Adaptive Methods for Real-World Domain Generalization : Appendix

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

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 $\ell : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}_+$,

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

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.

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{nN} \left(\sum_{\widehat{D} \in \mathcal{S}_{\text{trn}}} \sum_{(\mathbf{x},y)\in D} \ell(f(\mathbf{x}), y) \right). \tag{2}$$

We denote the RHS in the above equation for any $f \in \mathcal{F}$ as the *empirical risk* $\widehat{L}(f)$. Existing *invariant* approaches [1] build on exploiting the traditional decomposition of the empirical risk as a function of the variance of f across $\mathfrak{P}$.

$$\underbrace{\left| L(f) - \widehat{L}(f) \right|}_{\text{generalization gap}} \preccurlyeq \underbrace{\mathcal{O}\left(\frac{\text{Var}_{D \in \mathcal{S}_{\text{trn}}}(f)}{N} \right)^{\frac{1}{2}}}_{\text{inter-domain variance}}. \tag{3}$$

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 $\mathcal{X}$)[1]. 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 [2]). 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 $P^{(1)}_{XY}, ..., P^{(n)}_{XY}$ 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 $(\mathbf{x}^{(i)}_j, y^{(i)}_j)_{j\in[n_i]}$ sampled i.i.d. from $P^{(i)}_{XY}$.

Each *test* domain is sampled similarly to the training domain, where we first sample a probability distribution $P^T_{XY} \sim \mu$, and are then provided n_T samples $(\mathbf{x}^T_j, y^T_j)_{j\in[n_T]}$ from P^T_{XY} 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

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

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,

$$\widehat{L}_N(f, n) := \frac{1}{N} \sum_{i\in[N]} \left[\frac{1}{n} \sum_{(\mathbf{x},y)\in\mathcal{D}_i} \ell(f(\widehat{P}_X^{(i)}, \mathbf{x}), y) \right]. \tag{4}$$

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

$$L(f, n) := \mathbb{E}_{P_{XY}\sim\mu, \mathcal{D}\sim(P_{XY})^{\otimes n}} \left[\frac{1}{n} \sum_{i=1}^{n} \ell(f(\widehat{P}_X, \mathbf{x}_i), y_i) \right]. \tag{5}$$

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),

$$L(f, \infty) := \mathbb{E}_{P_{XY}\sim\mu,(\mathbf{x},y)\sim P_{XY}} \left[\ell(f(P_X, \mathbf{x}), y) \right]. \tag{6}$$

An approach to this problem was proposed in Blanchard *et al*. [2] 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

$$f_\lambda = \operatorname*{arg\,min}_{f\in\mathcal{H}_\kappa} \frac{1}{N} \sum_{i=1}^{N} \frac{1}{n_i} \sum_{j=1}^{n_i} \ell(f(\widehat{P}_X^{(i)}, \mathbf{x}_{ij}), y_{ij}) + \lambda \|f\|^2_{\mathcal{H}_\kappa}. \tag{7}$$

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

$$\kappa((P_X, \mathbf{x}), (P_{X'}, \mathbf{x}')) = f_\kappa(k_P(P_X, P_{X'}, k_X(\mathbf{x}, \mathbf{x}')). \tag{8}$$

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:

$$k_P(P_X, P_{X'}) = \mathfrak{K}(\phi(P_X), \phi(P_{X'})). \tag{9}$$

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.,

$$P_X \to \phi(P_X) := \int_\mathcal{X} k'_X(\mathbf{x}, \cdot) dP_X(\mathbf{x}). \tag{10}$$

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 1. *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 $\|\Phi(\mathbf{x})\| \leq B^2_{k'}$ 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 $0 < \epsilon < \sigma^2/B_{k'}$

$$\mathbb{P}\left(\left\|\boldsymbol{\mu}(D) - \boldsymbol{\mu}(\widehat{D})\right\| \geq \epsilon\right) \leq \exp\left(-n \cdot \frac{\epsilon^2}{8\sigma^2} - \frac{1}{4}\right). \tag{11}$$

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. □

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

Theorem 2 (Uniform Risk Bound). *Let $(P^{(i)}_{XY})_{i\in[N]}$ 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.

$$\sup_{f\in\mathcal{B}_\kappa(R)} |L(f,\infty) - \hat{L}_N(f,n)| \leq \underbrace{\sup_{f\in\mathcal{B}_\kappa(R)} |L(f,\infty) - \hat{L}_N(f,\infty)|}_{A} + \underbrace{\sup_{f\in\mathcal{B}_\kappa(R)} |\hat{L}_N(f,\infty) - \hat{L}_N(f,n)|}_{B}. \tag{12}$$

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

$$\sup_{f\in\mathcal{B}_\kappa(R)} |\hat{L}_N(f,\infty) - \hat{L}_N(f,n)| \leq \frac{L_\ell}{nN} \cdot \sup_{f\in\mathcal{B}_\kappa(R)} \left|\sum_{i=1}^{N}\sum_{j=1}^{n} f(\mathbf{x}_{ij}, \boldsymbol{\mu}(D_i)) - f(\mathbf{x}_{ij}, \boldsymbol{\mu}(\widehat{D}_i))\right| \tag{13}$$

$$\leq \frac{L_\ell}{nN}\sum_{i=1}^{N}\sum_{j=1}^{n} \sup_{f\in\mathcal{B}_\kappa(R)} \left|f(\mathbf{x}_{ij}, \boldsymbol{\mu}(D_i)) - f(\mathbf{x}_{ij}, \boldsymbol{\mu}(\widehat{D}_i))\right| \tag{14}$$

$$\leq \frac{L_\ell}{N}\sum_{i=1}^{N} \sup_{f\in\mathcal{B}_\kappa(R)} \left|f(\cdot, \boldsymbol{\mu}(D_i)) - f(\cdot, \boldsymbol{\mu}(\widehat{D}_i))\right|. \tag{15}$$

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,

$$\sup_{f\in\mathcal{B}_\kappa(R)} \left|f(\mathbf{x}, \boldsymbol{\mu}(D_i)) - f(\mathbf{x}, \boldsymbol{\mu}(\widehat{D}_i))\right| \leq \|f\|_\kappa \sup \left|f_\kappa(k_P(\boldsymbol{\mu}(D),\cdot), k_X(\mathbf{x},\cdot)) - f_\kappa(k_P(\boldsymbol{\mu}(\widehat{D}),\cdot), k_X(\mathbf{x},\cdot))\right| \tag{16}$$

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

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

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

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

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 $1-\delta$,

$$\sup_{f\in\mathcal{B}_\kappa(R)} |\hat{L}_N(f,\infty) - \hat{L}_N(f,n)| \leq \frac{L_\ell L_P}{N}\left(R\frac{8\sigma^2\log(N/\delta)}{n} - \frac{1}{4}\right)^{\alpha/2} \tag{17}$$

Control of the term B is done identically as Section 3.2 of the supplementary material in Blanchard *et al*. [2], 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*. [2], 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.

B. Two-Step Training Procedure

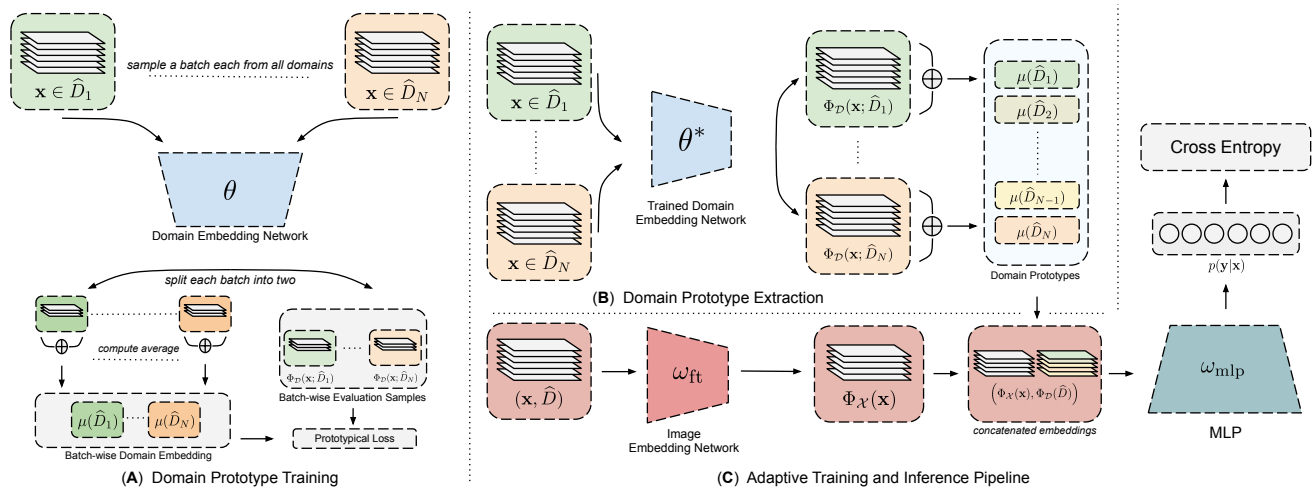

Figure 1: Illustrative figure for the two-step training process.

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 1.

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 1 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 [4] and use their default values, and the search distribution for each hyperparameter via random search. These values are summarized in Table 1.

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	$\text{RandomChoice}([0, 0.5])$
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	$\text{RandomChoice}([0, 0.1, 0.5])$

Table 1: 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 [3] 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 2: Number of points used to construct prototype.

Table 2 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] 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

[2] 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

[3] 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. 5

[4] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020. 4