Adaptive Methods for Real-World Domain Generalization

Abhimanyu Dubey[*]
MIT
dubeya@mit.edu

Vignesh Ramanathan
Facebook AI
vigneshr@fb.com

Alex Pentland
MIT
pentland@mit.edu

Dhruv Mahajan
Facebook AI
dhruvm@fb.com

Abstract

Invariant approaches have been remarkably successful in tackling the problem of domain generalization, where the objective is to perform inference on data distributions different from those used in training. In our work, we investigate whether it is possible to leverage domain information from the unseen test samples themselves. We propose a domain-adaptive approach consisting of two steps: a) we first learn a discriminative domain embedding from unsupervised training examples, and b) use this domain embedding as supplementary information to build a domain-adaptive model, that takes both the input as well as its domain into account while making predictions. For unseen domains, our method simply uses few unlabelled test examples to construct the domain embedding. This enables adaptive classification on any unseen domain. Our approach achieves state-of-the-art performance on various domain generalization benchmarks. In addition, we introduce the first real-world, large-scale domain generalization benchmark, Geo-YFCC, containing $1.1M$ *samples over* 40 *training,* 7 *validation and* 15 *test domains, orders of magnitude larger than prior work. We show that the existing approaches either do not scale to this dataset or underperform compared to the simple baseline of training a model on the union of data from all training domains. In contrast, our approach achieves a significant* 1% *improvement.*

1. Introduction

Domain generalization refers to the problem of learning a classifier from a heterogeneous collection of distinct *training* domains that can generalize to new unseen *test* domains [7]. Among the various formalizations proposed for the problem [19, 6], the most effective is *domain-invariant* learning [4, 15]. It learns feature representations invariant to the underlying domain, providing a *universal* classifier that can generalize to new domains [26, 26, 33, 25, 16].

It can be demonstrated that domain-*invariant* classifiers

[*]Work done while visiting Facebook AI.

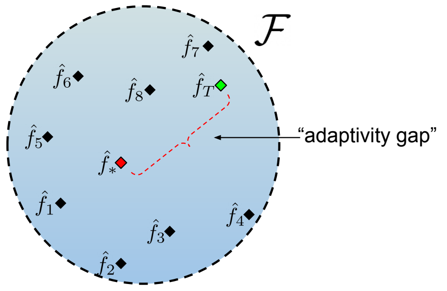


Figure 1: A visual for *adaptive* domain generalization. Here we denote by $\hat{f}_i$ the optimal classifier in a class $\mathcal{F}$ for domain i (out of 8). For test domain D_T, the optimal *universal* classifier $\hat{f}_*$ may be far from the optimal classifier for D_T.

minimize the *average risk* across domains [33]. However, this may not guarantee good performance for any *specific* test domain, particularly if the distribution of domains has high variance, as visualized heuristically in Figure 1. In this case, the optimal classifier for a target domain can lie far from the optimal *universal* classifier. In our work, we propose an *adaptive* classifier that can be adapted to any new domain using very few unlabelled samples without any further training. Unlike invariant approaches, this requires a few *unsupervised* samples from any domain while *testing*[1]. However, this requirement is trivial, since this set is available by definition in all practical settings.

Our approach consists of two steps. We first embed each domain into a vector space using very few unlabelled samples from the domain. Next, we leverage these *domain embeddings* as supplementary signals to learn a *domain-adaptive* classifier. During testing, the classifier is supplied the corresponding embedding obtained from test samples. Our contributions can be summarized as follows.

1. We adapt low-shot *prototypical learning* [39] to construct *domain embeddings* from unlabelled samples of each domain. We also provide an algorithm to use these embeddings as supplementary signals to train *adaptive* classifiers.
2. We justify our design choices with novel theoreti-

[1]This does not correspond to unsupervised domain adaptation, where unsupervised samples are assumed to be present during *training*.

cal results based on the framework of kernel mean embeddings [18]. Furthermore, we leverage the theory of risk minimization under product kernel spaces [7] to derive generalization bounds on the average risk for adaptive classifiers.

3. We introduce the first large-scale, real-world domain generalization benchmark, dubbed **Geo-YFCC**, which contains 40 training, 7 validation and 15 test domains, and an overall $1.1M$ samples. Geo-YFCC is constructed from the popular YFCC100M [42] dataset partitioned by geographical tags, and exhibits several characteristics of *domain shift* [23], *label shift* [3], and long-tailed class distributions [49, 45, 9] common in real-world classification. Because of its scale and diversity, this benchmark is substantially more challenging than the incumbent benchmarks.

4. On existing benchmarks and notably, two large-scale benchmarks, we demonstrate the strength of fine-tuning (ERM) with *adaptive* classification over existing approaches. In addition to the effectiveness of adaptive classification, this suggests, along with the claims of Gulrajani and Lopez-Paz [19] that rigorous model selection and large benchmarks are essential in understanding domain generalization, where naive ERM can also be a powerful baseline.

2. Related Work

Our paper is inspired by several areas of research, and we enumerate the connections with each area sequentially.

Domain Generalization. The problem of domain generalization was first studied as a variant of multi-task learning in Blanchard *et al.* [7]. Many *domain-invariant* approaches [41, 25, 26], have been developed using the popular domain adaptation algorithm introduced in [15], where the authors learned invariant representations via adversarial training. This has been followed by alternative formulations for invariant learning such as MMD minimization [26], correlation alignment [41], and class-conditional adversarial learning [25]. Other approaches include meta learning [24], invariant risk minimization [2], distributionally robust optimization [38], mixup [46, 47, 44], and causal matching [32]. *Adversarial training* with improvements [15, 35] has also been used to learn invariant representations. Complementary to these approaches, we focus instead on learning *adaptive* classifiers that are specialized to each target domain[2]. Our approach also does not use any unsupervised data from unseen domains during training, as is done in the problem of unsupervised domain adaptation [17, 4].

Domain-Adaptive Learning. The idea of using *kernel mean embeddings* (KME) for adaptive domain generalization was proposed in the work of Blanchard *et al.* [7]. Kernel mean embeddings have also been used for personalized learning in both multi-task [11] and multi-agent learning [13] bandit problems. A rigorous treatment of *domain-adaptive* generalization in the context of KME approaches is provided in Deshmukh *et al.* [12]. Our approach complements this theoretical line of work, by providing an efficient algorithm for classification, that requires only a few unsupervised samples for competitive performance. We also extend the results of [7] to a larger class of functions.

Large-Scale Learning. We propose domain generalization benchmarks based on both the ImageNet LSVRC12 [10] and YFCC100M [42] datasets, which have been instrumental in accelerating research for image classification. We consider the challenges encountered specifically for large-scale computer vision, which is plagued by issues such as long-tailed data distributions [27], and large training times [1]. It is important to note that in large-scale settings, it is difficult to perform extensive hyper-parameter tuning and optimal model selection owing to large training times.

3. Approach

We assume that each *domain* $D \in \mathcal{D}$ is a probability distribution over $\mathcal{X} \times \mathcal{Y}$, i.e., the product of the input space $\mathcal{X}$ and output space $\mathcal{Y}$, and there exists a *mother* distribution $\mathfrak{P}$ which is a measure over the space of domains $\mathcal{D}$. A *training domain* $\widehat{D}(n)$ is obtained by first sampling a domain D from $\mathfrak{P}$, and then sampling n points (X and Y) from $\mathcal{X} \times \mathcal{Y}$ following D. A training set can then be constructed by taking the union of N such domains $(\widehat{D}_i(n))_{i=1}^N$. Correspondingly, any test domain $\widehat{D}_T(n_T)$ is also constructed by first drawing a sample $D_T \sim \mathcal{D}$, then drawing n_T samples from D_T, and discarding the labels.

Consider a family of functions $\mathcal{F}$. For each $D \in \mathcal{D}$, the optimal classifier f_D in $\mathcal{F}$ (for some loss function ℓ) can be given by $f_D = \arg\min_{f\in\mathcal{F}} \mathbb{E}_{(\mathbf{x},y)\sim D}[\ell(f(\mathbf{x}), y)]$. We denote the optimal empirical risk minimization (ERM) classifier for each domain $\widehat{D}_i$ in the training set domain $D_i(n)$ as $\hat{f}_i = \arg\min_{f\in\mathcal{F}} \mathbb{E}_{(\mathbf{x},y)\sim D(n)}[\ell(f(\mathbf{x}), y)]$. The *universal* expected risk minimizer (i.e., classifier minimizing overall risk over $D \sim \mathfrak{P}$) can then be given by f_* such that,

$$f_* = \arg\min_{f\in\mathcal{F}} \mathbb{E}_{D\sim\mathfrak{P}} \mathbb{E}_{\mathbf{x},y\sim D}[\ell(f(\mathbf{x}), y)]. \tag{1}$$

Typically, the optimal classifier for D_T within $\mathcal{F}$ may be far from the (non-adaptive) universal classifier $\hat{f}_*$, a distance which we call the "adaptivity gap". A visual heuristic representation of the above intuition is provided in Figure 1. Ideally, we would like the classifier to be chosen closer to $\hat{f}_T$ such that it obtains lower error. Note that the *adaptivity gap* may be large even when the training set contains samples close to $\hat{f}_T$ (in the figure, this refers to functions $\hat{f}_7$ and $\hat{f}_8$), since the *universal* classifier minimizes overall risk.

Our approach reduces this discrepancy by learning an *adaptive* classifier, i.e., a function $F : \mathcal{X} \times \mathcal{D} \rightarrow \mathcal{Y}$ that

[2]Note that our algorithm is not opposed to invariant learning, we merely focus on learning *distribution-adaptive* classifiers, that can be combined with invariant approaches with ease, see Section 5.1.

takes *both* the point $\mathbf{x}$ and the domain D of $\mathbf{x}$ into consideration. Our goal is to make sure that for any test domain D_T, the resulting classifier $f_T = F(\cdot, D_T)$ from F lies close to $\hat{f}_T$. We assume that a small set of unsupervised data is available from D_T to characterize $F(\cdot, D_T)$ from F[3]. We follow a two step approach:

(A) **Computing domain embeddings.** We first learn a function $\boldsymbol{\mu} : D \rightarrow \mathbb{R}^{d_{\mathcal{D}}}$, that maps any domain $D \in \mathcal{D}$ (including empirical domains) to a finite vector (domain embedding). Ideally, we want this embedding to be learned from a few samples belonging to the domain. To achieve this, we leverage kernel mean embeddings [34], a formalization that can represent probability distributions as the average of some feature vectors $\Phi_{\mathcal{D}}$ in a suitable Hilbert space. $\boldsymbol{\mu}$ is obtained as the feature $\Phi_{\mathcal{D}}(\mathbf{x})$ averaged over all $\mathbf{x} \in D$.

(B) **ERM using augmented inputs.** With a suitable domain embedding $\boldsymbol{\mu}$, we learn a neural network F directly over the *augmented input* $(\mathbf{x}, \boldsymbol{\mu}(\widehat{D}))$, where $\mathbf{x}$ is a point from the domain $\widehat{D}$, using regular ERM (i.e., minimizing cross-entropy loss from predictions, see Algorithm 2).

Inference. For an unseen domain $\widehat{D}_T$, we first compute the domain embedding $\boldsymbol{\mu}(\widehat{D}_T)$ using unsupervised samples (or the test set when none are present) from $\widehat{D}_T$ and the learned function Φ from (A). We then provide the computed embedding as an additional input to the model F from (B) to get the *adaptive* model $F(\cdot, \boldsymbol{\mu}(\widehat{D}_T))$.

3.1. Prototypical Domain Embeddings

Kernel Mean Embeddings. An elegant approach to embed distributions into vector spaces is the nonparametric method of *kernel mean embeddings* (KME) [34]. The fundamental idea behind KMEs is to consider each probability distribution $D \in \mathcal{D}$ as a member of a reproducing kernel Hilbert space (RKHS) $\mathcal{H}(\mathcal{X})$ with some kernel k and feature $\Phi_{\mathcal{D}} : \mathbb{R}^d \rightarrow \mathbb{R}^{d_D}$. Then, the KME of any distribution $D \in \mathcal{D}$ can be given as $\boldsymbol{\mu}(D) = \int_{\mathcal{X}} \Phi_{\mathcal{D}}(\mathbf{x}) dD(\mathbf{x})$.

The strength of this approach is that, if a suitable $\Phi_{\mathcal{D}}$ is chosen, then it is possible to learn a function over the space of distributions $\mathcal{D}$ via the average feature embedding of samples drawn from D. Hence, we propose to learn both a domain embedding $\boldsymbol{\mu}$ using features $\Phi_{\mathcal{D}}$ and the target function $F(\cdot, \boldsymbol{\mu}(\cdot))$, such that when any domain D is presented, we can provide an *adapted* function $F(\cdot, \boldsymbol{\mu}(D))$. We learn a neural network $\Phi_{\mathcal{D}} : \mathcal{X} \rightarrow \mathbb{R}^{d_D}$, parameterized by weights $\boldsymbol{\theta}$, and then compute the n-sample *empirical* KME $\boldsymbol{\mu}$ of each training domain $\widehat{D}$ as:

$$\boldsymbol{\mu}(\widehat{D}) = \frac{1}{n} \sum_{\mathbf{x} \in \widehat{D}} \Phi_{\mathcal{D}}(\mathbf{x}; \boldsymbol{\theta}). \tag{2}$$

[3]This assumption is trivially satisfied by the test set itself. Note that we do not access D_T during training and hence we do not perform any *transductive learning*, unlike unsupervised domain adaptation.

Low-Shot Domain Prototypes. To be useful in classification, we require $\boldsymbol{\mu}$ to satisfy two central criteria:

(A) **Expressivity.** $\boldsymbol{\mu}$, and consequently, $\Phi_{\mathcal{D}}$, must be expressive, i.e., for two domains D and D', $\|\boldsymbol{\mu}(D) - \boldsymbol{\mu}(D')\|_2$ must be large if they are very distinct (i.e., domain shift), and small if they are similar. For example, if $\Phi_{\mathcal{D}}$ is constant, the KME will provide no additional information about D, and we can expect the *expected risk* itself to be large in this case, equivalent to non-adaptive classification.

(B) **Consistency.** Φ must have little dependence on both the choice, and the number of samples used to construct it. This is desirable as we only have access to the empirical distribution $\widehat{D}$ corresponding to any domain D. Consistency can be achieved, if we learn $\Phi_{\mathcal{D}}$ such that for any sample $\mathbf{x}$ from domain D, $\Phi_{\mathcal{D}}(\mathbf{x})$ is strongly clustered around $\boldsymbol{\mu}(D)$.

We show that *prototypical networks* [39] originally proposed for the task of few-shot learning, can be adapted to construct domain embeddings which satisfy both criteria. We train the network based on the algorithm from Snell *et al.* [39], but using domain identities as the labels instead of class labels. The network is trained such that embeddings of all points from a domain are tightly clustered and far away from embeddings corresponding to points of other domains. In Section 3.3, we demonstrate consistency by deriving a concentration bound for the clustering induced by the network. Expressivity is guaranteed by the loss function used to train the network, which forces domain embeddings to be discriminative, as we show next.

Our prototypical network (embedding function $\Phi_{\mathcal{D}}$) is trained with SGD [8]. At each iteration, we first sample a subset of N_t domains from the training domains. Next, we sample two sets of points from each domain. The first set is used to obtain the embedding for the domain, by running the points through the network $\Phi_{\mathcal{D}}$ and averaging (Eq. 2). This results in an approximate embedding $\boldsymbol{\mu}(\widehat{D})$ for each domain D. Every point $\mathbf{x}$ in the second set is given a probability of belonging to a domain D:

$$p_{\boldsymbol{\theta}}(\mathbf{x} \in D) = \frac{\exp\left(-\|\boldsymbol{\mu}(\widehat{D}) - \Phi_{\mathcal{D}}(\mathbf{x})\|_2^2\right)}{\sum_{i=1}^{N_t} \exp\left(-\|\boldsymbol{\mu}(\widehat{D}_i) - \Phi_{\mathcal{D}}(\mathbf{x})\|_2^2\right)}. \tag{3}$$

Learning proceeds by minimizing the negative log-likelihood $J(\boldsymbol{\theta}) = -\log p_{\boldsymbol{\theta}}(\mathbf{x} \in D_{\mathbf{x}})$ for the correct domain $D_{\mathbf{x}}$ from which $\mathbf{x}$ was sampled. This ensures that probability is high only if a point is closer to its own domain embedding and farther away from other domains, making the domain embedding *expressive*. Once training is complete, we obtain the final model parameters $\boldsymbol{\theta}^*$. For each domain $\widehat{D}_i(n)$, the *domain prototype* is constructed as:

$$\boldsymbol{\mu}(\widehat{D}_i(n)) := \frac{1}{n} \sum_{j=1}^{n} \Phi_{\mathcal{D}}(\mathbf{x}; \boldsymbol{\theta}^*). \tag{4}$$

Algorithm 1 Prototypical Training Pseudocode

```
PROTOTYPE TRAINING
Input. $N$ domains $(D_i)_{i=1}^N$ with $n$ samples each. $N_t$ : #domains sampled each batch, $N_s$: #support examples, $N_q$: #query examples per batch.
Output. Embedding neural network $\Phi_{\mathcal{D}} : \mathcal{X} \to \mathbb{R}^{d_{\mathcal{D}}}$.
  Initialize. weights $\boldsymbol{\theta}$ of $\Phi_{\mathcal{D}}$ randomly.
  for Round $t = 1$ to $T$ do
    $D_t \leftarrow$ RANDOMLY SAMPLE$(N, N_t)$. // sample $N_t$ domains
    for Domain $d$ in $D_t$ do
      $\mathcal{S}_d \leftarrow$ RANDOMLY SAMPLE$(D_d, N_s)$. // sample support points
      $\mathcal{S}_q \leftarrow$ RANDOMLY SAMPLE$(D_d, N_q)$. // sample query points
      $\widehat{\boldsymbol{\mu}}_d \leftarrow \frac{1}{N_s} \sum_{\mathbf{x} \in \mathcal{S}_d} \Phi_{\mathcal{D}}(\mathbf{x}; \boldsymbol{\theta})$. // compute prototype
    end for
    $J_{\boldsymbol{\theta}}(t) \leftarrow 0$.
    for Domain $d$ in $D_t$ do
      for $\mathbf{x} \in \mathcal{S}_q$ do
        Update Loss $J_{\boldsymbol{\theta}}(t)$ based on Equation 3.
      end for
    end for
    $\boldsymbol{\theta} \leftarrow$ SGD STEP$(J(t), \boldsymbol{\theta})$. // gradient descent step
  end for
```

```
PROTOTYPE COMPUTATION $(\Phi_{\mathcal{D}}, \mathcal{S})$
Input. Trained prototype network $\Phi_{\mathcal{D}}$ with weights $\boldsymbol{\theta}^*$.
Set $\mathcal{S}$ of points to create prototype.
Output. Prototype $\boldsymbol{\mu}(\mathcal{S})$.
  return $\boldsymbol{\mu}(\mathcal{S}) \leftarrow \frac{1}{|\mathcal{S}|} \sum_{\mathbf{x} \in \mathcal{S}} \Phi_{\mathcal{D}}(\mathbf{x}, \boldsymbol{\theta}^*)$.
```

Algorithm 2 Adaptive Training Pseudocode

```
ADAPTIVE TRAINING
Input. Prototype network $\Phi_{\mathcal{D}}$ with weights $\boldsymbol{\theta}^*$, Domains $(D_i)_{i=1}^N$.
# points $N_p$ to sample from each domain to create prototype.
Output. $F_{\text{ft}}$ with weights $\boldsymbol{\omega}_{\text{ft}}$, $F_{\text{mlp}}$ with weights $\boldsymbol{\omega}_{\text{mlp}}$.
  // compute prototypes for training domains
  for Domain $d$ from 1 to $N$ do
    $\mathcal{S}_d \leftarrow$ RANDOMLY SAMPLE$(D_d, N_p)$.
    $\boldsymbol{\mu}(\widehat{D}_d) \leftarrow$ PROTOTYPE COMPUTATION$(\Phi_{\mathcal{D}}, \mathcal{S}_d)$.
  end for
  Create augmented dataset $(\widetilde{D}_i)_{i=1}^n$ (Equation 5).
  for SGD round $t = 1$ to $T$ do
    Sample batch $(\mathbf{x}, \boldsymbol{\mu}, y)$ from augmented dataset.
    $\Phi_{\mathcal{X}}(\mathbf{x}) \leftarrow F_{\text{ft}}(\mathbf{x}; \boldsymbol{\omega}_{\text{ft}})$. // compute image features
    $\Phi(\mathbf{x}, \boldsymbol{\mu}) \leftarrow$ CONCAT$(\Phi_{\mathcal{X}}(\mathbf{x}), \boldsymbol{\mu})$. // concatenate features
    $\hat{y} \leftarrow F_{\text{mlp}}(\Phi(\mathbf{x}, \boldsymbol{\mu}); \boldsymbol{\omega}_{\text{mlp}})$. // compute predictions
    $J_{\boldsymbol{\omega}}(t) \leftarrow$ CROSSENTROPY$(\hat{y}, y)$. // compute loss
    $\boldsymbol{\omega}_{\text{ft}}, \boldsymbol{\omega}_{\text{mlp}} \leftarrow$ SGD STEP$(J_{\boldsymbol{\omega}}(t), \boldsymbol{\omega}_{\text{ft}}, \boldsymbol{\omega}_{\text{mlp}})$. // gradient descent
  end for
```

```
ADAPTIVE INFERENCE
Input. Trained networks $\Phi_{\mathcal{D}}, F_{\text{ft}}, F_{\text{mlp}}$.
A set $\mathcal{S}$ of $N$ points for prototype from domain $D$.
Output. Adaptive classifier for domain $D$.
  $\boldsymbol{\mu} \leftarrow$ PROTOTYPE COMPUTATION$(\Phi_{\mathcal{D}}, \mathcal{S})$.
  return $F(\mathbf{x}) = F_{\text{mlp}}(\text{CONCAT}(F_{\text{ft}}(\mathbf{x}), \boldsymbol{\mu}))$.
```

Prototype training and construction is completely unsupervised as long as we are provided the inputs partitioned by domains. We repeat *only* Equation 4 for each test domain as well (no retraining). See Algorithm 1 for details.

3.2. ERM on Augmented Inputs

Once we have the domain prototype for each training domain from the previous step, we create the *augmented* training set by appending the corresponding prototype to each input. We obtain the training set $\widetilde{\mathcal{S}}_{\text{trn}} = \cup_{i=1}^N \widetilde{D}_i$, where,

$$\widetilde{D}_i = \left(\mathbf{x}_{ij}, \boldsymbol{\mu}(\widehat{D}_i), y_{ij}\right)_{j=1}^n. \tag{5}$$

The second step is to then train the *domain-adaptive* classifier F over $\widetilde{\mathcal{S}}_{\text{trn}}$ using regular ERM training. F consists of two neural networks (Algorithm 2). The first network F_{ft} with weights $\boldsymbol{\omega}_{\text{ft}}$ takes $\mathbf{x}$ (image) as input and outputs a feature vector $\Phi_{\mathcal{X}}(\mathbf{x})$. We then concatenate the image feature $\Phi_{\mathcal{X}}$ with the domain embedding $\boldsymbol{\mu}(D)$ and pass through the second network F_{mlp} which predicts the class label y. We refer to this approach as DA-ERM . In Section 5.1, we present a few variants that combine the complimentary nature of adaptive and invariant approaches.

When testing, we first compute the domain prototypes for a test domain $\widehat{D}_T$ (Equation 4) using a small number of unlabeled examples[4], and then produce predictions via $F(\cdot, \widehat{D}_T)$. Alternatively, it is possible to decouple the prototype construction and inference procedures in the test phase (since we can use unsupervised samples obtained *a priori* to construct prototypes). Note that once the prototype $\boldsymbol{\mu}(\widehat{D}_T)$ is computed, we only need the adapted classifier $F(\cdot, \boldsymbol{\mu}(\widehat{D}_T))$ for inference, not the prototypical network.

3.3. Theoretical Guarantees

Here we provide an abridged summary of our theoretical contributions, and defer the complete results and proofs to the appendix. We first demonstrate that expressivity and consistency are theoretically motivated using the framework of kernel mean embeddings [34]. Our first result is a concentration bound on the n-point approximation error of $\boldsymbol{\mu}$.

Theorem 1. *($\boldsymbol{\mu}$ approximation, informal) For any domain D and feature Φ such that $\mathbb{E}_{\mathbf{x}\sim D}[\|\Phi(\mathbf{x})\|_2^2] \leq \sigma^2$, let the corresponding empirical dataset be $\widehat{D}(n)$, let $\boldsymbol{\mu}(D)$ be the true mean embedding and $\boldsymbol{\mu}(\widehat{D})$ be its approximation. Then with high probability, $\|\boldsymbol{\mu}(D) - \boldsymbol{\mu}(\widehat{D})\|_\infty \lesssim \widetilde{\mathcal{O}}(\sigma/\sqrt{n})$*[5].

This suggests that for low error, the variance σ must be small for each domain $D \in \mathcal{D}$, a property achieved by prototypes [39]. Next, we provide a generalization bound that controls the gap between the training error $\widehat{L}_{n,N}(F)$ and test error $L(F)$ (RHS of Eqn. 1) for F that is *adaptive*, i.e., a function of $\mathcal{X} \times \mathcal{D}$ and lies in the Hilbert space determined by some kernel κ. We build on the framework of [7] and extend their results to a more general class of kernel functions.

Theorem 2. *(error bound, informal) Let F be an adaptive classifier defined over $\mathcal{X} \times \mathcal{D}$ that lies in an RKHS $\mathcal{H}_\kappa$ such*

[4]We can also use a small sample of the total test set to create $\boldsymbol{\mu}(\widehat{D}_T)$.

[5]The $\widetilde{\mathcal{O}}$ notation hides polylogarithmic factors and failure probability.

that $\|f\|_{\mathcal{H}_\kappa} \leq R$*. Then, we have, with high probability that* $|L(F) - \widehat{L}_{n,N}(F)| \lesssim \widetilde{\mathcal{O}}\left(R \cdot \sigma((\log N)/n)^{\frac{1}{2}} + RN^{-\frac{1}{2}}\right)$.

Theorem 4 quantifies the error in terms of the feature variance σ, "norm" of function R and the number of samples n. In contrast to the results presented in prior work [7, 33, 21], we provide a tighter dependence on n that incorporates variance σ as well. See Appendix for more details.

4. Large Scale Benchmarks

There has been significant interest recently in domain generalization for vision problems [19]. However, we still lack large-scale benchmarks for rigorous evaluation. The largest dataset currently in use consists of only 7 domains and 5 classes [36], much smaller compared to real-world settings. Moreover, the model selection method of choice for this benchmark is *leave-one-domain-out* cross validation, which is infeasible in most large-scale applications with many domains. To address these issues, we introduce two large-scale domain generalization benchmarks.

4.1. Geo-YFCC

YFCC100M [42] is a large-scale dataset consisting of 100M images sourced from Flickr, along with tags submitted by users. Geotags (latitude and longitude) are also available for many of these images. We construct a dataset where each domain corresponds to a specific country.

Construction. We use geotags to partition images based on their country of origin. For the label space $\mathcal{Y}$, we consider the 4K categories from ImageNet-5K [10] not present in ILSVRC12 [37]. These categories are selected in order to eliminate biased prior knowledge from pre-training on ILSVRC12, which is the de-facto choice for large-scale pre-training. For each of the 4K labels, we select the corresponding images from YFCC100M based on a simple keyword-filtering of image tags. This provides us $1,261$ categories with at least 1 image present. Furthermore, each category is present in at least 5 countries. We group images by their country of origin and only retain countries that have at least 10K images. For any domain with more than 20K images, we randomly sub-sample to limit it to 20K images. Therefore, each *domain* (i.e., country) has anywhere between 10K-20K images, giving us a total of 1,147,059 images from 1,261 categories across 62 countries (domains), and each image is associated with a class label and country (domain). We refer to this resulting dataset as **Geo-YFCC**. To the best of our knowledge, the scale of this dataset in the number of images, labels and domains is orders of magnitude more than prior datasets.

Train-Test Split. We randomly partition the data in to 45 training, 7 validation and 15 test domains (by country). For each domain, we sample 3K points to create a per-domain test set and use the remaining points for training and validation. Since any image may have multiple labels, we convert it to a single-label dataset by replicating each such image and associating each separate copy for each label[6], expanding the total image set to 1,809,832. We plan on releasing the complete partitions and annotations upon publication.

Analysis: Geo-YFCC exhibits several properties of real-world data, such as long-tailed label distributions [29], considerable *covariate shift* [40] and *label shift* [3] across domains. Figure 2A shows the label distributions from 4 sample domains, where the categories are ordered based on the frequency of labels from the "USA" domain. We see that each domain exhibits long tails and labels shifts (i.e., the marginal distribution of labels is different in each domain [28]). For example, "Cambodia" has "Temple" as the most frequent category while it is a low-frequency concept for "USA". Figure 2B shows sample images for some categories from different domains, highlighting substantial *covariate* shift (i.e., shift in the marginal distribution of $\mathcal{X}$).

4.2. Long-Tailed ImageNet (LT-ImageNet)

We construct another dataset from the ImageNet-5K [10] benchmark by artificially introducing label shifts and long-tailed label distributions. The controlled settings in this dataset allow us to perform various ablations and understand our algorithm (Section 5.3). We subsample 500 categories from the 5K categories as the *base set* (excluding ILSVRC classes as earlier), from which we partition samples from each class into distinct train, val and test subsets. Each training domain is constructed as follows. We first select K head classes randomly from the base set. We then randomly sample A points from train samples of each head class and Af, $f < 1$ points from the remaining $500 - K$ tail classes to construct the train split of that domain, in order to mimic long-tailed phenomena. We use $A = 50$ and 300 to construct val and test splits for the domain from the corresponding splits of each class. We repeat this process N times to construct N training domains. Our overall training set has a non-uniform representation of classes in each domain, along with variation in long-tailed effects. We refer to the resulting dataset as **LT-ImageNet**.

To construct the val and test set of domains, we sample 10 domains with classes restricted to the set of all classes present in the training set (since we are not tackling zero-shot domain generalization), to ensure that the training data has at least 1 sample from each class (across all domains). We believe that these two large-scale datasets will help the community in properly understanding the effectiveness of existing and future domain generalization algorithms.

[6]This does not create any overlap between the train and test data as this duplication is done after the train-test split. As each image can belong only to one country, this ensures that no images are shared across domains.

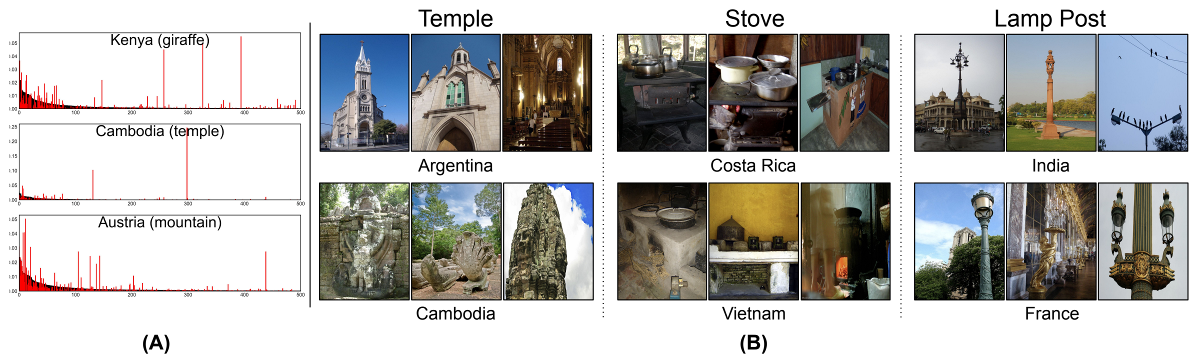


Figure 2: (A, left) Relative frequencies of categories from 3 different domains overlaid on the those of the USA, with most frequent category in parentheses. (B, right) Samples from 3 classes from different domains, highlighting covariate shift.

5. Experiments

We compare our algorithm with various recent algorithms on domain generalization problems both at the small-scale (i.e., fewer domains and classes per domain, and significant hyper-parameter tuning is possible) and at the real-world (large-scale) level. All benchmark algorithms are implemented from the DOMAINBED [19] suite with the hyper-parameter value ranges suggested therein.

Domain Embeddings (Section 3.1). We use a ResNet-50 [20] neural network pre-trained on ILSVRC12 [37] and introduce an additional fully-connected layer of dimension d_D on top of the `pool5` layer. The output of this layer is used to construct the domain embedding (Equation 4). Unless otherwise stated, $d_D = 1024$, and, the number of domains N_t used in N_t-way classification during each training step is set to 4. See Appendix for optimal hyper-parameters. To construct the domain embedding (Equation 4), we use at most 200 random points from the test set itself for small scale benchmarks in Section 5.1. For large-scale benchmarks, we leverage a subset of the samples from training domains and use a separate set of held-out points for test domains. For LT-ImageNet and Geo-YFCC, 300 and 5K points are used respectively.

DA-ERM (Section 3.2). The first neural network, F_{ft}, is again a ResNet-50 [20] network pre-trained on ILSVRC12 [37] with an additional fully-connected layer of dimension 1024 added to the `pool5` layer. This layer is then concatenated with the d_D dimensional domain embedding and passed through F_{mlp}. Unless otherwise stated, the MLP has two layers with hidden dimension $d_{\text{mlp}} = 1024$, and output dimension set to number of classes. We use the standard cross-entropy loss for training. See the Appendix for optimal training hyper-parameters for each dataset.

5.1. Small-Scale Benchmarks

We conduct small-scale benchmark comparisons carefully following the DOMAINBED suite [19], which contains 7 datasets in total. This paper focuses specifically on real-world domain generalization, and hence we select the 5 larger datasets (VLCS [14], PACS [23], Office-Home [43], Domain-Net [36] and Terra Incognita [5]) for our experiments, and discard the MNIST-based [22] datasets.

In this setting we select hyper-parameters by *leave one domain out* cross validation, the de facto technique for model selection for domain generalization, wherein we run N trials of each hyper-parameter setting, setting one domain for testing and the rest for training, and selecting the set of hyper-parameters that maximize validation accuracy over the training domains, averaged over all N trials. Performance under alternative techniques for model selection as well as the training details and hyper-parameter ranges for different algorithms can be found in the Appendix.

Domain Mixup. Since the small-scale datasets have typically 4 and at most 7 domains in total (1 of which is used as testing), the resulting training set (of domains) is not representative enough to learn a sufficiently discriminatory domain prototype. To alleviate this issue, we introduce *mixup* [48] in the prototype training step, where, for each batch of points from training domains, we construct *synthetic* domains by averaging points belonging to different domains and providing that as additional samples (belonging to a new domain). For example, during training, if we have a batch of points from each of the N_t training domains, we will create additional N_t synthetic batches where each synthetic batch is created by randomly averaging two batches. The averaging ratio for each synthetic domain is randomly chosen uniformly between $(0.2, 0.8)$.

Results. Results are summarized in Table 1, where each experiment is averaged over 8 independent trials (for our algorithm, this includes retraining of the domain prototypical network). DA-ERM provides an average improvement of **1.2%** in accuracy across all 5 datasets over the ERM baseline that simply combines data from all the training domains to train a model. The improvement is more sub-

Algorithm	VLCS [14]	PACS [23]	OffHome [43]	DNet [36]	TerraInc [5]	Average
ERM [19]	76.7 ± 0.9	83.2 ± 0.7	67.2 ± 0.5	41.1 ± 0.8	46.2 ± 0.3	62.9 ± 0.6
IRM [2]	76.9 ± 0.5	82.8 ± 0.5	67.0 ± 0.4	35.7 ± 1.9	43.8 ± 1.0	61.2 ± 0.8
DRO [38]	77.3 ± 0.2	83.3 ± 0.3	66.8 ± 0.1	33.0 ± 0.5	42.0 ± 0.6	60.4 ± 0.4
Mixup [44]	78.2 ± 0.6	83.9 ± 0.8	68.3 ± 0.4	40.2 ± 0.4	46.2 ± 0.5	63.3 ± 0.7
MLDG [24]	76.8 ± 0.4	82.6 ± 0.6	67.7 ± 0.3	42.2 ± 0.6	46.0 ± 0.4	63.0 ± 0.4
CORAL [41]	77.3 ± 0.3	83.3 ± 0.5	68.6 ± 0.1	42.1 ± 0.7	47.7 ± 0.3	63.8 ± 0.3
MMD [25]	76.1 ± 0.7	83.1 ± 0.7	67.3 ± 0.2	38.7 ± 0.5	45.8 ± 0.6	62.2 ± 0.6
C-DANN [26]	73.8 ± 1.1	81.4 ± 1.3	64.2 ± 0.5	39.5 ± 0.2	40.9 ± 0.7	60.1 ± 0.9
DA-ERM	78.0 ± 0.2	84.1 ± 0.5	67.9 ± 0.4	43.6 ± 0.3	47.3 ± 0.5	64.1 ± 0.8
DA-MMD	$\mathbf{78.6 \pm 0.5}$	84.4 ± 0.3	68.2 ± 0.1	43.3 ± 0.5	47.9 ± 0.4	64.5 ± 0.5
DA-CORAL	78.5 ± 0.4	$\mathbf{84.5 \pm 0.7}$	$\mathbf{68.9} \pm 0.4$	$\mathbf{43.9} \pm 0.3$	$\mathbf{48.1} \pm 0.3$	$\mathbf{64.7} \pm 0.6$

Table 1: Small-Scale Benchmark Comparisons on DOMAINBED.

Algorithm	LT-ImageNet Train	LT-ImageNet Test	Geo-YFCC Train	Geo-YFCC Test
	Top-1 / 5	Top-1 / 5	Top-1 / 5	Top-1 / 5
ERM [19]	70.9 / 90.3	53.7 / 77.5	**28.5 / 56.3**	22.4 / 48.2
CORAL [41]	71.1 / 89.3	53.5 / 76.2	25.5 / 51.2	21.8 / 46.4
MMD [25]	70.9 / 88.0	52.8 / 76.4	25.4 / 50.9	21.8 / 46.2
DA-ERM	**73.4 / 91.8**	**56.1 / 80.5**	28.3 / 55.8	**23.4 / 49.1**

Table 2: Large-Scale Comparisons.

stantial in Domain-Net [36], that has a larger number of domains. This can be attributed to the increased diversity in the training data, which allows us to construct embeddings that cover the space of domains better. In comparison to other domain-invariant approaches, DA-ERM either performs better or competitively with the best approach on each individual dataset.

To demonstrate the complimentary value of *adaptive* learning to domain-invariant methods, we modify the second step of our algorithm and consider two variants of the DA-ERM approach: DA-MMD and DA-CORAL, by replacing the cross-entropy in DA-ERM with the loss terms in domain-invariant approaches MMD [25] and CORAL [41] respectively[7] (see the appendix for details). Compared to CORAL, DA-CORAL provides a significant improvement of 0.9%, and DA-MMD achieves an impressive improvement of **2.3%** (62.2% vs. 64.5%) over MMD.

5.2. Large-Scale Benchmarks

We now present results on two large-scale benchmarks proposed in Section 4. We compare our approach with ERM, CORAL [41] (the best performing baseline in Section 5.1), and, MMD [25]. We were unable to replicate several algorithms at scale, such as DANN [15], Conditional-DANN [30] and MMLD [35]: these algorithms rely on adversarial training that requires careful hyper-parameter optimization, which we were unable to accomplish based on hyper-parameters suggested in DOMAINBED [19].

LT-ImageNet. We set N=50, K=100, A=350 and f=0.1 to construct the LT-ImageNet (50, 100, 350, 0.1) dataset and report top-1 and top-5 accuracies on the test splits of train and test domains in Table 2 (left). We observe that DA-ERM improves top-1 performance by 2.3% (71.1% vs. 73.4%) on train domains and 2.4% (53.7% vs 56.1%) on test domains. We show the effect of varying the dataset settings (N, K, A and f) in Section 5.3.

Geo-YFCC. Table 2 (right) shows comparisons on the test splits for train and test domains of the Geo-YFCC dataset. DA-ERM provides an absolute improvement of 1% in comparison to benchmark techniques on test domains. It must be noted that this problem is substantially more challenging, as we have 1,261 classes with significant domain differences between the countries (Section 4). We also observe that all other domain-invariant baselines were unable to obtain improvements over the simple ERM baseline. Figure 3A shows t-SNE [31] visualizations of prototypical embeddings for each domain, grouped by continent. Countries belonging to the same continent are mapped closer together, expressing the semantics captured by $\boldsymbol{\mu}$. Additionally, we note that our algorithm simultaneously improves performance on test domains while maintaining performance on training domains, unlike other techniques.

5.3. Ablations

5.3.1 Dataset Ablations

We consider the LT-ImageNet dataset with default settings mentioned in Section 5.2, unless otherwise stated.

Effect of diversity in training data. We first vary the number of available training domains (N), while sampling the same number of points from each domain. From Figure 3B, we observe that compared to other algorithms, the improvements in domain-aware training are larger, which we attribute to diversity in prototype training. Next, we repeat the same experiment, however, we subsample points from each domain such that the overall number of training points are the same, i.e., NA, remains constant. We see in Figure 3C that DA-ERM performs much better than other algorithms with increasing number of domains. Note that once $N > 100$, a slight dip is observed in our algorithm. This can be understood intuitively from Theorem 4, as it suggests such an inflection point when the total number of points (nN) is fixed due to the interaction of the two terms of the RHS. As the number of domains N increases, the first term decreases, while the second term increases.

Effect of Tail Index. In Figure 3D, we demonstrate the

[7] Auxiliary loss terms minimizing the differences between the domains is applied to the d_{mlp} dimensional hidden layer in the MLP F_{mlp}.

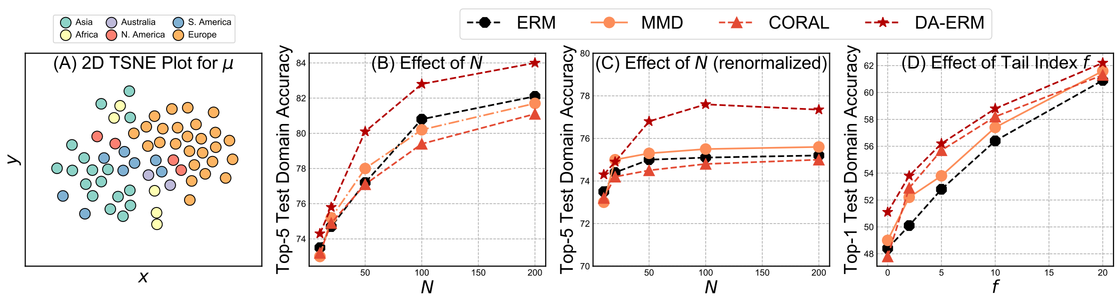


Figure 3: Ablation experiments: Fig. A (left) is a t-SNE visualization of domain embeddings. Figs. B (center left) and C (center right) show the effect of changing the number of domains, where the total number of training points increases linearly with the number of domains in (B) and is constant in (C). Fig. D (right) demonstrates the effect of the tail index f.

effect of varying f: the parameter f controls the extent of imbalance present in each domain, by adjusting the fraction of points available in the *tail classes* compared to the head classes. We see that when the dataset is severely imbalanced (i.e., $f < 5\%$), domain-adaptive training considerably outperforms existing techniques, with a top-1 performance gap of about **3.5%** on average comapred to naive ERM. As f increases and the imbalance becomes smaller, our method performs closer to ERM itself, with an average improvement of only **1.8%**. Since most real-world datasets are extremely long-tailed, we expect this to be a valuable feature of domain-adaptive learning in practice.

5.3.2 Algorithm Ablations

We perform several ablation studies on the domain embedding algorithm itself to verify the two properties of *consistency* and *expressivity* defined in Section 3.1.

Consistency. We varied the number of points used to construct the prototype on Geo-YFCC and observed that the performance declines only for fewer than 50 points. For other values up to 2K, performance remains virtually identical (see Appendix). This is desirable as in many settings we do not have access to many samples from new domains.

Expressivity. We measure the ability of the prototype to efficiently embed different domains distinctly within the embedding space. We evaluate the following two alternate approaches to construct domain embedding: (a) *Mean*: we select the ERM baseline model and simply compute the average `pool5` features for each domain; (b) *Softmax*: we train the domain embedding network via cross-entropy with domain IDs as labels instead of the prototypical loss. The results are summarized in Table 3. For reference, we also report results of naive ERM (*None*) and supplying incorrect embeddings (i.e., the embedding of some other random domain) during evaluation of test domains (*Random*). We observe that using random features degrades performance significantly, indicating that domain embeddings indeed play an important role in the final model. Mean features, while useful (i.e., perform better than naive ERM), are not as expressive as prototypical learning. Softmax features show mixed results by performing worse than Prototype on LT-ImageNet and slightly better on Geo-YFCC. We additionally study the effect of number of domains sampled per round in prototypical training, and observe no significant effect for values in the range $[3, 8]$.

Dataset	Top-1 Accuracy on Test Domains				
	None	Random	Mean	Softmax	Prototype
LT - ImageNet	76.9	75.3	77.9	78.8	80.5
Geo - YFCC	22.4	21.3	23.0	23.5	23.4

Table 3: Ablation of various domain embedding algorithms.

6. Conclusion

The problem of domain generalization lies at the heart of many machine learning applications. Research on the problem has largely been restricted to small-scale datasets that are constructed from artificially diverse sources that are not representative of real-world settings. For example, it is impractical to train a model on sketches if the target domain contains photographs: it is much easier in practice to curate a dataset of photographs instead. With this paper *work*, our goal is to provide an algorithm and evaluation procedure that is closer to real-world applications, and involves experimentation at scale. We presented an *adaptive* domain generalization framework, and conducted the first set of experiments at scale for this problem. Our results suggest that we need significant effort to scale to real-world problem settings, and with our contributions we expect to initiate research in this new direction.

Acknowledgements. We would like to sincerely thank Deepti Ghadiyaram, Filip Radenovic and Cheng-Yang Fu for their helpful feedback on our paper.

A. Theory

A.1. Motivation

The traditional objective of the domain generalization problem is to learn a function $f^* : \mathcal{X} \to \mathcal{Y}$ that minimizes the empirical risk over $\mathfrak{P}$, i.e., for some class of functions $\mathcal{F}$ and loss function $\ell : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}_+$,

$$f^* = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{D \sim \mathfrak{P}} \left[\mathbb{E}_{(\mathbf{x},y) \sim D} \left[\ell(f(\mathbf{x}), y) \right] \right]. \tag{6}$$

We denote the RHS in the above equation for any $f \in \mathcal{F}$ as the *expected risk* $L(f)$. Correspondingly, the optimal ERM solution $\hat{f}$ on the training data can be given as follows.

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{nN} \left(\sum_{\widehat{D} \in \mathcal{S}_{\text{trn}}} \sum_{(\mathbf{x},y) \in D} \ell(f(\mathbf{x}), y) \right). \tag{7}$$

We denote the RHS in the above equation for any $f \in \mathcal{F}$ as the *empirical risk* $\widehat{L}(f)$. Existing *invariant* approaches [6] build on exploiting the traditional decomposition of the empirical risk as a function of the variance of f across $\mathfrak{P}$.

$$\underbrace{\left| L(f) - \widehat{L}(f) \right|}_{\text{generalization gap}} \preccurlyeq \underbrace{\mathcal{O}\left(\frac{\text{Var}_{D \in \mathcal{S}_{\text{trn}}}(f)}{N} \right)^{\frac{1}{2}}}_{\text{inter-domain variance}}. \tag{8}$$

The particular approximation of the variance penalty (term 2 of the RHS) leads to the class of *invariant* domain generalization algorithms. In contrast, in this paper we focus on controlling the LHS directly by selecting a stronger function class $\mathcal{F}$ over the product space $\mathcal{D} \times \mathcal{X}$ (instead of $\mathcal{X}$)[8]. The key modeling choice we make is to learn a function $F : \mathcal{D} \times \mathcal{X} \to \mathcal{Y}$ that predicts $\hat{y} = F(D_X, \mathbf{x})$ for any $(\mathbf{x}, y) \in D$, where D_X is the marginal of $\mathcal{X}$ under D.

A.2. Background

Let $\mathcal{X} \subset \mathbb{R}^d$ be a *compact* input space, $\mathcal{Y} \in [-1, 1]$ to be the output space, and let $\mathfrak{P}_{\mathcal{X} \times \mathcal{Y}}$ denote the set of all probability distributions over the measurable space $(\mathcal{X} \times \mathcal{Y}, \Sigma)$, where Σ is the σ-algebra of subsets of $\mathcal{X} \times \mathcal{Y}$. Additionally, we assume there exists sets of probability distributions $\mathfrak{P}_{\mathcal{X}}$ and $\mathfrak{P}_{\mathcal{Y}|\mathcal{X}}$ such that for any sample $P_{XY} \in \mathfrak{P}_{\mathcal{X} \times \mathcal{Y}}$ there exist samples $P_X \in \mathfrak{P}_{\mathcal{X}}$ and $P_{Y|X} \in \mathfrak{P}_{\mathcal{Y}|\mathcal{X}}$ such that $P_{XY} = P_X \bullet P_{Y|X}$ (this characterization is applicable under suitable assumptions, see Sec. 3 of [7]). We assume that there exists a measure μ over $\mathfrak{P}_{\mathcal{X} \times \mathcal{Y}}$ and each *domain* $\mathcal{D}$ is an i.i.d. sample from $\mathcal{P}_{\mathcal{X} \times \mathcal{Y}}$, according to μ. The *training set* is generated as follows. We sample N realizations $P_{XY}^{(1)}, ..., P_{XY}^{(n)}$ from $\mathfrak{P}_{\mathcal{X} \times \mathcal{Y}}$ according to μ. For each domain, we then are provided $\mathcal{D}_i$, which is a set of n_i i.i.d. samples $(\mathbf{x}_j^{(i)}, y_j^{(i)})_{j \in [n_i]}$ sampled i.i.d. from $P_{XY}^{(i)}$.

Each *test* domain is sampled similarly to the training domain, where we first sample a probability distribution $P_{XY}^T \sim \mu$, and are then provided n_T samples $(\mathbf{x}_j^T, y_j^T)_{j \in [n_T]}$ from P_{XY}^T that forms a test domain. The key modeling assumption we make is to learn a decision function $f : \mathfrak{P}_{\mathcal{X}} \times \mathcal{X} \to \mathbb{R}$ that predicts $\hat{y} = f(\widehat{P}_X, \mathbf{x})$ for any $(\mathbf{x}, y) \in \mathcal{D}$ and $\widehat{P}_X$ is the associated empirical distribution of $\mathcal{D}$. For any loss function $\ell : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}_+$, then the empirical loss on any domain $\mathcal{D}_i$ is $\frac{1}{n_i} \sum_{(\mathbf{x},y) \in \mathcal{D}_i} \ell(f(\widehat{P}_X^{(i)}, \mathbf{x}), y)$. We can define the average generalization error over N test domains with n samples as,

$$\widehat{L}_N(f, n) := \frac{1}{N} \sum_{i \in [N]} \left[\frac{1}{n} \sum_{(\mathbf{x},y) \in \mathcal{D}_i} \ell(f(\widehat{P}_X^{(i)}, \mathbf{x}), y) \right]. \tag{9}$$

In the limit as the number of available domains $N \to \infty$, we obtain the expected n-sample generalization error as,

$$L(f, n) := \mathbb{E}_{P_{XY} \sim \mu, \mathcal{D} \sim (P_{XY}) \otimes n} \left[\frac{1}{n} \sum_{i=1}^{n} \ell(f(\widehat{P}_X, \mathbf{x}_i), y_i) \right]. \tag{10}$$

[8] While we do not investigate this in detail, our approach is compatible with *invariant* approaches, as we consider separate aspects of the problem.

The modeling assumption of f being a function of the empirical distribution $\widehat{P}_X$ introduces the primary difference between the typical notion of *training error*. As $n \to \infty$, we see that the empirical approximation $\widehat{P}_X \to P_X$. This provides us with a true generalization error, in the limit of $n \to \infty$ (i.e., we also have complete knowledge of P_X),

$$L(f, \infty) := \mathbb{E}_{P_{XY} \sim \mu, (\mathbf{x}, y) \sim P_{XY}} \left[\ell(f(P_X, \mathbf{x}), y) \right]. \tag{11}$$

An approach to this problem was proposed in Blanchard *et al.* [7] that utilizes product kernels. The authors consider a kernel κ over the product space $\mathfrak{P}_{\mathcal{X}} \times \mathcal{X}$ with associated RKHS $\mathcal{H}_\kappa$. They then select the function f_λ such that

$$f_\lambda = \underset{f \in \mathcal{H}_\kappa}{\arg\min} \frac{1}{N} \sum_{i=1}^{N} \frac{1}{n_i} \sum_{j=1}^{n_i} \ell(f(\widehat{P}_X^{(i)}, \mathbf{x}_{ij}), y_{ij}) + \lambda \|f\|^2_{\mathcal{H}_\kappa}. \tag{12}$$

The kernel κ is specified as a Lipschitz kernel on $\mathfrak{P}_{\mathcal{X}} \times \mathcal{X}$:

$$\kappa((P_X, \mathbf{x}), (P_{X'}, \mathbf{x}')) = f_\kappa(k_P(P_X, P_{X'}, k_X(\mathbf{x}, \mathbf{x}')). \tag{13}$$

Where K_P and K_X are kernels defined over $\mathfrak{P}_{\mathcal{X}}$ and $\mathcal{X}$ respectively, and f_κ is Lipschitz in both arguments, with constants L_P and L_X with respect to the first and second argument respectively. Moreover, K_P is defined with the use of yet another kernel $\mathfrak{K}$ and feature extractor ϕ:

$$k_P(P_X, P_{X'}) = \mathfrak{K}(\phi(P_X), \phi(P_{X'})). \tag{14}$$

Here, $\mathfrak{K}$ is a necessarily non-linear kernel and ϕ is the kernel mean embedding of P_X in the RKHS of a distinct kernel k'_X, i.e.,

$$P_X \to \phi(P_X) := \int_{\mathcal{X}} k'_X(\mathbf{x}, \cdot) dP_X(\mathbf{x}). \tag{15}$$

A.3. Kernel Assumptions

To obtain bounds on the generalization error, the kernels k_X, k'_X and $\mathfrak{K}$ have the assumptions of boundedness, i.e., $k_X(\cdot, \cdot) \leq B_k^2$, $k'_X(\cdot, \cdot) \leq B_{k'}^2$ and $\mathfrak{K}(\cdot, \cdot) \leq B_{\mathfrak{K}}^2$. Moreover, ϕ satisfies a Hölder condition of order $\alpha \in (0, 1]$ with constant $L_{\mathfrak{K}}$ in the RKHS ball of k'_X, i.e., $\forall \mathbf{x}, \mathbf{x}' \in \mathcal{B}(B_{k'}), \|\phi(\mathbf{x}) - \phi(\mathbf{x}')\| \leq L_{\mathfrak{K}} \|\mathbf{x} - \mathbf{x}'\|^\alpha$.

A.4. Consistency Bound (Theorem 1 of the Main Paper)

We first state the formal theorem.

Theorem 3. *Let D be a distribution over $\mathcal{X}$, and $\widehat{D}$ be the empirical distribution formed by n samples from $\mathcal{X}$ drawn according to D, and $\|\Phi_{\mathcal{D}}(\mathbf{x})\| \leq B_{k'}, \mathbb{E}_{\mathbf{x} \sim D}[\|\Phi_{\mathcal{D}}(\mathbf{x})\|_2^2] \leq \sigma^2$. Then, we have with probability at least $1 - \delta$ over the draw of the samples,*

$$\left\| \boldsymbol{\mu}(D) - \boldsymbol{\mu}(\widehat{D}) \right\|_\infty \leq \sqrt{\frac{8\sigma^2 \log(1/\delta)}{n} - \frac{1}{4}}.$$

Proof. We first note that $\boldsymbol{\mu}(D) = \mathbb{E}_{\mathbf{x} \sim D}[\Phi(\mathbf{x})]$, and each $\Phi(\mathbf{x})$ is a d_D-dimensional random vector. Furthermore, since the kernel k'_X is bounded, we have that $\|\Phi(\mathbf{x})\| \leq B_{k'}^2$ for any $\mathbf{x} \in \mathcal{X}$. Next, we present a vector-valued Bernstein inequality.

Lemma 1 (Vector-valued Bernstein bound []). *Let $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ be n finite-dimensional vectors such that $\mathbb{E}[\mathbf{x}_i] = 0$, $\|\mathbf{x}_i\| \leq \mu$ and $\mathbb{E}[\|\mathbf{x}_i\|^2] \leq \sigma^2$ for each $i \in [n]$. Then, for $0 < \epsilon < \sigma^2/\mu$,*

$$\mathbb{P}\left(\left\| \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \right\| \geq \epsilon \right) \leq \exp\left(-n \cdot \frac{\epsilon^2}{8\sigma^2} - \frac{1}{4} \right).$$

Now, we have, by the above result, for any $0 < \epsilon < \sigma^2/B_{k'}$

$$\mathbb{P}\left(\left\| \boldsymbol{\mu}(D) - \boldsymbol{\mu}(\widehat{D}) \right\| \geq \epsilon \right) \leq \exp\left(-n \cdot \frac{\epsilon^2}{8\sigma^2} - \frac{1}{4} \right). \tag{16}$$

This is obtained by noting that $\boldsymbol{\mu}(\widehat{D}) = \frac{1}{n} \sum_{i=1}^{n} \Phi(\mathbf{x}_i)$ and that $\boldsymbol{\mu}(D) = \mathbb{E}_{\mathbf{x} \sim D}[\Phi(\mathbf{x})]$. Setting the RHS as δ and rearranging terms gives us the final result. $\square$

A.5. Generalization Bound (Theorem 2 of Main Paper)

Theorem 4 (Uniform Risk Bound)**.** *Let $(P_{XY}^{(i)})_{i\in[N]}$ be N training domains sampled i.i.d. from μ, and let $S_i = (\mathbf{x}_{ij}, y_{ij})_{j\in[n]}$ be n-sample training sets for each domain. For any loss function ℓ that is bounded by B_ℓ and Lipschitz with constant L_ℓ, we have, with probability at least $1-\delta$ for any R:*

$$\sup_{f\in\mathcal{B}_\kappa(R)} |L(f,\infty) - \hat{L}_N(f,n)| \leq \mathcal{O}\left(R\left(\frac{\log(N/\delta)}{n}^{\frac{\alpha}{2}} + \sqrt{\frac{\log(1/\delta)}{N}}\right)\right).$$

Proof. We begin by decomposing the LHS.

$$\sup_{f\in\mathcal{B}_\kappa(R)} |L(f,\infty) - \hat{L}_N(f,n)| \leq \underbrace{\sup_{f\in\mathcal{B}_\kappa(R)} |L(f,\infty) - \hat{L}_N(f,\infty)|}_{A} + \underbrace{\sup_{f\in\mathcal{B}_\kappa(R)} |\hat{L}_N(f,\infty) - \hat{L}_N(f,n)|}_{B}. \tag{17}$$

We first control B. Note that the loss function is Lipschitz in terms of f, and therefore we have,

$$\sup_{f\in\mathcal{B}_\kappa(R)} |\hat{L}_N(f,\infty) - \hat{L}_N(f,n)| \leq \frac{L_\ell}{nN} \cdot \sup_{f\in\mathcal{B}_\kappa(R)} \left|\sum_{i=1}^N \sum_{j=1}^n f(\mathbf{x}_{ij}, \boldsymbol{\mu}(D_i)) - f(\mathbf{x}_{ij}, \boldsymbol{\mu}(\widehat{D}_i))\right| \tag{18}$$

$$\leq \frac{L_\ell}{nN} \sum_{i=1}^N \sum_{j=1}^n \sup_{f\in\mathcal{B}_\kappa(R)} \left|f(\mathbf{x}_{ij}, \boldsymbol{\mu}(D_i)) - f(\mathbf{x}_{ij}, \boldsymbol{\mu}(\widehat{D}_i))\right| \tag{19}$$

$$\leq \frac{L_\ell}{N} \sum_{i=1}^N \sup_{f\in\mathcal{B}_\kappa(R)} \left|f(\cdot, \boldsymbol{\mu}(D_i)) - f(\cdot, \boldsymbol{\mu}(\widehat{D}_i))\right|. \tag{20}$$

We will now bound $\sup_{f\in\mathcal{B}_\kappa(R)} \left|f(\cdot, \boldsymbol{\mu}(D_i)) - f(\cdot, \boldsymbol{\mu}(\widehat{D}_i))\right|$. Note that by the reproducing property,

$$\sup_{f\in\mathcal{B}_\kappa(R)} \left|f(\mathbf{x}, \boldsymbol{\mu}(D_i)) - f(\mathbf{x}, \boldsymbol{\mu}(\widehat{D}_i))\right| \leq \|f\|_\kappa \sup \left|f_\kappa(k_P(\boldsymbol{\mu}(D),\cdot), k_X(\mathbf{x},\cdot)) - f_\kappa(k_P(\boldsymbol{\mu}(\widehat{D}),\cdot), k_X(\mathbf{x},\cdot))\right| \tag{21}$$

$$\leq R \cdot \sup \left|f_\kappa(k_P(\boldsymbol{\mu}(D),\cdot), k_X(\mathbf{x},\cdot)) - f_\kappa(k_P(\boldsymbol{\mu}(\widehat{D}),\cdot), k_X(\mathbf{x},\cdot))\right| \quad \text{(Since } \|f\|_\kappa \leq R\text{)}$$

$$\leq RL_P \cdot \sup \left|\mathfrak{K}(\boldsymbol{\mu}(D),\cdot) - \mathfrak{K}(\boldsymbol{\mu}(\widehat{D}),\cdot)\right| \quad \text{(Since } f_\kappa \text{ is Lipschitz)}$$

$$\leq RL_P \cdot \sup \left\|\Phi_{\mathfrak{K}}(\boldsymbol{\mu}(D)) - \Phi_{\mathfrak{K}}(\boldsymbol{\mu}(\widehat{D}))\right\| \quad \text{(Triangle inequality)}$$

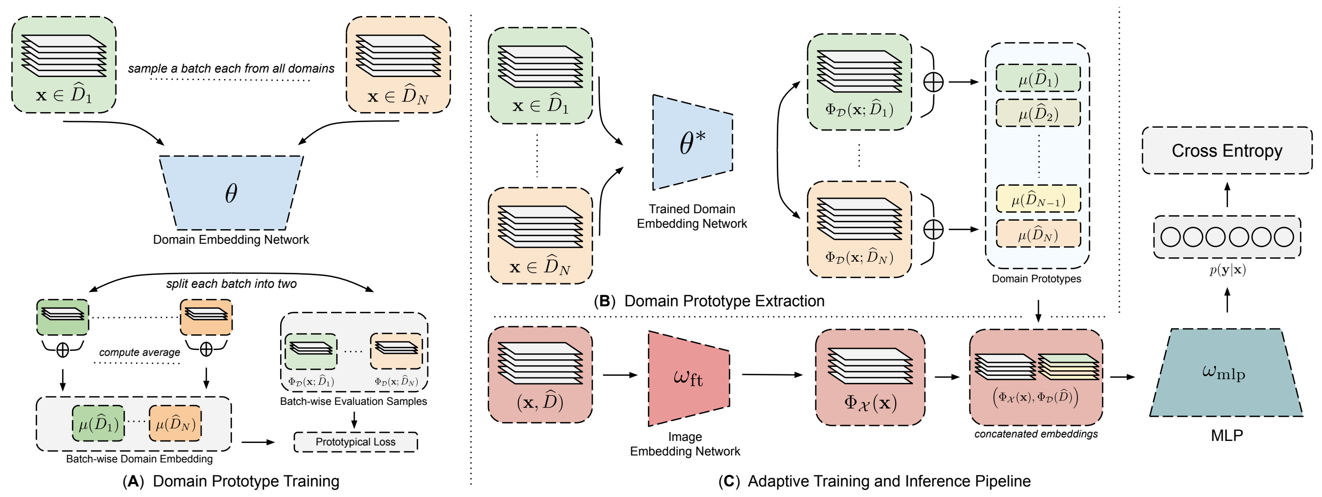
$$\leq RL_P \cdot \sup \left\|\boldsymbol{\mu}(D) - \boldsymbol{\mu}(\widehat{D})\right\|_\infty^\alpha \quad (\alpha\text{-Hölder assumption})$$

By Theorem 1, we can bound this term as well. Putting the results together and taking a union bound over all N domains, we have with probability at least $1-\delta$,

$$\sup_{f\in\mathcal{B}_\kappa(R)} |\hat{L}_N(f,\infty) - \hat{L}_N(f,n)| \leq \frac{L_\ell L_P}{N}\left(R\frac{8\sigma^2\log(N/\delta)}{n} - \frac{1}{4}\right)^{\alpha/2} \tag{22}$$

Control of the term B is done identically as Section 3.2 of the supplementary material in Blanchard *et al.* [7], with one key difference: in the control of term (IIa) (in their notation), we use the Lipschitz kernel instead of the product kernel, giving a constant $\sqrt{L_P}$ (instead of B_κ) in the bound of IIa. Combining the two steps gives the final stated result. □

Our proof admits identical dependences as Blanchard *et al.* [7], but the analysis differs in two key aspects: first, we use a Bernstein concentration to obtain a result in terms of the variance (Theorem 1 and term A of Theorem 2), which can potentially be tighter if the variance is low (see main paper and consistency). Next, we extend their result from product kernels to a general form of Lipschitz kernels, more suitable for deep-learning systems.



$\mathbf{x} \in \widehat{D}_1$
sample a batch each from all domains
$\mathbf{x} \in \widehat{D}_N$
θ
Domain Embedding Network
split each batch into two
compute average
$\Phi_{\mathcal{D}}(\mathbf{x}; \widehat{D}_1)$
$\Phi_{\mathcal{D}}(\mathbf{x}; \widehat{D}_N)$
Batch-wise Evaluation Samples
$\mu(\widehat{D}_1)$
$\mu(\widehat{D}_N)$
Batch-wise Domain Embedding
Prototypical Loss
(A) Domain Prototype Training
θ^*
Trained Domain Embedding Network
$\mu(\widehat{D}_2)$
$\mu(\widehat{D}_{N-1})$
Domain Prototypes
(B) Domain Prototype Extraction
Cross Entropy
$p(\mathbf{y}|\mathbf{x})$
$(\mathbf{x}, \widehat{D})$
ω_{ft}
Image Embedding Network
$\Phi_{\mathcal{X}}(\mathbf{x})$
$(\Phi_{\mathcal{X}}(\mathbf{x}), \Phi_{\mathcal{D}}(\widehat{D}))$
concatenated embeddings
ω_{mlp}
MLP
(C) Adaptive Training and Inference Pipeline

Figure 4: Illustrative figure for the two-step training process.

B. Two-Step Training Procedure

In our approach, we directly train a *domain-aware* neural network $g : \mathbb{R}^d \times \mathbb{R}^{d_D} \to [K]$. For any input image $\mathbf{x}$ from domain $\mathcal{D}$, g takes in the *augmented* input $(\mathbf{x}, \phi(\mathcal{D}; \boldsymbol{\theta}^*))$, which is composed of the input $\mathbf{x}$ and the corresponding domain prototype $\phi(\mathcal{D}; \boldsymbol{\theta}^*)$ obtained from the previous section, and predicts the class label $\hat{y}$ as output. The neural network architecture is described in Figure 4.

g is a composition of an image feature extractor (Φ_X) whose output is concatenated with the domain prototype $\Phi(\mathcal{D})$ and fed into a series of non-linear layers $(\mathfrak{K})$ to produce the final output. The domain-aware neural network parameters are denoted by $\boldsymbol{\omega}$. f therefore is parameterized by both $\boldsymbol{\omega}$ and $\boldsymbol{\theta}$ and is described as $f(\mathbf{x}, \mathcal{D}; \boldsymbol{\omega}, \boldsymbol{\theta}) = g(\mathbf{x}, \phi(\mathcal{D}; \boldsymbol{\theta}); \boldsymbol{\omega})$.

Remark 1. *It is possible to decouple the prototype construction and inference procedures in the test phase (since we can use unsupervised samples from a domain obtained a priori to construct prototypes). In this setting, we can use a distinct set of n_p points to construct the domain prototype for each test domain. We can see directly from Theorem 3 that for any Lipschitz loss function, the maximum disparity between classification using the optimal prototype (i.e., formed with knowledge of $\mathcal{D}$) and the n_p-sample approximation is (with high probability) at most $\widetilde{\mathcal{O}}\left(n_p^{-\alpha} + (\sigma_P/n_p)^{-\alpha/2}\right)$, which is small when σ_P is small, i.e., the prototype is consistent.*

C. Hyperparameters

C.1. Small-Scale Hyperparameters

We follow the setup for small-scale datasets that is identical to Gulrajani and Lopez-Paz [19] and use their default values, and the search distribution for each hyperparameter via random search. These values are summarized in Table 4.

Condition	Parameter	Default value	Random distribution
Basic hyperparameters	learning rate	0.00005	$10^{\text{Uniform}(-5,-3.5)}$
	batch size	32	$2^{\text{Uniform}(3,5.5)}$
	weight decay	0	$10^{\text{Uniform}(-6,-2)}$
C-DANN	lambda	1.0	$10^{\text{Uniform}(-2,2)}$
	generator learning rate	0.00005	$10^{\text{Uniform}(-5,-3.5)}$
	generator weight decay	0	$10^{\text{Uniform}(-6,-2)}$
	discriminator learning rate	0.00005	$10^{\text{Uniform}(-5,-3.5)}$
	discriminator weight decay	0	$10^{\text{Uniform}(-6,-2)}$
	discriminator steps	1	$2^{\text{Uniform}(0,3)}$
	gradient penalty	0	$10^{\text{Uniform}(-2,1)}$
	adam β_1	0.5	RandomChoice$([0, 0.5])$
IRM	lambda	100	$10^{\text{Uniform}(-1,5)}$
	iterations of penalty annealing	500	$10^{\text{Uniform}(0,4)}$
Mixup	alpha	0.2	$10^{\text{Uniform}(0,4)}$
DRO	eta	0.01	$10^{\text{Uniform}(-1,1)}$
MMD	gamma	1	$10^{\text{Uniform}(-1,1)}$
MLDG	beta	1	$10^{\text{Uniform}(-1,1)}$
all	dropout	0	RandomChoice$([0, 0.1, 0.5])$

Table 4: Hyperparameters for small-scale experiments.

C.1.1 Prototypical Network

In addition to these, the prototypical network optimal hyperparameters are as follows: We use a ResNet-50 architecture initialized with ILSVRC12 [10] weights, available in the PyTorch Model Zoo. We run 1000 iterations of prototypical training with $N_t = 4$ domains sampled each batch, and a batch size of 32 per domain. The learning rate is $1e-6$ and weight decay is $1e-5$ with 50% of the points in each batch used for classification and the rest for creating the prototype.

C.1.2 DA-CORAL and DA-MMD

The DA-CORAL and DA-MMD architectures are identical to DA-ERM. We additionally add the MMD and CORAL regularizers with $\gamma = 1$ for each in the final penultimate layer of the network (after the bottleneck, before the logits).

C.2. Large-Scale Hyperparameters

C.2.1 LT-ImageNet

DA-ERM Prototype Training. We did a small grid-search for learning rate (in the set $1e-1, 1e-2, 1e-3, 1e-4, 1e-5$) and the final parameters are: learning rate of 0.001 and weight decay of $1e-5$. We train with a batch size of 100 (per domain), over 8 GPUs and run the prototypical algorithm for 1300 iterations. We use 50% of the points for the support set per batch per domain and the remaining as test points.

Main Training for all methods. For all methods (MMD, CORAL, ERM, DA-ERM) we perform main training for 5 epochs of the training data with a batch-size of 100 (per domain), learning rate of 0.005 and weight-decay of $1e-5$ over a system of 8 GPUs. We additionally tune the value of the loss weight in the range $(0, 5)$ for MMD and CORAL. The reported results are with the loss weight ($\gamma = 1$).

C.2.2 Geo-YFCC

DA-ERM Prototype Training. We did a small grid-search for learning rate (in the set $1e-2, 1e-3, 1e-4, 1e-5$), weight-decay (in the set $1e-4, 1e-5, 1e-6$), and the final parameters are: learning rate of $2e-2$ and weight decay of $1e-5$. We train with a batch size of 80 (per domain), over 48 GPUs and run the prototypical algorithm for 1300 iterations. We use 80% of the points for the support set per batch per domain and the remaining as test points.

Main Training for all methods. We perform a grid search on number of epochs of training data in the range $(3, 6, 12, 25)$ and found that all methods performed best (overfit the least to training domains) at 6 epochs. For all methods (MMD, CORAL, ERM, DA-ERM) we perform main training with a batch-size of 80 (per domain), learning rate of 0.04 and weight-decay of $1e-5$ over a system of 64 GPUs. We additionally tune the value of the loss weight in the range $(0, 1.5)$ for MMD and CORAL. The reported results are with the loss weight ($\gamma = 1$).

D. Consistency Experiment

Dataset	Accuracy on Validation Set, top1/top5					
	50	100	250	500	1000	2000
DG - ImageNet	–	56.0/80.1	56.1/80.2	56.2/80.2	–	–
Geo - YFCC	23.7/49.0	23.3/49.1	23.4/49.3	23.6/49.3	23.4/49.1	23.4/49.2

Table 5: Number of points used to construct prototype.

Table 5 shows the effect of varying the number of data points used to consruct the domain prototypes for LT-ImageNet and Geo-YFCC datasets. We observe that performance remains similar till 50 points. This is desirable as in many settings we do not have access to many samples from new domains.

References

[1] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016. 2

[2] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019. 2, 7

[3] Kamyar Azizzadenesheli, Anqi Liu, Fanny Yang, and Animashree Anandkumar. Regularized learning for domain adaptation under label shifts. *arXiv preprint arXiv:1903.09734*, 2019. 2, 5

[4] Mahsa Baktashmotlagh, Mehrtash T Harandi, Brian C Lovell, and Mathieu Salzmann. Unsupervised domain adaptation by domain invariant projection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 769–776, 2013. 1, 2

[5] Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 456–473, 2018. 6, 7

[6] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010. 1, 9

[7] Gilles Blanchard, Gyemin Lee, and Clayton Scott. Generalizing from several related classification tasks to a new unlabeled sample. In *Advances in neural information processing systems*, pages 2178–2186, 2011. 1, 2, 4, 5, 9, 10, 11

[8] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010. 3

[9] Dong Cao, Xiangyu Zhu, Xingyu Huang, Jianzhu Guo, and Zhen Lei. Domain balancing: Face recognition on long-tailed domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5671–5679, 2020. 2

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 2, 5, 13

[11] Aniket Anand Deshmukh, Urun Dogan, and Clay Scott. Multi-task learning for contextual bandits. In *Advances in neural information processing systems*, pages 4848–4856, 2017. 2

[12] Aniket Anand Deshmukh, Yunwen Lei, Srinagesh Sharma, Urun Dogan, James W Cutler, and Clayton Scott. A generalization error bound for multi-class domain generalization. *arXiv preprint arXiv:1905.10392*, 2019. 2

[13] Abhimanyu Dubey and Alex Pentland. Kernel methods for cooperative multi-agent contextual bandits. *arXiv preprint arXiv:2008.06220*, 2020. 2

[14] Chen Fang, Ye Xu, and Daniel N Rockmore. Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1657–1664, 2013. 6, 7

[15] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016. 1, 2, 7

[16] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *Proceedings of the IEEE international conference on computer vision*, pages 2551–2559, 2015. 1

[17] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073. IEEE, 2012. 2

[18] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012. 2

[19] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020. 1, 2, 5, 6, 7, 12

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[21] Shoubo Hu, Kun Zhang, Zhitang Chen, and Laiwan Chan. Domain generalization via multidomain discriminant analysis. In *Uncertainty in Artificial Intelligence*, pages 292–302. PMLR, 2020. 5

[22] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*. 6

[23] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550, 2017. 2, 6, 7

[24] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Learning to generalize: Meta-learning for domain generalization. *arXiv preprint arXiv:1710.03463*, 2017. 2, 7

[25] Ya Li, Mingming Gong, Xinmei Tian, Tongliang Liu, and Dacheng Tao. Domain generalization via conditional invariant representation. *arXiv preprint arXiv:1807.08479*, 2018. 1, 2, 7

[26] Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep domain generalization via conditional invariant adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 624–639, 2018. 1, 2, 7

[27] Dahua Lin. Distribution-balanced loss for multi-label classification in long-tailed datasets. 2020. 2

[28] Zachary C Lipton, Yu-Xiang Wang, and Alex Smola. Detecting and correcting for label shift with black box predictors. *arXiv preprint arXiv:1802.03916*, 2018. 5

[29] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2537–2546, 2019. 5

[30] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pages 1640–1650, 2018. 7

[31] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 7

[32] Divyat Mahajan, Shruti Tople, and Amit Sharma. Domain generalization using causal matching. *arXiv preprint arXiv:2006.07500*, 2020. 2

[33] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pages 10–18, 2013. 1, 5

[34] Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, and Bernhard Schölkopf. Kernel mean embedding of distributions: A review and beyond. *arXiv preprint arXiv:1605.09522*, 2016. 3, 4

[35] Zhongyi Pei, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. Multi-adversarial domain adaptation. *arXiv preprint arXiv:1809.02176*, 2018. 2, 7

[36] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1406–1415, 2019. 5, 6, 7

[37] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 5, 6

[38] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019. 2, 7

[39] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017. 1, 3, 4

[40] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert MÃžller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(May):985–1005, 2007. 5

[41] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pages 443–450. Springer, 2016. 2, 7

[42] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016. 2, 5

[43] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *(IEEE) Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 6, 7

[44] Yufei Wang, Haoliang Li, and Alex C Kot. Heterogeneous domain generalization via domain mixup. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3622–3626. IEEE, 2020. 2, 7

[45] Liuyu Xiang, Guiguang Ding, and Jungong Han. Learning from multiple experts: Self-paced knowledge distillation for long-tailed classification. In *European Conference on Computer Vision*, pages 247–263. Springer, 2020. 2

[46] Minghao Xu, Jian Zhang, Bingbing Ni, Teng Li, Chengjie Wang, Qi Tian, and Wenjun Zhang. Adversarial domain adaptation with domain mixup. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6502–6509, 2020. 2

[47] Shen Yan, Huan Song, Nanxiang Li, Lincan Zou, and Liu Ren. Improve unsupervised domain adaptation with mixup training. *arXiv preprint arXiv:2001.00677*, 2020. 2

[48] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 6

[49] Xiao Zhang, Zhiyuan Fang, Yandong Wen, Zhifeng Li, and Yu Qiao. Range loss for deep face recognition with long-tailed training data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5409–5418, 2017. 2