 - \alpha) \times \Theta_e, \quad (8)$$

where $\alpha \in [0, 1)$ is a momentum coefficient. We only use obtained encoder parameters $\theta_{e,f}, \theta_{e,c}$ and classifier parameters ϕ during test time inference.

3.6. Extension to MSDA

Another advantage of the proposal is that, loss Eq. (7) can be transferred to deal with multi-source domain adaptation (MSDA) tasks with few modifications. The only difference is that for target data without any annotations, only those augmentation images of $\mathbf{x}_{d,i}$ are used to define $\mathbf{x}_{d,i}^+$ and there is no classifier applied on target data semantic features, as the class label is not available. Except for this difference, all target data can be conveniently plugged into our algorithm with a domain ID, $d \in [1, D]$. The training procedure then progresses under the exactly same losses and network constructions as we did for DG tasks.

Table 1. Performance (%) comparisons with the state-of-the-art approaches for DG.

Method	Digits-DG					PACS				Office-Home					
	MNIST	MNIST-M	SVHN	SYN	Avg	Art	Cartoon	Photo	Sketch	Avg	Artistic	Clipart	Product	Real World	Avg
MMD-AAE [20]	96.5	58.4	65.0	78.4	74.6	75.2	72.7	96.0	64.2	77.0	56.5	47.3	72.1	74.8	62.7
CCSA [28]	95.2	58.2	65.5	79.1	74.5	80.5	76.9	93.6	66.8	79.4	59.9	49.9	74.1	75.7	64.9
JiGen [7]	96.5	61.4	63.7	74.0	73.9	79.4	75.3	96.0	71.6	80.5	53.0	47.5	71.5	72.8	61.2
CrossGrad [41]	96.7	61.1	65.3	80.2	75.8	79.8	76.8	96.0	70.2	80.7	58.4	49.4	73.9	75.8	64.4
Epi-FCR [19]	-	-	-	-	-	82.1	77.0	93.9	73.0	81.5	-	-	-	-	-
EISNet [45]	-	-	-	-	-	81.9	76.4	95.9	74.3	82.1	-	-	-	-	-
L2A-OT [53]	96.7	63.9	68.6	83.2	78.1	83.3	78.2	96.2	73.6	82.8	60.6	50.1	74.8	77.0	65.6
DecAug [1]	-	-	-	-	-	79.0	79.6	95.3	75.6	82.4	-	-	-	-	-
MixStyle [55]	96.5	63.5	64.7	81.2	76.5	84.1	78.8	96.1	75.9	83.7	58.7	53.4	74.2	75.9	65.5
Vanilla	95.8	58.8	61.7	78.6	73.7	77.0	75.9	96.0	69.2	79.5	58.9	49.4	74.3	76.2	64.7
STEAM	96.8	67.5	76.0	92.2	83.1	85.5	80.6	97.5	82.9	86.6	62.1	52.3	75.4	77.5	66.8

4. Experiments

4.1. Evaluation on Domain Generalization

Datasets. We perform DG tasks via extensive evaluations on the following benchmarks: (1) **Digits-DG** [53] includes 4 domains (*MNIST* [17], *MNIST-M* [14], *SVHN* [30] and *SYN* [14]) with an evident domain shift in font style, stroke color and background. (2) **PACS** [18] is a widely used domain generalization benchmark, which is composed of four domains (*Art Painting*, *Cartoon*, *Photo* and *Sketch*). Each domain includes samples from 7 different categories, including a total of 9,991 samples. (3) **Office-Home** [44] contains around 15,500 images of 65 classes, distributed across 4 domains (*Artistic*, *Clipart*, *Product* and *Real world*). (4) **DomainNet** [33] is a recently established large-scale dataset for multi-source domain adaptation and domain generalization, which includes about 0.6 million images in 345 classes distributed across 6 domains (i.e., *Clipart*, *Infograph*, *Quickdraw*, *Painting*, *Real*, *Sketch*).

For a fair comparison with prior works, we follow the standard leave-one-domain-out evaluation procedure as in [7, 18, 19], where one domain is chosen as the unseen target and the remaining domains are used as source domains for training. For Digits-DG, PACS and Office-Home, the authors of [53, 54, 55] have specified particular train and val splits for each domain to ensure a fair comparison. They use the entire *train + val* target data as the *test* data. We use the same data split definition for our experiments. For DomainNet, according to [9], we employ their dataset division and report the accuracy on the *test* split of target domain.

Implementation details. Following the backbone setting of [53], we use 4 *conv* layers and a softmax layer for Digits-DG dataset. ReLU and 2×2 max-pooling are inserted after each convolution layer. The model is trained with SGD, initial learning rate of 0.05 and batch size of 128 for 50 epochs. For both PACS and Office-Home, we use ResNet-18 pretrained on ImageNet as the CNN backbone, as in [53, 55]. We train the model with SGD, initial learning rate of 0.002 and batch size of 30 for 60 epochs. The learning rate is further decayed by the cosine annealing rule. For DomainNet, we experiment with ResNet-18 and ResNet-50 backbone architectures, as in [9]. For all experiments, the semantic encoder, style encoder and classifier are all imple-

mented using a fully connected (FC) layer. The size of style and semantic memory bank is set as 2,048.

Baselines. To evaluate our method, we consider comparisons with the following approaches: (1) **Vanilla** simply trains the plain classification model on all available source domains using all annotations, the model is then directly used to classify target samples. (2) **CrossGrad** [41] perturbs input using adversarial gradients from a domain classifier. (3) **CCSA** [28] explores a contrastive semantic alignment loss for domain-invariant representation learning. (4) **MMD-AAE** [20] imposes an MMD loss on the hidden layers of an autoencoder. (5) **JiGen** [7] utilizes an auxiliary self-supervision loss so that the features can be used to solve the Jigsaw puzzle task. (6) **Epi-FCR** [19] designs an episodic training strategy. (7) **EISNet** [45] develops a momentum metric learning scheme with the K -hard negative mining to improve the network generalization ability. (8) **L2A-OT** [53] synthesizes extra data from pseudo-novel domains to augment the source domains. (9) **DecAug** [1] generates extra data augmentations through perturbing the disentangled style feature and semantic features. (10) **DMG** [9] learns domain specific masks for generalization on different domains. (11) **MixStyle** [55] mixes instance level feature statistics of training samples across various sources to introduce more domain diversity via synthesizing.

Results on Digits-DG. Table 1 shows that, our method exhibits clear advantages over the existing state-of-the-art methods. Please note that our model is especially effective and commanding on challenging DG directions, e.g., *MNIST-M* and *SVHN*, as they seem to have large domain variations compared with other directions. This is a valid justification that the instance level style invariance along with the proposed “jury” mechanism together is a legitimate idea to deal with domain generalization problems. Compared with the methods that hinge on marginal distribution alignment across domains, e.g., MMD-AAE and CCSA, our model even demonstrates around 8.5% performance boost on average. This well validates our assumption in Section 3.2, that intra-domain style invariance seems to be a more practical and suitable hypothesis in presence of computing background statistics hidden in domain styles. In other words, the prior knowledge on intra-domain style invariance effectively reduce the uncertainty when search-

Table 2. Leave-one-domain-out results on DomainNet for DG.

	Method	Clp	Inf	Pnt	Qdr	Rel	Skt	Avg.
ResNet-18	Vanilla	56.5	18.4	45.3	12.4	57.9	38.8	38.2
	Multi-Headed [9]	55.4	17.5	40.8	11.2	52.9	38.6	36.1
	MetaReg [2]	53.6	21.0	45.2	10.6	58.4	42.3	38.5
	DMG [9]	60.0	18.7	44.5	14.1	54.7	41.7	39.0
	STEAM	58.3	22.1	47.4	14.4	58.6	45.9	41.1
ResNet-50	Vanilla	64.0	23.6	51.0	13.1	64.4	47.7	44.0
	Multi-Headed [9]	61.7	21.2	46.8	13.8	58.4	45.4	41.2
	MetaReg [2]	59.7	25.5	50.1	11.5	64.5	50.0	43.6
	DMG [9]	65.2	22.1	50.0	15.6	59.6	49.0	43.6
	STEAM	64.6	27.0	54.0	15.8	65.6	52.2	46.5

ing for optimal network parameters so that the parameters reduces overfitting to domain styles.

Results on PACS. This part of results is shown in Table 1. Our method achieves the best performance on all test domains. Note the recently proposed EISNet also involves the usage of a feature memory. However, the memory in EISNet is only used for the sake of hard triplets selection without consideration on feature invariance. In DecAug [1], similar semantic-style orthogonal regularization loss was used. However, orthogonality is perhaps the most widely used tool everywhere, and we simply employ orthogonality as an auxiliary regularizer. Nevertheless, our motivation and hypothesis are completely different from DecAug, highlighting the outstanding performance given our instance-level style invariance and “jury” mechanism.

Results on Office-Home. The results are shown in Table 1. Our STEAM achieves the best average performance. Notably, the simple vanilla model shows strong results on this benchmark. Most of baselines provide only marginal improvements than vanilla model that are under 1.0%. As discussed in L2A-OT, this might be owing to the fact that Office-Home is relatively a large composition of data, compared with PACS and Digits-DG, thus offering inherently bigger domain diversity in training data already. In contrast, our method shows the best performance on average with an impressive 2.1% improvement over the Vanilla.

Results on DomainNet. DomainNet is considered as perhaps the most challenging benchmark, owing to its dataset size, both in terms of image number and category numbers. Table 2 shows that, on DomainNet dataset, the Vanilla baseline achieves competitive results in comparison to domain generalization methods MetaReg and DMG, while our method again surpasses all competitors. We observe that our model is leading in the table with improved average performance, especially when ResNet-18 and ResNet-50 are used as the backbone architectures, showing respectively 2.1% and 2.5% accuracy improvement. At this point, it is worthy to mention that our method has achieved better robustness, for consistently offering better performance than other baselines.

Table 3. Domain adaptation results on PACS.

Method	Art	Cartoon	Photo	Sketch	Avg
Source-only	74.9	72.1	94.5	64.7	76.6
DANN [14]	81.9	77.5	91.8	74.6	81.5
MDAN [52]	79.1	76.0	91.4	72.0	79.6
WBN [26]	89.9	89.7	97.4	58.0	83.8
MCD [37]	88.7	88.9	96.4	73.9	87.0
M ³ SDA [33]	89.3	89.9	97.3	76.7	88.3
CMSS [49]	88.6	90.4	96.9	82.0	89.5
STEAM	94.0	93.7	99.3	85.1	93.0

Table 4. Domain adaptation results on miniDomainNet.

Method	Clipart	Painting	Real	Sketch	Avg
Source-only	63.4	49.9	61.5	44.1	54.7
MCD [37]	62.9	45.7	57.5	45.8	53.0
DCTN [48]	62.0	48.7	58.8	48.2	54.4
DANN [14]	65.5	46.2	58.6	47.8	54.6
M ³ SDA [33]	64.1	49.0	57.7	49.2	55.0
MME [36]	68.0	47.1	63.3	43.5	55.5
DAEL [54]	69.9	55.1	66.1	55.7	61.7
STEAM	71.4	61.9	71.1	60.9	66.3

4.2. Evaluation on Domain Adaptation

Datasets and implementation details. To justify STEAM’s validity on the application of MSDA, we implement STEAM under the problem definition described in Section 3.6. We firstly consider evaluation on PACS dataset again with the same problem definition and setting of [49]. Next, we follow [54] and use miniDomainNet for evaluation, which is a sampled subset of DomainNet and reformatted into a smaller image size (96 × 96). In general, miniDomainNet consists of 4 domains (*Clipart*, *Painting*, *Real* and *Sketch*) across 126 classes, which mostly resembles data diversity of the original DomainNet. For PACS, we use ResNet-18 pretrained on ImageNet as the CNN backbone, by following training protocols in [49]. Batch size is 32. For miniDomainNet, we use ResNet-18 as the CNN backbone, the same used as in [54]. Similarly, we use SGD with momentum as the optimizer, and the learning rate decays according to cosine annealing rule. The model is trained with an initial learning rate of 0.005 for 60 epochs. For each mini-batch, we sample from each domain 64 images.

Results. Given the standard test protocol in [33] on PACS, we use one domain as target and the remaining as sources. Classification accuracy on the target domain test set is reported. We compare our STEAM with two state-of-the-art multi-source domain adaptation approaches: M³SDA [33] and CMSS [49]. In addition, we also present the following methods as our baselines: DANN [14], MDAN [52], WBN [26], MCD [37]. The results are shown in Table 3. Our method achieves the state-of-the-art average accuracy of 93.0%. On the most challenging *sketch* domain, we obtain 85.1%, clearly outperforming other baselines. On the miniDomainNet, we compare with the same methods presented in DAEL [54], including MCD [37], DCTN

Table 5. Ablation on Digits-DG. MT, MM, SV, and SY indicates *MNIST*, *MNIST-M*, *SVHN*, and *SYN*, respectively.

Method	\mathcal{L}_{cls}	\mathcal{L}_s	\mathcal{L}_o	\mathcal{L}_c	MT	MM	SV	SY	Avg.
Vanilla	✓				95.8	58.8	61.7	78.6	73.7
Vanilla-style	✓	✓	✓		96.3	63.6	69.3	82.4	77.9
Vanilla-semantic	✓			✓	96.0	65.2	74.5	86.2	80.5
STEAM	✓	✓	✓	✓	96.5	67.5	76.0	92.2	83.0

[48], DANN [14], M³SDA [33], MME [36]. The results are shown in Table 4. Our method achieves 66.3% average accuracy, again, justifying significant advantages out of our interesting invariance hypothesis.

4.3. Further Analysis

Ablation study. We investigate the impact of each component in Eq. (7) by comparing several variations of STEAM using the Digits-DG and PACS datasets. **Vanilla:** a conventional supervised learning formulation that uses all source domains for training, i.e., with training loss \mathcal{L}_{cls} ; **Vanilla-style:** we add intra-domain style invariance loss and orthogonal regularization to Vanilla model, i.e., train using loss $\mathcal{L}_{cls} + \mathcal{L}_s + \mathcal{L}_o$; **Vanilla-semantic:** we only add inter-domain invariance constraint on semantic features, i.e., training loss $\mathcal{L}_{cls} + \mathcal{L}_c$; **STEAM:** our complete training scheme, with training loss defined in Eq. (7). Note we don’t separate the \mathcal{L}_s and \mathcal{L}_o here, but please see supplementary file for more related ablation studies.

Table 5 and 6 display the results. Notably, we observe three phenomenons: (1) *Vanilla-style* outperforms *Vanilla* by average 4.2% and 3.5% on Digits-DG and PACS, supporting the advantage of “intra-domain style invariance” hypothesis for domain generalization alone; (2) *Vanilla-semantic* also surpasses over *Vanilla* by 6.8% on Digits-DG, and 4.9% on PACS, which well validates the effectiveness of semantic invariance imposed through “jury” mechanism alone. (3) By leveraging on both intra-domain style invariance and inter-domain semantic invariance assumptions, our entire STEAM framework further drastically improves the performance across all settings, indicating the complementary roles of these two factors.

Design choices for STEAM. We now explore alternative definitions for each loss term, to better understand the essence behind STEAM. (1) **Domain classification:** we replace the entire style bank and associated instance level contrastive loss described in Eq. (1) by a domain classifier. We then use the domain classifier to predict the domain label, so that the style features becomes domain-discriminative. (2) **L2-matching:** we replace Eq. (4), by directly minimizing the l_2 -distance of each semantic feature pairs $\|c_{d,i} - \bar{c}_{d,i}^+\|_2^2$ where $c_{d,i}$ and $\bar{c}_{d,i}^+$ share identical class label. (3) **Contrastive:** we define semantic features $\bar{c}_{d,i}^+$ as the positive key of $c_{d,i}$, with random instances sampled from the memory encoders forming V^c as negative samples. We then perform conventional instance level

Table 6. Ablation on PACS for domain generalization. A, C, P, and S indicates *Art*, *Cartoon*, *Photo*, and *Sketch*, respectively.

Method	\mathcal{L}_{cls}	\mathcal{L}_s	\mathcal{L}_o	\mathcal{L}_c	A	C	P	S	Avg.
Vanilla	✓				77.0	75.9	96.1	69.2	79.5
Vanilla-style	✓	✓	✓		80.2	78.7	96.5	76.6	83.0
Vanilla-semantic	✓			✓	83.3	77.6	96.7	80.1	84.4
STEAM	✓	✓	✓	✓	85.5	80.6	97.5	82.9	86.6

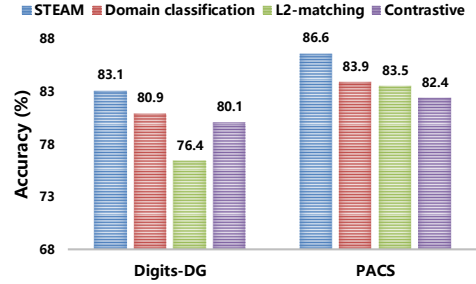


Figure 3. Study on different design choices of STEAM.

Info-NCE loss on $(c_{d,i}, \bar{c}_{d,i}^+)$.

Fig. 3 shows that NONE of the above replacements using alternative losses have achieved any better design than our original STEAM framework. The *Domain classification* achieves some descent test accuracy though (worse than STEAM), while apparently the idea of instance level style invariance is a much stronger prior than simply requiring the style features to be linear separable. *L2-matching* performs worse than our STEAM, meaning directly minimizing the geometry distance within each $(c_{d,i}, \bar{c}_{d,i}^+)$ pair has ignored important information to be inter-semantically contrastive. Finally, if we would use plain Info-NCE loss as in contrastive learning, i.e., *Contrastive*, worse performance is observed, showing the superiority of our proposed “jury” mechanism for inter-domain semantic invariance learning.

5. Conclusion

In this paper, we propose a novel algorithm capitalizing on both Style and sEmAntic memory mechanism (STEAM) for domain generalization tasks. Importantly, we find leveraging on intra-domain style invariance can lead to a significant improvement on the efficacy of domain generalization. The intra-domain style invariance prior can help improve the learning of semantic features, owing to reduced overfitting to domain styles during training. We introduce efficient memory bank construction policies for both style and semantic features that store useful statistics for computing our losses. Specifically, our semantic feature bank serves as a “jury” and helps effectively improve intra-class invariance cross different domains. Empirical results verify our assumption on various benchmarks.

Acknowledgments. This work was funded by JD AI RESEARCH, NSFC No. 61872329 and the Fundamental Research Funds for the Central Universities under contract WK3490000005.

References

- [1] Haoyue Bai, Rui Sun, Lanqing Hong, Fengwei Zhou, Nanyang Ye, Han-Jia Ye, S-H Gary Chan, and Zhenguo Li. Decaug: Out-of-distribution generalization via decomposed feature representation and semantic augmentation. In *AAAI*, 2020.
- [2] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chelappa. Metareg: Towards domain generalization using meta-regularization. In *NeurIPS*, 2018.
- [3] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, 2017.
- [4] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *NeurIPS*, 2016.
- [5] Qi Cai, Yingwei Pan, Chong-Wah Ngo, Xinmei Tian, Lingyu Duan, and Ting Yao. Exploring object relation in mean teacher for cross-domain detection. In *CVPR*, 2019.
- [6] Qi Cai, Yu Wang, Yingwei Pan, Ting Yao, and Tao Mei. Joint contrastive learning with infinite possibilities. In *NeurIPS*, 2020.
- [7] Fabio M Carlucci, Antonio D’Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *CVPR*, 2019.
- [8] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020.
- [9] Prithvijit Chattopadhyay, Yogesh Balaji, and Judy Hoffman. Learning to balance specificity and invariance for in and out of domain generalization. In *ECCV*, 2020.
- [10] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.
- [11] Yang Chen, Yingwei Pan, Ting Yao, Xinmei Tian, and Tao Mei. Mocycle-gan: Unpaired video-to-video translation. In *ACM MM*, 2019.
- [12] Hyunsun Choi and Eric Jang. Generative ensembles for robust anomaly detection. *arXiv:1810.01392*, 2018.
- [13] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPRW*, 2020.
- [14] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015.
- [15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.
- [16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [17] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.
- [18] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *ICCV*, 2017.
- [19] Da Li, Jianshu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy M Hospedales. Episodic training for domain generalization. In *ICCV*, 2019.
- [20] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *CVPR*, 2018.
- [21] Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep domain generalization via conditional invariant adversarial networks. In *ECCV*, 2018.
- [22] Yehao Li, Ting Yao, Yingwei Pan, and Tao Mei. Contextual transformer networks for visual recognition. *arXiv preprint arXiv:2107.12292*, 2021.
- [23] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *ICML*, 2015.
- [24] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *NeurIPS*, 2018.
- [25] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *ICML*, 2017.
- [26] Massimiliano Mancini, Lorenzo Porzi, Samuel Rota Bulo, Barbara Caputo, and Elisa Ricci. Boosting domain adaptation by discovering latent domains. In *CVPR*, 2018.
- [27] Jovana Mitrovic, Brian McWilliams, Jacob Walker, Lars Buesing, and Charles Blundell. Representation learning via invariant causal mechanisms. In *ICLR*, 2021.
- [28] Saeid Motiian, Marco Piccirilli, Donald A Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *ICCV*, 2017.
- [29] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don’t know? In *ICLR*, 2019.
- [30] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bisacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop*, 2011.
- [31] Yingwei Pan, Ting Yao, Yehao Li, Chong-Wah Ngo, and Tao Mei. Exploring category-agnostic clusters for open-set domain adaptation. In *CVPR*, 2020.
- [32] Yingwei Pan, Ting Yao, Yehao Li, Yu Wang, Chong-Wah Ngo, and Tao Mei. Transferrable prototypical networks for unsupervised domain adaptation. In *CVPR*, 2019.
- [33] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *ICCV*, 2019.
- [34] Vihari Piratla, Praneeth Netrapalli, and Sunita Sarawagi. Efficient domain generalization via common-specific low-rank decomposition. In *ICML*, 2020.
- [35] Jie Ren, Peter J Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark A DePristo, Joshua V Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. In *NeurIPS*, 2019.
- [36] Kuniaki Saito, Donghyun Kim, Stan Sclaroff, Trevor Darrell, and Kate Saenko. Semi-supervised domain adaptation via minimax entropy. In *ICCV*, 2019.

- [37] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*, 2018.
- [38] Swami Sankaranarayanan, Yogesh Balaji, Carlos D Castillo, and Rama Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. In *CVPR*, 2018.
- [39] Seonguk Seo, Yumin Suh, Dongwan Kim, Jongwoo Han, and Bohyung Han. Learning to optimize domain specific normalization for domain generalization. In *ECCV*, 2020.
- [40] Joan Serra, David Álvarez, Vicenç Gómez, Olga Slizovskaia, José F Nuñez, and Jordi Luque. Input complexity and out-of-distribution detection with likelihood-based generative models. In *ICLR*, 2020.
- [41] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. In *ICLR*, 2018.
- [42] Nathan Somavarapu, Chih-Yao Ma, and Zsolt Kira. Frustratingly simple domain generalization via image stylization. *arXiv preprint arXiv:2006.11207*, 2020.
- [43] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *ECCV*, 2020.
- [44] Hemant Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, 2017.
- [45] Shujun Wang, Lequan Yu, Caizi Li, Chi-Wing Fu, and Pheng-Ann Heng. Learning from extrinsic and intrinsic supervisions for domain generalization. In *ECCV*, 2020.
- [46] Yu Wang, Jingyang Lin, Qi Cai, Yingwei Pan, Ting Yao, Hongyang Chao, and Tao Mei. A low rank promoting prior for unsupervised contrastive learning. *arXiv preprint arXiv:2108.02696*, 2021.
- [47] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018.
- [48] Ruijia Xu, Ziliang Chen, Wangmeng Zuo, Junjie Yan, and Liang Lin. Deep cocktail network: Multi-source unsupervised domain adaptation with category shift. In *CVPR*, 2018.
- [49] Luyu Yang, Yogesh Balaji, Ser-Nam Lim, and Abhinav Srivastava. Curriculum manager for source selection in multi-source domain adaptation. In *ECCV*, 2020.
- [50] Ting Yao, Yingwei Pan, Chong-Wah Ngo, Houqiang Li, and Tao Mei. Semi-supervised domain adaptation with subspace learning for visual recognition. In *CVPR*, 2015.
- [51] Ting Yao, Yiheng Zhang, Zhaofan Qiu, Yingwei Pan, and Tao Mei. Seco: Exploring sequence supervision for unsupervised representation learning. In *AAAI*, 2021.
- [52] Han Zhao, Shanghang Zhang, Guanhang Wu, José MF Moura, Joao P Costeira, and Geoffrey J Gordon. Adversarial multiple source domain adaptation. In *NeurIPS*, 2018.
- [53] Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Learning to generate novel domains for domain generalization. In *ECCV*, 2020.
- [54] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain adaptive ensemble learning. *arXiv preprint arXiv:2003.07325*, 2020.
- [55] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. In *ICLR*, 2021.