# PROGRESSIVE DOMAIN GENERALIZATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

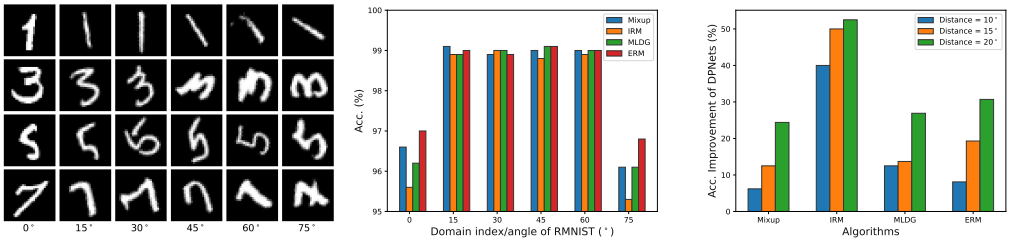
Domain generalization aims to learn a predictive model from multiple different but related source tasks that can generalize well to a target task without the need of accessing any target data. Existing domain generalization methods ignore the relation between tasks, implicitly assuming that all the tasks are sampled from a stationary environment. Therefore, they can fail when deployed in an evolving environment. To this end, we formulate and study the *progressive domain generalization* (PDG) scenario, which exploits not only the source data but also their evolving pattern to generate a model for the unseen task. Our theoretical result reveals the benefits of modeling the relation between two consecutive tasks by learning a globally consistent directional mapping function. In practice, our analysis also suggest solving the PDG problem in a meta-learning manner, which leads to *directional prototypical network*, the first method for the PDG problem. Empirical evaluation on both synthetic and real-world data sets validates the effectiveness of our approach.

## 1 INTRODUCTION

Modern machine learning techniques have achieved unprecedented success over the past decades in various areas. However, one fundamental limitation of most existing techniques is that a model trained on one data set cannot generalize well on another data set if it is sampled from a different distribution. Domain generalization (DG) aims to alleviate the prediction gap between the observed source domains and an *unseen* target domain by leveraging the knowledge extracted from multiple source tasks (Ganin & Lempitsky, 2015; Arjovsky et al., 2019; Li et al., 2018b).

Existing DG methods can be roughly categorized into three groups: data augmentation/generation, disentangled/domain-invariant feature learning, and meta-learning (Wang et al., 2021). One intrinsic problem with these methods is that they treat all the domains equally and ignore the relationship between them, implicitly assuming that they are all sampled from a stationary environment. However, in many real-world applications, the data are usually collected sequentially and the learning tasks can vary in an *evolving* manner. For example, geological exploration is often carried out periodically and the distribution of data collected can change from year to year due to environmental changes. Medical data are also often collected with age or other indicators as intervals, and there is an evolving trend in the data of different groups. As a more concrete example, Fig. 1(a) shows several instances from the rotated MNIST (RMNIST) data set, a widely used benchmark in the DG literature, where the digit images of each subsequent domain are rotated by $15^\circ$. Fig. 1(b) reports the generalization performances of several state-of-the-art DG algorithms on the data set, from which it can be clearly observed that the performances drop when deploying the models on outer domains (i.e., tasks of 0 and 75 degrees). The results indicate that the algorithms ignore the evolving pattern between the domains properly. As a consequence, they are good at "interpolation" but not at "extrapolation".

In this paper, we formulate this learning scenario as *progressive domain generalization* (PDG), which aims to capture and exploit the evolving patterns in the environment. In contrast to most existing DG methods, which produce models that are isotropic with respect to all the domains, PDG can generalize to a target domain along a specific direction by extracting and leveraging the relations between source tasks. Specifically, we develop a novel theoretical analysis that highlights the importance of modeling the relation between two consecutive tasks to extract the evolving pattern of the environment. Moreover, our analysis also suggests learning a globally consistent directional mapping function via meta-learning. Inspired the theoretical results, we slightly modify prototypical




(a) (b) (c)

Figure 1: (a) Evolving manner among RMNIST domains. (b) Accuracy of traditional DG methods on evolving domains. These methods cannot generalize well on outer domains ($0^\circ$ and $75^\circ$). (c) Comparison between the performance of our method and baselines on outer domains. The proposed method outperforms all the baselines.

networks (Snell et al., 2017) and propose *directional prototypical networks* (DPNets), a simple and efficient PDG algorithm that adapts to the environment shift and generalizes well on the evolving target domain. As a comparison, Fig. 1(c) shows the performance improvement of DPNets over the other algorithms on RMNIST data set. It can be observed that the performance gap between DPNets and the other baseline algorithms has widened as the domain distance increases. More details can be found in Section 4.

Here, we would like to emphasize the key difference between PDG and domain adaptation in evolving domains (Liu et al., 2020; Wang et al., 2020; Kim et al., 2020). While both learning paradigms aim to tackle the issue of evolving domain shifts, the latter still requires unlabeled instances from the target domain. In this sense, PDG is more challenging and existing theoretical and algorithmic results cannot be applied to this problem directly.

## 2 Related Work

**Domain Generalization (DG).** Domain generalization aims to train a model which generalizes on all domains. Existing DG methods can be classified into three categories. The first and most popular category is representation learning, which focus on learning a common representation across domains. It can be achieved by domain-invariant representation learning (Blanchard et al., 2011; Ghifary et al., 2015; Ganin & Lempitsky, 2015; Arjovsky et al., 2019) and feature disentanglement (Xu et al., 2014; Ilse et al., 2020). The former focuses on aligning latent features across domains, and the later tries to distill domain-shared features. Secondly, data manipulation can also empower the model with generalization capability. Data manipulating techniques include data augmentation (Yue et al., 2019; Shankar et al., 2018b), which usually extend the dataset by applying specific transformations on existing samples, and data generation (Rahman et al., 2019), which often applies neural networks to generate new samples. Nguyen et al. (2021) convert DG to an infinite-dimensional constrained statistical learning problem under a natural model of data generation. The theoretically grounded method proposed in Nguyen et al. (2021) leverages generating model among domains to learn domain-invariant representation. The last part are meta-learning. As a widely applicable method, meta-learning framework (Li et al., 2018b; Balaji et al., 2018; Li et al., 2019) is used to improve the generalizing capability by simulating the shift among domains. Apart from the above three categories, there are some other learning strategies that help to improve the generalization ability of the model. Mancini et al. (2018) tries to ensemble multiple models into a unified one which can generalize across domains. The DRO-based methods (Rahimian & Mehrotra, 2019) which aim to learn a model at worst-case distribution scenario also match the target of DG well. Besides, gradient operation (Huang et al., 2020b), self-supervision (Carlucci et al., 2019) and random forest (Ryu et al., 2019) are also exploited to improve generalizing capability. Different from the existing DG methods that focus on learning one unified model for all domains, our approach tries to train prediction model for the target domain specifically by leveraging the evolving pattern among domains.

**Evolving Domain Adaptation (EDA).** Many previous work in domain adaptation area notice the evolving pattern of domains and leverage it to improve the performance in different settings. Liu et al. (2020) proposes a meta-adaptation framework which enables the learner to adapt from one single source domain to continually evolving target domains without forgetting. Kim et al. (2020) tries to adapt to multiple target domains sequentially without forgetting, while the most important difference is that there is no assumption about the evolving pattern between these domains. Kumar et al. (2020) focuses on adapting from source domain to the target domain with large shifts by leveraging the unlabeled intermediate samples. Wang et al. (2020) combines the traditional adversarial adaptation strategy with a novel regression discriminator that models the encoding-conditioned domain index distribution. Chen & Chao (2021) investigate how to discover the sequence of intermediate domains without index information then adapt to the final target. The experimental results and theoretical analysis demonstrate the value of leveraging index information when working on evolving domains. These studies fully demonstrate that leveraging evolving pattern between domains is beneficial and worth more exploration. However, there are two significant limitation in previous works. The first one is the requirements for accessing unlabeled data in the target domain. The second one is that all previous theoretical results are base on the assumption that distance between sequential domains is small, which would become vacuous as the environment evolves. This is contrary to the fact that more domains provide more evolving information which can help to improve the performance. In this paper, the evolving information is leveraged without accessing any samples from the target domain. Also, our theoretical results are based on proposed $\lambda$-consistency, a intuitive and realistic measurement of evolving level in the environments.

## 3 Theoretical Analysis and Methodology

Let $\{\mathcal{D}_1(x,y), \mathcal{D}_2(x,y), ..., \mathcal{D}_m(x,y)\}$ be $m$ observed source domains sampled from an environment $\mathcal{E}$ where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ are, respectively, the data point and its corresponding label. The goal of DG is to learn a hypothesis $h \in \mathcal{H}$ so that it can have a low risk on an unseen but related target domain $\mathcal{D}_t$:

$$R_{\mathcal{D}_t}(h) \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}_t}[\ell(h(x), y)]$$

where $\ell : \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}_+$ is a non-negative loss function, and $\mathcal{H}$ is a hypothesis class that maps $\mathcal{X}$ to the set $\mathcal{Y}$. In the setting of traditional DG, as there is no relation exploited between the source domains and the target domain, most existing techniques essentially either "enlarge" the input space $\mathcal{X}$ along all possible directions (Volpi et al., 2018; Shankar et al., 2018a; Qiao et al., 2020) or learn a domain-invariant feature representation via domain alignment (Li et al., 2018c; Arjovsky et al., 2019; Zhao et al., 2020). In contrast, the objective of PDG is to generalize the model on $\mathcal{D}_t$ along a specific direction when there is underlying evolving pattern between the source domains and $\mathcal{D}_t = \mathcal{D}_{m+1}$.

### 3.1 Theoretical Motivations

In order to leverage the evolving pattern in $\mathcal{E}$, it is reasonable to assume that such a pattern can be captured by a globally consistent mapping function $g \in \mathcal{G} : \mathcal{D}^g_{i+1} \triangleq g(\mathcal{D}_i)$ in a way such that the *synthetic* domain $\mathcal{D}^g_{i+1}$ is close to $\mathcal{D}_{i+1}$ as much as possible, where $\mathcal{G}$ is the class of mapping functions. We first obtain the following bound of the risk on the target domain $\mathcal{D}_t$ with respect to $\mathcal{D}^g_t$, as shown in Lemma 1.

**Lemma 1.** *Let $\mathcal{D}^g_t(h) = g(\mathcal{D}_m)$ be the* synthetic *target domain, and suppose the loss function $\ell$ is bounded within an interval $G : G = \max(\ell) - \min(\ell)$. Then, for any $h \in \mathcal{H}$, its target risk $R_{\mathcal{D}_t}(h)$ can be upper bounded by:*

$$R_{\mathcal{D}_t}(h) \leq R_{\mathcal{D}^g_t}(h) + \frac{G}{\sqrt{2}}\sqrt{d_{JS}(\mathcal{D}^g_t||\mathcal{D}_t)},$$

*where $d_{JS}(\mathcal{D}^g_t||\mathcal{D}_t)$ is the Jensen-Shannon (JS) divergence between $\mathcal{D}^g_t$ and $\mathcal{D}_t$ (Fuglede & Topsoe, 2004).*

**Remark 1.** To achieve a low risk on $\mathcal{D}_t$, Lemma 1 suggests (1) learning $h$ and $g$ to minimize the risk over the synthetic domain $\mathcal{D}^g_t$ and (2) learning $g$ to minimize the JS divergence between $\mathcal{D}^g_t$ and $\mathcal{D}_t$. While in practice $R_{\mathcal{D}^g_t}(h)$ can be approximated by the empirical risk, Lemma 1 still cannot

provide any practical guidelines for learning $g$ since $\mathcal{D}_t$ is unavailable. Moreover, note that $\mathcal{D}_t^g$ can be replaced by $g(\mathcal{D}_i)$ for any other source domain $i$ in $\mathcal{E}$ and the bound still holds. Thus, Lemma 1 does not provide any theoretical insight into how to discover and leverage the evolving pattern in $\mathcal{E}$.

Intuitively, capturing the evolving pattern in $\mathcal{E}$ is hopeless if it varies arbitrarily. On the other hand, if the underlying pattern is consistent over domains, it is reasonable to assume that there exists $g^* \in \mathcal{G}$ would perform consistently well over all the domain pairs. For example, given numbers $100, 202, 301$, one would expect that the numbers increase by around 100 and the next number will be around $400$, but it is challenging to guess the number if the first three numbers are $-20, 1300, 4$. To formulate the this intuition, we first introduce the notion of *consistency* of an environment $\mathcal{E}$ below.

**Definition 1** (Consistency). *Let $g^* = \arg\min_{g\in\mathcal{G}} \max_{\mathcal{D}_i\in\mathcal{E}} d_{JS}(\mathcal{D}_i^g||\mathcal{D}_i)$ be the ideal mapping function in the worst-case domain. Then, an evolving environment $\mathcal{E}$ is* $\lambda$-consistent *if the following holds:*

$$|d_{JS}(\mathcal{D}_i^{g^*}||\mathcal{D}_i) - d_{JS}(\mathcal{D}_j^{g^*}||\mathcal{D}_j)| \leq \lambda, \qquad \forall \mathcal{D}_i, \mathcal{D}_j \in \mathcal{E}.$$

Note that $\lambda$ does not characterize how fast $\mathcal{E}$ evolves, but if there exists a global mapping function that can model the evolving pattern consistently well in $\mathcal{E}$.

Given the definition of consistency, we can bound the target risk in terms of $d_{JS}(\mathcal{D}_i^g||\mathcal{D}_i)$ in the source domains.

**Theorem 1.** *Let $\{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_m\}$ be $m$ observed source domains sampled sequentially from an evolving environment $\mathcal{E}$, and $\mathcal{D}_t$ be the next unseen target domain: $\mathcal{D}_t = \mathcal{D}_{m+1}$. Then, if $\mathcal{E}$ is $\lambda$-consistent, we have*

$$R_{\mathcal{D}_t}(h) \leq R_{\mathcal{D}_t^{g^*}}(h) + \frac{G}{\sqrt{2(m-1)}}\left(\sqrt{\sum_{i=2}^{m} d_{JS}(\mathcal{D}_i^{g^*}||\mathcal{D}_i)} + \sqrt{(m-1)\lambda}\right).$$

**Remark 2.** **(1)** Theorem 1 highlights the role of the mapping function and $\lambda$-consistency in PDG. Given $g^*$, the target risk $R_{\mathcal{D}_t}(h)$ can be upper bounded by in terms of loss on the synthetic target domain $R_{\mathcal{D}_t^{g^*}}(h)$, $\lambda$, and the JS divergence between $\mathcal{D}_i$ and $\mathcal{D}_i^{g^*}$ in all *observed* source domains. When $g^*$ can properly capture the evolving pattern of $\mathcal{E}$, we can train the classifier $h$ over the synthetic domain $\mathcal{D}_t^{g^*}$ generated from $\mathcal{D}_m$ and can still expect a low risk on $\mathcal{D}_t$. **(2)** $\lambda$ is *unobservable* and is determined by $\mathcal{E}$ and $\mathcal{G}$. Intuitively, a small $\lambda$ suggests high *predictability* of $\mathcal{E}$, which indicates that it is easier to predict the target domain $\mathcal{D}_t$. On the other hand, a large $\lambda$ indicates that there does not exist a global mapping function that captures the evolving pattern consistently well over domains. Consequently, generalization to the target domain could be challenging and we cannot expect to learn a good hypothesis $h$ on $\mathcal{D}_t$. **(3)** In practice, $g^*$ is not given, but can be learned by minimizing $d_{JS}(\mathcal{D}_i^g||\mathcal{D}_i)$ in source domains. Besides, aligning $D_i^g$ and $\mathcal{D}_i$ is usually achieved by *representation learning:* that is, learning $g : \mathcal{X} \to \mathcal{Z}$ to minimize $d_{JS}(\mathcal{D}_i^g||\mathcal{D}_i), \forall z \in \mathcal{Z}$.

In addition, we can decompose $\mathcal{D}(x, y)$ into marginal and conditional distributions to motivate more practical PDG algorithms. For example, when it is decomposed into class prior $\mathcal{D}(y)$ and semantic conditional distribution $\mathcal{D}(x|y)$, we have the following Corollary.

**Corollary 1.** *Following the assumptions of Theorem 1, the target risk can be bounded by*

$$R_{\mathcal{D}_t}(h) \leq R_{\mathcal{D}_t^{g^*}}(h) + \frac{G}{\sqrt{2(m-1)}}\Bigg(\underbrace{\sqrt{\sum_{i=2}^{m} d_{JS}(\mathcal{D}_i^{g^*}(y)||\mathcal{D}_i(y))}}_{\mathbf{I}} + \sqrt{(m-1)\lambda}$$
$$+ \underbrace{\sqrt{\sum_{i=2}^{m} \mathbb{E}_{y\sim\mathcal{D}_i^{g^*}(y)} d_{JS}(\mathcal{D}_i^{g^*}(x|y)||\mathcal{D}_i(x|y))}}_{\mathbf{II}} + \underbrace{\sqrt{\sum_{i=2}^{m} \mathbb{E}_{y\sim\mathcal{D}_i(y)} d_{JS}(\mathcal{D}_i^{g^*}(x|y)||\mathcal{D}_i(x|y))}}_{\mathbf{III}}\Bigg).$$

**Remark 3.** To generalize well to $\mathcal{D}_t$, Corollary 1 suggests that a good mapping function should capture both label shifts (term I) and semantic conditional distribution shifts (terms II & III). If we

further assume that the label distribution does not evolve over domains[1], we will have I = 0 and II = III, and the upper bound can be simplified as

$$R_{\mathcal{D}_t}(h) \leq R_{\mathcal{D}_t^{g^*}}(h) + \frac{G}{\sqrt{2(m-1)}}\left(2\sqrt{\sum_{i=2}^{m}\mathbb{E}_{y\sim\mathcal{D}_i(y)}d_{JS}(\mathcal{D}_i^{g^*}(x|y)||\mathcal{D}_i(x|y))} + \sqrt{(m-1)\lambda}\right).$$

Finally, we note two key theoretical differences between PDG and previous studies of DA in evolving environments (Ben-David et al., 2010; Liu et al., 2020). (1) In DA, the target risk is bounded in terms of source and target domains (e.g., $\mathcal{H}\Delta\mathcal{H}$-divergence), while our analysis relies on the distance between synthetic and real domains. (2) DA theories are built upon the assumption that there exists an ideal joint hypothesis that achieves a low combined error on both domains, while our assumption is the $\lambda$-consistency of $\mathcal{E}$. These differences eventually lead to fundamentally different guidelines for PDG, as shown in Section 3.2.

### 3.2 Practical Implementations

Our analysis reveals several general strategies to follow when designing an algorithm for PDG.

(i) Learning the mapping function $g$ to capture the evolving pattern by minimizing the distance between the distributions of synthetic and real domains.

(ii) Learning $g$ and $h$ to minimize the risk on the synthetic target domain $\mathcal{D}_t^g$.

(iii) Note that $\mathcal{D}_{i+1}^g = g(\mathcal{D}_i)$ is produced from $\mathcal{D}_i$, but its quality is evaluated on $\mathcal{D}_{i+1}$. Thus, minimizing $d_{JS}(\mathcal{D}_i^g||\mathcal{D}_i)$ naturally suggests a **meta-learning** strategy for learning $g$.

In practice, mapping the samples from $\mathcal{D}_i$ to $\mathcal{D}_{i+1}$ is not necessarily performed in the original data space since our ultimate goal is making predictions rather than generating instances themselves. Thus, we minimize the distance between $\mathcal{D}_{i+1}^g$ and $\mathcal{D}_{i+1}$ in a representation space. Based on these ideas, we slightly modify prototypical networks (Snell et al., 2017) and propose *directional prototypical networks* (DPNets) for PDG. Specifically, the mapping function $g$ of DPNets consists of two different embedding functions: $f_\phi$ for $\mathcal{D}_i$ and $f_\psi$ for $\mathcal{D}_{i+1}$, where $\phi$ and $\psi$ are learnable parameters. The key idea of DPNets is to learn $g = \{f_\phi, f_\psi\}$ to capture the evolving pattern of $\mathcal{E}$ by estimating the prototypes of $f_\psi(\mathcal{D}_{i+1})$ using $f_\phi(\mathcal{D}_i)$. As each prototype can be viewed as the centroid of instances of each class, which is an approximation of the semantic conditional distribution of each class (Xie et al., 2018; Shui et al., 2021a), DPNets essentially minimizes the distance between $\mathcal{D}_{i+1}^g(x|y)$ and $\mathcal{D}_{i+1}(x|y)$, as suggested by Corollary 1 Remark 3.

Let $\mathcal{S}_i = \{(x_n^i, y_n^i)\}_{n=1}^{N_i}$ be the data set of size $N_i$ sampled from $\mathcal{D}_i$, and $\mathcal{S}_i^k$ be the subset of $\mathcal{S}_i$ with class $k \in \{1, ..., K\}$, where $K$ is the total number of classes. Then, the prototype of domain $i$ is the mean vector of the *support* instances belonging to $\mathcal{S}_i^k$:

$$c_i^k = \frac{1}{|\mathcal{S}_i^k|}\sum_{(x_n^i, y_n^i)\in\mathcal{S}_i^k} f_\phi(x_n^i)$$

In (Snell et al., 2017), the prototype is used to produce a distribution $\mathcal{D}(y = k|x)$ to make a prediction for a *query* instance $x$ in the context of few-shot learning, and the support and query instances are sampled from the same domain. By contrast, in DPNets, the prototypes are computed from the support set $\mathcal{S}_i$ through the embedding function $f_\phi$, but the query instances are from $\mathcal{S}_{i+1}$ and are passed through another function $f_\psi$. Then, the predictive distribution for $\mathcal{D}_{i+i}$ is given by

$$\mathcal{D}_{\phi,\psi}(y^{i+1} = k|x^{i+1}) = \frac{\exp(-d(f_\psi(x^{i+1}), c_i^k))}{\sum_{k'=1}^{K}\exp(-d(f_\psi(x^{i+1}), c_i^{k'})}, \tag{1}$$

where $d : \mathcal{Z} \times \mathcal{Z} \to [0, +\infty)$ is a distance function of embedding space, and we adopt squared Euclidean distance in our implementation, as suggested in (Snell et al., 2017). During the training

[1]When label shifts exist, term I can be minimized by reweighting/resampling the instances according to the class ratios between domains (Shui et al., 2021a).

**Algorithm 1** The loss computation for DPNets (one episode)

**Input:** $\{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_m\}$: $m$ data sets from consecutive domains. $N_B$: the number of support and query instances for each class. RANDOMSAMPLE$(\mathcal{S}, N)$: a set of $N$ instances sampled uniformly from the set $\mathcal{S}$ without replacement.
**Output:** The loss $J_{\phi,\psi}$ for a randomly generated training episode.
$t \leftarrow$ RANDOMSAMPLE$(\{1, ..., m\})$
**for** $k$ in $\{1, ..., K\}$ **do**
  $S^k \leftarrow$ RANDOMSAMPLE$(\mathcal{S}_i^k, N_B)$
  $Q \leftarrow$ RANDOMSAMPLE$(\mathcal{S}_{i+1}^k, N_B)$
  $c^k = \frac{1}{|S^k|} \sum_{(x_j, y_j) \in S^k} f_\phi(x_j)$ ▷ Compute prototypes
**end for**
$J(\phi, \psi) \leftarrow 0$
**for** $k$ in $\{1, ..., K\}$ **do**
  **for** $(x, y)$ in $Q$ **do**
    $J_{\phi,\psi} \leftarrow J_{\phi,\psi} + \frac{1}{KN_B}[d(f_\psi(x), c^k) + \log \sum_{k'} \exp(-d(f_\psi(x), c^k)]$
  **end for**
**end for**

stage, at each step, we randomly choose the data sets $\mathcal{S}_i, \mathcal{S}_{i+1}$ from two consecutive domains as support and query sets, respectively. Then, we sample $N_B$ samples from each class $k$ in $\mathcal{S}_i$, which is used to compute prototype $c_i^k$ for the query data in $\mathcal{S}_{i+1}$ . Model optimization proceeds by minimizing the negative log-probability: $J_{\phi,\psi} = -\log \mathcal{D}_{\phi,\psi}(y^{i+1} = k|x^{i+1})$. The pseudocode to compute $J_{\phi,\psi}$ for a training episode is shown in Algorithm 1. In the testing stage, we pass the instances from $\mathcal{S}_m$ and $\mathcal{S}_t$ through $f_\phi$ and $f_\psi$ respectively as support and query sets, and then make predictions for the instances in $\mathcal{D}_t$ using Eq. (1).

# 4 EXPERIMENTS

## 4.1 EXPERIMENTAL SETUP

We evaluate our algorithm on an extensive collection of five data sets, including **two synthetic data sets** (Envolving Circle (Wang et al., 2020) and Rotated Plate and **three real-world data sets** (RMNIST (Ghifary et al., 2015), Portrait (Kumar et al., 2020; Chen & Chao, 2021), and Cover Type (Kumar et al., 2020)).

(1) **Evolving Circle (EvolCircle, Fig. 2)** consists of 30 evolving domains, where the instances are generated from 30 2D Gaussian distributions with the same variances but different centers uniformly distributed on a half-circle. (2) **Rotated Plate (RPlate, Fig. 3)** consists of 30 domains, where the instances of each domain is generated by the same Gaussian distribution but the decision boundary rotates from $0^\circ$ to $348^\circ$ with an interval of $12^\circ$. (3) **Rotated MNIST (RMNIST)** We randomly select only 2400 instances in raw MNIST dataset and split them into 12 domains equally. Then we apply the rotations with degree of $\theta = \{0^\circ, 10^\circ, ..., 110^\circ\}$ on each domain respectively. The amount of samples in each domain is only 200, which makes this task more challenging. (4) **Portrait** This task is to classify gender based on the photos of high school seniors across different decades (Ginosar et al., 2015). We divided the dataset into 12 domains by year. (5) **Cover Type** data set aims to predict cover type (the predominant kind of tree cover) from 54 strictly cartographic variables. To generate evolving domains, we sort the samples by the ascending order of the distance to the water body, as proposed in (Kumar et al., 2020). Then we equally divided the data set into 10 domains by distance. (5) **FMoW** A large satellite image dataset with target detection and classification tasks (Christie et al., 2018) . We select 5 common classes to compose a classfication task. The dataset is divided into 19 domains by time.

We compared the proposed method with the following baselines: (1) **GroupDRO** (Sagawa et al., 2019); (2) **MLDG** (Li et al., 2018a); (3) **MMD** (Li et al., 2018c); (4) **SagNet** (Nam et al., 2021); (5) **VREx** (Krueger et al., 2021); (6) **SD** (Pezeshki et al., 2020); (7) **IRM** (Arjovsky et al., 2019); (8) **Mixup** (Yan et al., 2020); (9) **CORAL** (Sun & Saenko, 2016); (10) **MTL** (Blanchard et al., 2021); (11) **RSC** (Huang et al., 2020a); (12) **DIRL** (Nguyen et al., 2021); (13) **Original Proto-**

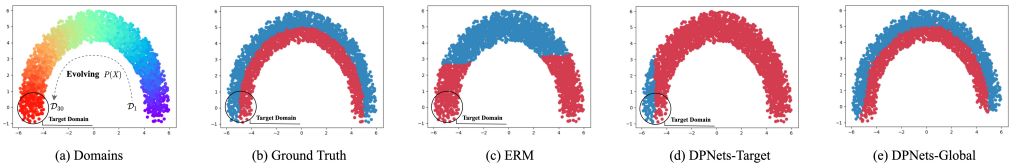



Figure 2: Visualization of the EvolCircle data set. (a) 30 domains indexed by different colors, where the left bottom one is target domain. (b) Positive and negative instances are denoted by red and blue dots respectively. (c) The decision boundaries learned by ERM. (d) Decision boundaries of last model on all domains. (e) Decision boundaries of models in each domain. (the results of first domain are missing due to the lack of prototypes.

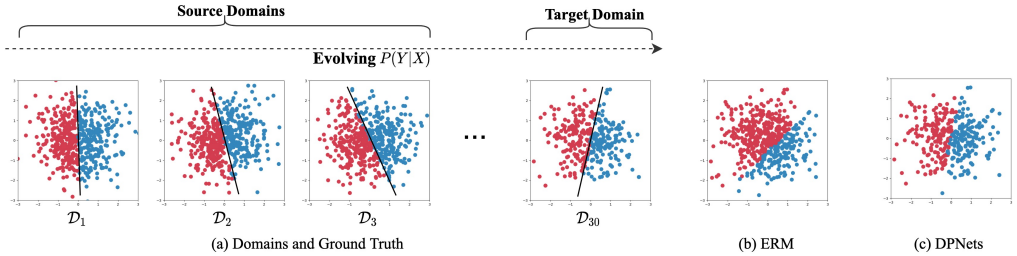



Figure 3: Visualization of the RPlate data set. (a) The true decision boundaries evolves over domains. (b) & (c) The decision boundaries learned by ERM and DPNets on the target domain.

Table 1: Comparison of accuracy (%) among different methods.

| Algorithm | EvolCircle | RPlate | RMNIST | Portrait | Cover Type | FMoW | Average |
|---|---|---|---|---|---|---|---|
| GroupDRO | 75.5 ± 1.0 | 70.0 ± 4.9 | 76.5 ± 0.2 | 94.8 ± 0.1 | 66.4 ± 0.5 | 57.3 ± 0.1 | 73.4 |
| MLDG | 91.5 ± 2.0 | 66.9 ± 1.8 | 75.0 ± 0.3 | 66.2 ± 1.7 | 68.4 ± 0.7 | 43.8 ± 0.0 | 68.6 |
| MMD | 86.7 ± 5.7 | 59.9 ± 1.4 | 35.4 ± 0.0 | 95.4 ± 0.1 | 69.8 ± 0.4 | 60.0 ± 0.0 | 67.8 |
| SagNet | 78.7 ± 3.2 | 63.8 ± 2.9 | 79.4 ± 0.1 | 95.3 ± 0.1 | 65.3 ± 2.2 | 56.2 ± 0.1 | 73.1 |
| VREx | 82.9 ± 6.6 | 61.1 ± 2.6 | 79.4 ± 0.1 | 94.3 ± 0.2 | 66.0 ± 0.9 | 61.2 ± 0.0 | 73.3 |
| SD | 81.7 ± 4.3 | 65.3 ± 1.4 | 78.8 ± 0.1 | 95.1 ± 0.2 | 69.1 ± 0.9 | 55.2 ± 0.0 | 74.2 |
| IRM | 86.2 ± 3.0 | 67.2 ± 2.1 | 47.5 ± 0.4 | 94.4 ± 0.3 | 66.0 ± 1.0 | 58.8 ± 0.0 | 70.0 |
| Mixup | 91.5 ± 2.6 | 66.8 ± 1.8 | 81.3 ± 0.2 | **96.4 ± 0.2** | 69.7 ± 0.6 | 59.5 ± 0.0 | 77.5 |
| CORAL | 86.8 ± 5.1 | 61.9 ± 1.4 | 78.4 ± 0.1 | 95.1 ± 0.1 | 68.1 ± 1.3 | 56.1 ± 0.0 | 74.4 |
| MTL | 77.7 ± 2.4 | 66.0 ± 1.2 | 77.2 ± 0.0 | 95.4 ± 0.1 | 69.2 ± 0.9 | 51.7 ± 0.0 | 72.9 |
| RSC | 91.5 ± 2.1 | 67.9 ± 4.2 | 74.7 ± 0.1 | 95.5 ± 0.1 | 69.4 ± 0.3 | 55.7 ± 0.1 | 75.8 |
| DIRL | 53.3 ± 0.2 | 56.3 ± 0.4 | 76.3 ± 0.3 | 93.2 ± 0.2 | 61.2 ± 0.3 | 43.4 ± 0.3 | 64.0 |
| Prototypical | 93.6 ± 0.5 | 66.3 ± 0.4 | 85.2 ± 0.4 | 96.2 ± 0.3 | 66.5 ± 0.4 | 53.3 ± 0.2 | 76.9 |
| ERM | 72.7 ± 1.1 | 63.9 ± 0.9 | 79.4 ± 0.0 | 95.8 ± 0.1 | 71.8 ± 0.2 | 54.6 ± 0.1 | 74.7 |
| DPNets (Ours) | **94.2 ± 0.9** | **95.0 ± 0.5** | **87.5 ± 0.1** | **96.4 ± 0.0** | **72.5 ± 1.0** | **66.8 ± 0.1** | **85.4** |

**typical Network** (Snell et al., 2017); (14) **ERM** (Vapnik, 1998). All the baselines and experiments were implemented with DomainBed package (Gulrajani & Lopez-Paz, 2020) under the same setting, which guarantees extensive and sufficient comparisons. Specifically, for each algorithm and data set, we conduct a random search (Bergstra and Bengio, 2012) of 20 trials over the hyperparameter distribution, and for each hyperparameter, five independent experiments with different random seeds are repeated to reduce the variances. Other details of hyperparameters and experimental setup are provided in the appendix.

### 4.2 Results and Analysis

**Overall Evaluation** The performances of our proposed method and baselines are reported in Table 4. It can be observed that DPNets consistently outperforms other baselines over all the data sets,

Table 2: Comparison of accuracy (%) of different methods on RMNIST data set with different number of domains.

| # Domains | 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 | 19 |
|---|---|---|---|---|---|---|---|---|---|
| Mixup | $82.3 \pm 0.3$ | $\mathbf{83.3 \pm 0.3}$ | $\mathbf{83.8 \pm 0.6}$ | $\mathbf{83.3 \pm 0.3}$ | $80.0 \pm 0.3$ | $78.3 \pm 0.3$ | $80.0 \pm 0.3$ | $77.3 \pm 0.3$ | $72.5 \pm 0.3$ |
| IRM | $46.0 \pm 0.2$ | $44.0 \pm 0.0$ | $35.6 \pm 0.3$ | $46.5 \pm 0.3$ | $40.4 \pm 0.4$ | $49.6 \pm 0.3$ | $46.0 \pm 0.1$ | $46.9 \pm 0.3$ | $41.3 \pm 0.1$ |
| MLDG | $\mathbf{85.0 \pm 0.2}$ | $81.9 \pm 0.4$ | $82.7 \pm 0.3$ | $80.0 \pm 0.6$ | $79.0 \pm 0.1$ | $74.2 \pm 0.1$ | $77.9 \pm 0.3$ | $71.7 \pm 0.1$ | $68.8 \pm 0.3$ |
| ERM | $80.0 \pm 0.3$ | $81.6 \pm 0.3$ | $81.3 \pm 0.1$ | $79.7 \pm 0.2$ | $79.7 \pm 0.3$ | $75.6 \pm 0.3$ | $77.8 \pm 0.3$ | $69.1 \pm 0.3$ | $74.4 \pm 0.1$ |
| DPNets (Ours) | $83.4 \pm 0.1$ | $83.1 \pm 0.3$ | $81.1 \pm 0.1$ | $82.8 \pm 0.3$ | $\mathbf{88.1 \pm 0.3}$ | $\mathbf{87.3 \pm 0.5}$ | $\mathbf{86.6 \pm 0.1}$ | $\mathbf{85.6 \pm 0.3}$ | $\mathbf{86.3 \pm 0.3}$ |

and achieves $89.1\%$ on average which is significantly higher the other algorithms ($\approx 8\% - 20\%$). The results indicate that existing DG methods cannot deal with domain shifts well while DPNets can properly capture the evolving patterns in the environments. It is also worth noting that directly employing Prototypical Network on our problem setting does not receive good result (81.5%), which further illustrates the effectiveness of our architectural design.

To further investigate the learning behaviors in evolving environments, we study the synthetic data sets, where the evolving pattern can be manually controlled. Here, we studied two typical evolving scenarios $P(X)$ and $P(Y|X)$, corresponding to the EvolCircle and RPlate data sets respectively.

**Evolving $P(X)$ (EvolCircle).** The decision boundaries on the unseen target domain $\mathcal{D}_{30}$ learned by ERM and DPNets are shown in Fig. 2(c) and Fig. 2(d) respectively. We can observe that DPNets fits the ground truth significantly better than that of ERM. This indicates that our approach can capture the evolving pattern of $P(X)$ according to source domains and then learn a better classifier for the target domain. Furthermore, it can also be observed that the decision boundary learned by ERM achieves better performance on the observed source domains. This is because it focuses on improving generalization ability on all source domains, which leads the poor performance on the outer target domain $\mathcal{D}_{30}$. On the contrary, DPNets can "foresee" the prototypes for the target domain, which guarantees a good generalization performance even though it may not perform well on tge source domains.

**Evolving $P(Y|X)$ (RPlate).** By visualizing the data sets, we can observe that the predicted boundary of DPNets better approximates the ground-truth, compared with the result of ERM. This indicates that our approach can also capture the $P(Y|X)$ evolving pattern. Existing DG methods perform poorly on this data set because the ground truth labeling function varies. Under the evolving labeling functions, even the same instance can have different labels in different domains. Thus, there does not exist a single model that can perform well across all the domains. For this situation, learning a model specifically for one domain instead of all domains can be a possible solution. DPNets can capture the evolving pattern and produce a model specifically for the target domain.

**When to apply DPNets?** Existing DG methods assume that the distances among observed and unseen domains does not very large. However, the dissimilarity between domains is a crucial factor which can fundamentally influence the possibility and performance of generalization. To investigate the impact of variances of the environment, we create a series of variations on the raw RMNST data by jointly varying the number of domains (Table 2) and the degree interval (Table 3) between two consecutive domains. The performance improvement of DPNets over the baseline ERM ($\text{Acc}_{\text{DPNets}} - \text{Acc}_{\text{ERM}}$) is shown in Fig. 4. Experimental results indicate that DPNets performs better when the number of domains and the distance between domains increase On one hand, greater number of domains and larger distance between them lead to more significant difference across domains. This makes traditional DG methods harder to train one model from all domains, but oppositely more domains benefit our DPNets to learn the evolving pattern to achieve better performance. On the other hand, we observed that the DPNets significantly outperforms other baselines when the number of domains and the distance between domains increase.

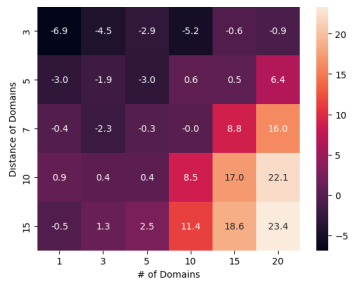


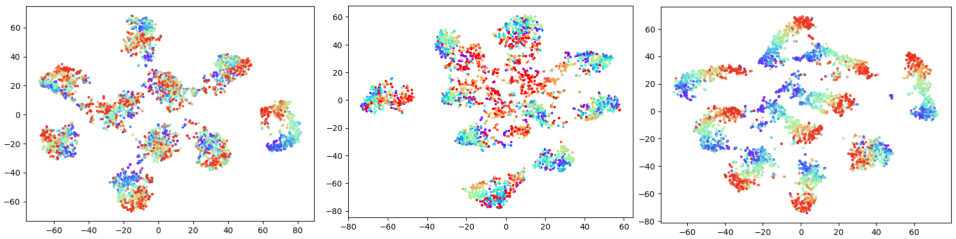

Figure 4: Performance of RMNIST w.r.t. different domain numbers and distances.

Table 3: Comparison of accuracy (%) of different methods on RMNIST data set with different distance between domains.

| Domain Distance | 3° | 5° | 7° | 10° | 15° | 20° |
|---|---|---|---|---|---|---|
| Mixup | **92.5 ± 0.1** | **91.9 ± 0.3** | **88.4 ± 0.3** | 81.3 ± 0.2 | 73.1 ± 0.1 | 59.4 ± 0.3 |
| IRM | 69.4 ± 0.2 | 63.4 ± 0.1 | 49.7 ± 0.0 | 47.5 ± 0.4 | 35.6 ± 0.3 | 31.3 ± 0.3 |
| MLDG | 90.9 ± 0.1 | 87.5 ± 0.2 | 85.0 ± 0.2 | 75.0 ± 0.3 | 71.9 ± 0.1 | 56.9 ± 0.3 |
| ERM | 92.2 ± 0.1 | 88.8 ± 0.0 | 82.8 ± 0.1 | 79.4 ± 0.0 | 66.3 ± 0.1 | 53.1 ± 0.3 |
| DPNets (Ours) | 91.9 ± 0.3 | 91.3 ± 0.3 | **88.4 ± 0.3** | **87.5 ± 0.1** | **85.6 ± 0.3** | **83.8 ± 0.3** |

In Table 2 and Table 3, we respectively analyzed the affect of *the domain distance* and *the number of domains* on the generalization ability of different models. We can observe that, with the increasing complexity of source domains, the DPNets benefits a lot from the evolving information and the performance gap between our method and baselines increase. In addition, from Table 2 we can find that the performance of traditional DG methods fluctuates when the number of domains increases. As for the DPNets , its performance continuously improves when the domain number increases, since it easily learns the evolving manners from more domains. Please refer Section more discussion about this. From Table 3, we can see that when the domain distance increases, the performance of DG methods decreases severely while the performance of DPNets drops slightly.

In conclusion, the experimental results imply that it is hard for traditional DG methods to solve the PDG problem when the number of domains and distance between domains increase, while our DPNets can still perform well in such a scenario.



(a) ERM (b) Prototypical Network (c) DPNets

Figure 5: T-SNE Visualization of embedded RMNIST data learned by ERM and DPNets.

**T-SNE visualization over evolving domains.** In this part, we investigate the ability of DPNets and DG methods in distilling domain evolving information. Most domain generalization approaches aim to learn an invariant representation across all domains. While in the PDG scenario, we need to leverage the evolving pattern to improve the generalization process. Here, we use t-SNE to visualize, respectively, the representations learned from the second-to-last layer by ERM, DPNets, and original Prototypical Network in Fig. 5. The colors from red to blue correspond to the domains index from 1 to 30. The feature visualizations demonstrate that DPNets can keep the domain evolving even in the last layer of the network, which makes it possible to leverage that knowledge. On the contrary, the evolving pattern learned by Prototypical Network and ERM is less obvious. The results further prove that ERM and original Prototypical Network can not leverage evolving information well.

## 5 Conclusions

In this paper, we study the problem of domain generalization in an evolving environment, and propose progressive domain generalization (PDG) as a general framework to address it. Our theoretical analysis highlights the role of learning a mapping function to capture the evolving pattern over domains. Based on our analysis, we propose directional prototypical networks (DPNets), a simple and efficient algorithm for PDG. Experiments on both synthetic and real-world data sets validate the effectiveness of our method.

## ETHICS STATEMENT

This paper presents an algorithm that can exploit evolving information in a continuously changing environment to improve the performance of the model in target domains where data are not available. the proposed approach may also introduce the potential negative impact: the Portrait dataset we use is only intended to demonstrate algorithm's superior performance on classification tasks.

## REFERENCES


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

Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. *Advances in Neural Information Processing Systems*, 31: 998–1008, 2018.

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):151–175, 2010.

Gilles Blanchard, Gyemin Lee, and Clayton Scott. Generalizing from several related classification tasks to a new unlabeled sample. *Advances in neural information processing systems*, 24:2178–2186, 2011.

Gilles Blanchard, Aniket Anand Deshmukh, Ürün Dogan, Gyemin Lee, and Clayton Scott. Domain generalization by marginal transfer learning. *J. Mach. Learn. Res.*, 22:2–1, 2021.

Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2229–2238, 2019.

Hong-You Chen and Wei-Lun Chao. Gradual domain adaptation without indexed intermediate domains. *Advances in Neural Information Processing Systems*, 34, 2021.

Gordon Christie, Neil Fendley, James Wilson, and Ryan Mukherjee. Functional map of the world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6172–6180, 2018.

Bent Fuglede and Flemming Topsoe. Jensen-shannon divergence and hilbert space embedding. In *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.*, pp. 31. IEEE, 2004.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pp. 1180–1189. PMLR, 2015.

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*, pp. 2551–2559, 2015.

Shiry Ginosar, Kate Rakelly, Sarah Sachs, Brian Yin, and Alexei A. Efros. A century of portraits: A visual historical record of american high school yearbooks. *CoRR*, abs/1511.02575, 2015. URL `http://arxiv.org/abs/1511.02575`.

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *CoRR*, abs/2007.01434, 2020. URL `https://arxiv.org/abs/2007.01434`.

Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pp. 124–140. Springer, 2020a.

Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pp. 124–140. Springer, 2020b.

Maximilian Ilse, Jakub M Tomczak, Christos Louizos, and Max Welling. Diva: Domain invariant variational autoencoders. In *Medical Imaging with Deep Learning*, pp. 322–348. PMLR, 2020.

Joonhyuk Kim, Sahng-Min Yoo, Gyeong-Moon Park, and Jong-Hwan Kim. Continual unsupervised domain adaptation with adversarial learning. *CoRR*, abs/2010.09236, 2020. URL https://arxiv.org/abs/2010.09236.

David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pp. 5815–5826. PMLR, 2021.

Ananya Kumar, Tengyu Ma, and Percy Liang. Understanding self-training for gradual domain adaptation. In *International Conference on Machine Learning*, pp. 5468–5479. PMLR, 2020.

Bo Li, Yezhen Wang, Shanghang Zhang, Dongsheng Li, Kurt Keutzer, Trevor Darrell, and Han Zhao. Learning invariant representations and risks for semi-supervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1104–1113, 2021.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018a.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018b.

Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5400–5409, 2018c.

Yiying Li, Yongxin Yang, Wei Zhou, and Timothy Hospedales. Feature-critic networks for heterogeneous domain generalization. In *International Conference on Machine Learning*, pp. 3915–3924. PMLR, 2019.

Hong Liu, Mingsheng Long, Jianmin Wang, and Yu Wang. Learning to adapt to evolving domains. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 22338–22348. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/fd69dbe29f156a7ef876a40a94f65599-Paper.pdf.

Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. *arXiv preprint arXiv:1705.10667*, 2017.

Massimiliano Mancini, Samuel Rota Bulo, Barbara Caputo, and Elisa Ricci. Best sources forward: domain generalization through source-specific nets. In *2018 25th IEEE international conference on image processing (ICIP)*, pp. 1353–1357. IEEE, 2018.

Hyeonseob Nam, HyunJae Lee, Jongchan Park, Wonjun Yoon, and Donggeun Yoo. Reducing domain gap by reducing style bias. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8690–8699, 2021.

A. Tuan Nguyen, Toan Tran, Yarin Gal, and Atilim Gunes Baydin. Domain invariant representation learning with domain density transformations. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=l3vp7IDY6PZ.

Mohammad Pezeshki, Sékou-Oumar Kaba, Yoshua Bengio, Aaron Courville, Doina Precup, and Guillaume Lajoie. Gradient starvation: A learning proclivity in neural networks. *arXiv preprint arXiv:2011.09468*, 2020.

Fengchun Qiao, Long Zhao, and Xi Peng. Learning to learn single domain generalization. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12553–12562. IEEE, 2020.

Hamed Rahimian and Sanjay Mehrotra. Distributionally robust optimization: A review. *arXiv preprint arXiv:1908.05659*, 2019.

Mohammad Mahfujur Rahman, Clinton Fookes, Mahsa Baktashmotlagh, and Sridha Sridharan. Multi-component image translation for deep domain generalization. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 579–588. IEEE, 2019.

Jongbin Ryu, Gitaek Kwon, Ming-Hsuan Yang, and Jongwoo Lim. Generalized convolutional forest networks for domain generalization and visual recognition. In *International Conference on Learning Representations*, 2019.

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.

Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. In *International Conference on Learning Representations*, 2018a.

Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. *arXiv preprint arXiv:1804.10745*, 2018b.

Changjian Shui, Zijian Li, Jiaqi Li, Christian Gagné, Charles X Ling, and Boyu Wang. Aggregating from multiple target-shifted sources. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 9638–9648, 2021a.

Changjian Shui, Boyu Wang, and Christian Gagné. On the benefits of representation regularization in invariance based domain generalization. *arXiv preprint arXiv:2105.14529*, 2021b.

Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. *CoRR*, abs/1703.05175, 2017. URL `http://arxiv.org/abs/1703.05175`.

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

Vladimir N. Vapnik. *Statistical Learning Theory*. Bantam, 1998.

Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 5339–5349, 2018.

Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge University Press, 2019.

Hao Wang, Hao He, and Dina Katabi. Continuously indexed domain adaptation. *arXiv preprint arXiv:2007.01807*, 2020.

Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, and Tao Qin. Generalizing to unseen domains: A survey on domain generalization. *CoRR*, abs/2103.03097, 2021. URL `https://arxiv.org/abs/2103.03097`.

Shaoan Xie, Zibin Zheng, Liang Chen, and Chuan Chen. Learning semantic representations for unsupervised domain adaptation. In *International conference on machine learning*, pp. 5423–5432. PMLR, 2018.

Zheng Xu, Wen Li, Li Niu, and Dong Xu. Exploiting low-rank structure from latent domains for domain generalization. In *European Conference on Computer Vision*, pp. 628–643. Springer, 2014.

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

Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2100–2110, 2019.

Shanshan Zhao, Mingming Gong, Tongliang Liu, Huan Fu, and Dacheng Tao. Domain generalization via entropy regularization. *Advances in Neural Information Processing Systems*, 33, 2020.

# A PROOFS

## A.1 LEMMA 1

We first prove an intermediate lemma:

**Lemma 2.** *Let $z \in \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ be the real valued integrable random variable, let $P$ and $Q$ are two distributions on a common space $\mathcal{Z}$ such that $Q$ is absolutely continuous w.r.t. $P$. If for any function $f$ and $\lambda \in \mathbb{R}$ such that $\mathbb{E}_P[e^{\lambda(f(z)-\mathbb{E}_P(f(z))}] < \infty$, then we have:*

$$\lambda(\mathbb{E}_Q f(z) - \mathbb{E}_P f(z)) \leq D_{KL}(Q\|P) + \log \mathbb{E}_P[e^{\lambda(f(z)-\mathbb{E}_P(f(z))}],$$

*where $D_{KL}(Q\|P)$ is the Kullback–Leibler divergence between distribution $Q$ and $P$, and the equality arrives when $f(z) = \mathbb{E}_P f(z) + \frac{1}{\lambda}\log(\frac{dQ}{dP})$.*

*Proof.* We let $g$ be any function such that $\mathbb{E}_P[e^{g(z)}] < \infty$, then we define a random variable $Z_g(z) = \frac{e^{g(z)}}{\mathbb{E}_P[e^{g(z)}]}$, then we can verify that $\mathbb{E}_P(Z_g) = 1$. We assume another distribution $Q$ such that $Q$ (with distribution density $q(z)$) is absolutely continuous w.r.t. $P$ (with distribution density $p(z)$), then we have:

$$\begin{aligned}\mathbb{E}_Q[\log Z_g] &= \mathbb{E}_Q[\log \frac{q(z)}{p(z)} + \log(Z_g \frac{p(z)}{q(z)})] = D_{\mathrm{KL}}(Q\|P) + \mathbb{E}_Q[\log(Z_g \frac{p(z)}{q(z)})] \\ &\leq D_{\mathrm{KL}}(Q\|P) + \log \mathbb{E}_Q[\frac{p(z)}{q(z)} Z_g] = D_{\mathrm{KL}}(Q\|P) + \log \mathbb{E}_P[Z_g]\end{aligned}$$

Since $\mathbb{E}_P[Z_g] = 1$ and according to the definition we have $\mathbb{E}_Q[\log Z_g] = \mathbb{E}_Q[g(z)] - \mathbb{E}_Q \log \mathbb{E}_P[e^{g(z)}] = \mathbb{E}_Q[g(z)] - \log \mathbb{E}_P[e^{g(z)}]$ (since $\mathbb{E}_P[e^{g(z)}]$ is a constant w.r.t. $Q$) and we therefore have:

$$\mathbb{E}_Q[g(z)] \leq \log \mathbb{E}_P[e^{g(z)}] + D_{\mathrm{KL}}(Q\|P) \tag{2}$$

Since this inequality holds for any function $g$ with finite moment generation function, then we let $g(z) = \lambda(f(z) - \mathbb{E}_P f(z))$ such that $\mathbb{E}_P[e^{f(z)-\mathbb{E}_P f(z)}] < \infty$. Therefore we have $\forall\lambda$ and $f$ we have:

$$\mathbb{E}_Q \lambda(f(z) - \mathbb{E}_P f(z)) \leq D_{\mathrm{KL}}(Q\|P) + \log \mathbb{E}_P[e^{\lambda(f(z)-\mathbb{E}_P f(z)}]$$

Since we have $\mathbb{E}_Q \lambda(f(z) - \mathbb{E}_P f(z)) = \lambda \mathbb{E}_Q(f(z) - \mathbb{E}_P f(z))) = \lambda(\mathbb{E}_Q f(z) - \mathbb{E}_P f(z))$, therefore we have:

$$\lambda(\mathbb{E}_Q f(z) - \mathbb{E}_P f(z)) \leq D_{\mathrm{KL}}(Q\|P) + \log \mathbb{E}_P[e^{\lambda(\mathbb{E}_Q f(z)-\mathbb{E}_P f(z))}]$$

As for the attainment in the equality of Eq.(2), we can simply set $g(z) = \log(\frac{q(z)}{p(z)})$, then we can compute $\mathbb{E}_P[e^{g(z)}] = 1$ and the equality arrives. Therefore in Lemma 1, the equality reaches when $\lambda(f(z) - \mathbb{E}_P f(z)) = \log(\frac{dQ}{dP})$. □

In the classification problem, we define the observation pair $z = (x, y)$. We also define the loss function $\ell(z) = L \circ h(z)$ with deterministic hypothesis $h$ and prediction loss function $L$. Then for abuse of notation, we simply denote the loss function $\ell(z)$ in this part.

Given Lemma 2, we are ready to prove Lemma 1.

*Proof.* According to Lemma 2, $\forall\lambda > 0$ we have:

$$\mathbb{E}_Q f(z) - \mathbb{E}_P f(z) \leq \frac{1}{\lambda}(\log \mathbb{E}_P\, e^{[\lambda(f(z)-\mathbb{E}_P f(z))]} + D_{\mathrm{KL}}(Q\|P)) \tag{3}$$

And $\forall\lambda < 0$ we have:

$$\mathbb{E}_Q f(z) - \mathbb{E}_P f(z) \geq \frac{1}{\lambda}(\log \mathbb{E}_P\, e^{[\lambda(f(z)-\mathbb{E}_P f(z))]} + D_{\mathrm{KL}}(Q\|P)) \tag{4}$$

Then we introduce an intermediate distribution $\mathcal{M}(z) = \frac{1}{2}(\mathcal{D}(z) + \mathcal{D}'(z))$, then $\mathrm{supp}(\mathcal{D}) \subseteq \mathrm{supp}(\mathcal{M})$ and $\mathrm{supp}(\mathcal{D}') \subseteq \mathrm{supp}(\mathcal{M})$, and let $f = \ell$. Since the random variable $\ell$ is bounded

through $G = \max(\ell) - \min(\ell)$, then according to Wainwright (2019) (Chapter 2.1.2), $\ell - \mathbb{E}_P \ell$ is sub-Gaussian with parameter at most $\sigma = \frac{G}{2}$, then we can apply Sub-Gaussian property to bound the $\log$ moment generation function:

$$\log \mathbb{E}_P \, e^{[\lambda(\ell(z) - \mathbb{E}_P \ell(z))]} \leq \log e^{\frac{\lambda^2 \sigma^2}{2}} \leq \frac{\lambda^2 G^2}{8}.$$

In Eq.(3), we let $Q = \mathcal{D}'$ and $P = \mathcal{M}$, then $\forall \lambda > 0$ we have:

$$\mathbb{E}_{\mathcal{D}'} \, \ell(z) - \mathbb{E}_{\mathcal{M}} \, \ell(z) \leq \frac{G^2 \lambda}{8} + \frac{1}{\lambda} D_{\mathrm{KL}}(\mathcal{D}' \| \mathcal{M}) \tag{5}$$

In Eq.(4), we let $Q = \mathcal{D}$ and $P = \mathcal{M}$, then $\forall \lambda < 0$ we have:

$$\mathbb{E}_{\mathcal{D}} \, \ell(z) - \mathbb{E}_{\mathcal{M}} \, \ell(z) \geq \frac{G^2 \lambda}{8} + \frac{1}{\lambda} D_{\mathrm{KL}}(\mathcal{D} \| \mathcal{M}) \tag{6}$$

In Eq.(5), we denote $\lambda = \lambda_0 > 0$ and $\lambda = -\lambda_0 < 0$ in Eq.(6). Then Eq.(5), Eq.(6) can be reformulated as:

$$\begin{aligned} \mathbb{E}_{\mathcal{D}'} \, \ell(z) - \mathbb{E}_{\mathcal{M}} \, \ell(z) &\leq \frac{G^2 \lambda_0}{8} + \frac{1}{\lambda_0} D_{\mathrm{KL}}(\mathcal{D}' \| \mathcal{M}) \\ \mathbb{E}_{\mathcal{M}} \, \ell(z) - \mathbb{E}_{\mathcal{D}} \, \ell(z) &\leq \frac{G^2 \lambda_0}{8} + \frac{1}{\lambda_0} D_{\mathrm{KL}}(\mathcal{D} \| \mathcal{M}) \end{aligned} \tag{7}$$

Adding the two inequalities in Eq.(7), we therefore have:

$$\mathbb{E}_{\mathcal{D}'} \, \ell(z) \leq \mathbb{E}_{\mathcal{D}} \, \ell(z) + \frac{1}{\lambda_0} \big( D_{\mathrm{KL}}(\mathcal{D} \| \mathcal{M}) + D_{\mathrm{KL}}(\mathcal{D}' \| \mathcal{M}) \big) + \frac{\lambda_0}{4} G^2 \tag{8}$$

Since the inequality holds for $\forall \lambda_0$, then by taking $\lambda_0 = \frac{2}{G} \sqrt{D_{\mathrm{KL}}(\mathcal{D} \| \mathcal{M}) + D_{\mathrm{KL}}(\mathcal{D}' \| \mathcal{M})}$ we finally have:

$$\mathbb{E}_{\mathcal{D}'} \, \ell(z) \leq \mathbb{E}_{\mathcal{D}} \, \ell(z) + \frac{G}{\sqrt{2}} \sqrt{D_{\mathrm{JS}}(\mathcal{D}' \| \mathcal{D})} \tag{9}$$

Let $\mathcal{D}' = \mathcal{D}_t$ and $\mathcal{D} = \mathcal{D}_t^g$, we complete our proof. □

### A.2 THEOREM 1

*Proof.* According to Definition of $\lambda$-consistency, we have:

$$d_{JS}(\mathcal{D}_t^{g^*} || \mathcal{D}_t) \leq d_{JS}(\mathcal{D}_2^{g^*} || \mathcal{D}_2) + |d_{JS}(\mathcal{D}_t^{g^*} || \mathcal{D}_t) - d_{JS}(\mathcal{D}_2^{g^*} || \mathcal{D}_2)| \leq d_{JS}(\mathcal{D}_2^{g^*} || \mathcal{D}_2) + \lambda$$

Similarly, we have the followings:

$$\begin{aligned} d_{JS}(\mathcal{D}_t^{g^*} || \mathcal{D}_t) &\leq d_{JS}(\mathcal{D}_i^{g^*} || \mathcal{D}_i) + |d_{JS}(\mathcal{D}_t^{g^*} || \mathcal{D}_t) - d_{JS}(\mathcal{D}_i^{g^*} || \mathcal{D}_i)| \leq d_{JS}(\mathcal{D}_i^{g^*} || \mathcal{D}_i) + \lambda \\ &\cdots \\ d_{JS}(\mathcal{D}_t^{g^*} || \mathcal{D}_t) &\leq d_{JS}(\mathcal{D}_m^{g^*} || \mathcal{D}_m) + |d_{JS}(\mathcal{D}_t^{g^*} || \mathcal{D}_t) - d_{JS}(\mathcal{D}_m^{g^*} || \mathcal{D}_m)| \leq d_{JS}(\mathcal{D}_m^{g^*} || \mathcal{D}_m) + \lambda, \end{aligned}$$

which gives us

$$d_{JS}(\mathcal{D}_t^{g^*} || \mathcal{D}_t) \leq \frac{1}{m-1} \sum_{i=2}^{m} d_{JS}(\mathcal{D}_i^{g^*} || \mathcal{D}_i) + \lambda$$

Then, according to Lemma 1, we have

$$\begin{aligned} R_{\mathcal{D}_t}(h) &\leq R_{\mathcal{D}_t^{g^*}}(h) + \frac{G}{\sqrt{2}} \sqrt{d_{JS}(\mathcal{D}_t^{g^*} || \mathcal{D}_t)} \\ &\leq R_{\mathcal{D}_t^{g^*}}(h) + \frac{G}{\sqrt{2}} \sqrt{\frac{1}{m-1} \sum_{i=2}^{m} d_{JS}(\mathcal{D}_i^{g^*} || \mathcal{D}_i) + \lambda} \\ &\leq R_{\mathcal{D}_t^{g^*}}(h) + \frac{G}{\sqrt{2(m-1)}} \left( \sqrt{\sum_{i=2}^{m} d_{JS}(\mathcal{D}_i^{g^*} || \mathcal{D}_i)} + \sqrt{(m-1)\lambda} \right) \end{aligned}$$

□

### A.3 COROLLARY 1

We first introduce the upper bound for Jensen Shannon (JS) Divergence decomposition:

**Lemma 3.** *Let $\mathcal{D}(x,y)$ and $\mathcal{D}'(x,y)$ be two distributions over $\mathcal{X} \times \mathcal{Y}$, $\mathcal{D}(y)$ and $\mathcal{D}'(y)$ be the corresponding marginal distribution of $y$, $\mathcal{D}(x|y)$ and $\mathcal{D}'(x|y)$ be the corresponding conditional distribution given $y$, then we can get the following bound,*

$$d_{JS}(\mathcal{D}(x,y)||\mathcal{D}'(x,y)) \leq d_{JS}(\mathcal{D}(y)||\mathcal{D}'(y)) + \mathbb{E}_{y\sim\mathcal{D}(y)} d_{JS}(\mathcal{D}(x|y)||\mathcal{D}'(x|y)) + \mathbb{E}_{y\sim\mathcal{D}'(y)} d_{JS}(\mathcal{D}(x|y)||\mathcal{D}'(x|y))$$

*Proof.* Let $\mathcal{M}(x,y) = \frac{1}{2}(\mathcal{D}(x,y) + \mathcal{D}'(x,y))$, then we have

$$\begin{aligned}
2 \cdot d_{JS}(\mathcal{D}(x,y)||\mathcal{D}'(x,y)) &= d_{KL}(\mathcal{D}(x,y)||\mathcal{M}(x,y)) + d_{KL}(\mathcal{D}'(x,y)||\mathcal{M}(x,y)) \\
&= d_{KL}(\mathcal{D}(y)||\mathcal{M}(y)) + \mathbb{E}_{y\sim\mathcal{D}(y)} d_{KL}(\mathcal{D}(x|y)||\mathcal{M}(x|y)) \\
&+ d_{KL}(\mathcal{D}'(y)||\mathcal{M}(y)) + \mathbb{E}_{y\sim\mathcal{D}'(y)} d_{KL}(\mathcal{D}(x|y)||\mathcal{M}(x|y)) \\
&= 2 \cdot d_{JS}(\mathcal{D}(y)||\mathcal{D}'(y)) + \mathbb{E}_{y\sim\mathcal{D}(y)} d_{KL}(\mathcal{D}(x|y)||\mathcal{M}(x|y)) \\
&+ \mathbb{E}_{y\sim\mathcal{D}'(y)} d_{KL}(\mathcal{D}(x|y)||\mathcal{M}(x|y))
\end{aligned}$$

To bound the last two terms with JS divergence, we have:

$$\begin{aligned}
d_{KL}(\mathcal{D}(x|y)||\mathcal{M}(x|y)) &\leq d_{KL}(\mathcal{D}(x|y)||\mathcal{M}(x|y)) + d_{KL}(\mathcal{D}'(x|y)||\mathcal{M}(x|y)) \\
&= 2 \cdot d_{JS}(\mathcal{D}(x|y)||\mathcal{D}'(x|y)).
\end{aligned} \tag{10}$$

Also,

$$\begin{aligned}
d_{KL}(\mathcal{D}'(x|y)||\mathcal{M}(x|y)) &\leq d_{KL}(\mathcal{D}(x|y)||\mathcal{M}(x|y)) + d_{KL}(\mathcal{D}'(x|y)||\mathcal{M}(x|y)) \\
&= 2 \cdot d_{JS}(\mathcal{D}(x|y)||\mathcal{D}'(x|y)).
\end{aligned} \tag{11}$$

Combining (10) and (11) gives us

$$d_{JS}(\mathcal{D}(x,y)||\mathcal{D}'(x,y)) \leq d_{JS}(\mathcal{D}(y)||\mathcal{D}'(y)) + \mathbb{E}_{y\sim\mathcal{D}(y)} d_{JS}(\mathcal{D}(x|y)||\mathcal{D}'(x|y)) + \mathbb{E}_{y\sim\mathcal{D}'(y)} d_{JS}(\mathcal{D}(x|y)||\mathcal{D}'(x|y)),$$

which concludes the proof. □

Given Lemma 3, we are ready to prove Corollary 1.

*Proof.*

$$\begin{aligned}
R_{\mathcal{D}_t}(h) &\leq R_{\mathcal{D}_t^{g^*}}(h) + \frac{G}{\sqrt{2(m-1)}} \sqrt{\sum_{i=2}^{m} d_{JS}(\mathcal{D}_i^{g^*}||\mathcal{D}_i)} + G\sqrt{\frac{\lambda}{2}} \\
&\leq R_{\mathcal{D}_t^{g^*}}(h) + G\sqrt{\frac{\lambda}{2}} + \frac{G}{\sqrt{2(m-1)}} \cdot \\
&\sqrt{\sum_{i=2}^{m} d_{JS}(\mathcal{D}_i^{g^*}(y)||\mathcal{D}_i(y)) + \mathbb{E}_{y\sim\mathcal{D}_i(y)} d_{JS}(\mathcal{D}_i^{g^*}(x|y)||\mathcal{D}_i(x|y)) + \mathbb{E}_{y\sim\mathcal{D}_i^{g^*}(y)} d_{JS}(\mathcal{D}_i^{g^*}(x|y)||\mathcal{D}_i(x|y))} \\
&\leq R_{\mathcal{D}_t^{g^*}}(h) + \frac{G}{\sqrt{2(m-1)}} \left( \sqrt{\sum_{i=2}^{m} d_{JS}(\mathcal{D}_i^{g^*}(y)||\mathcal{D}_i(y))} + \sqrt{(m-1)\lambda} \right. \\
&\quad \left. + \sqrt{\sum_{i=2}^{m} \mathbb{E}_{y\sim\mathcal{D}_i^{g^*}(y)} d_{JS}(\mathcal{D}_i^{g^*}(x|y)||\mathcal{D}_i(x|y))} + \sqrt{\sum_{i=2}^{m} \mathbb{E}_{y\sim\mathcal{D}_i(y)} d_{JS}(\mathcal{D}_i^{g^*}(x|y)||\mathcal{D}_i(x|y))} \right).
\end{aligned}$$

□

]

### A.4 COMPARISON TO THE ASSUMPTIONS OF EXISTING STUDIES

Learning in a non-stationary environment is impossible if no assumption is imposed on the environment. Existing theoretical studies of evolving domain adaptation have made various assumptions on the evolving pattern of the environment to obtain meaningful results. Specifically, Kumar et al. (2020) assumes that $\rho(\mathcal{D}_t, \mathcal{D}_{t+1}) < \epsilon$, where $\rho(\cdot, \cdot)$ is some distance measurement of distribution, and the assumption in Liu et al. (2020) is $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{t_1}, \mathcal{D}_{t_2}) \leq \alpha|t_1 - t_2|$. In other words, they both assume that the distance between two consecutive domains is small. Although such an assumption seems intuitive and reasonable, there are two fundamental issues:

1. **Too restrictive for real-world scenarios.** In many problems, the distance between two domains can be much larger than the difference of domain indices. For example, a small angular rotation may result in a large difference of the pixel-level data distribution. Existing assumptions will fail to characterize such a scenario since both $\rho$ and $d_{\mathcal{H}\Delta\mathcal{H}}$ can be quite large, but this problem is still learnable in practice.
2. **Not taking the algorithm into account.** Both $\rho$ and $d_{\mathcal{H}\Delta\mathcal{H}}$ are algorithm-independent in the sense that they only characterize the nature of an environment itself but does not involve any specific learning algorithm. Consequently, these assumptions cannot directly motivate any concrete strategies for learning the evolving pattern.

In contrast, our notion of $\lambda$-consistency: $|d_{JS}(\mathcal{D}_i^{g^*}||\mathcal{D}_i) - d_{JS}(\mathcal{D}_j^{g^*}||\mathcal{D}_j)| \leq \lambda$ offers natural solutions to these issues:

1. It reveals that what really matters is not the distance between two consecutive domains but the *stability* (predictability) of the evolving pattern of an environment. Specifically, if the evolving pattern of an environment is stable (not necessarily slow), there will exist a mapping function such that $\lambda$ is small. In other words, our notion indicates that generalization performance can still be guaranteed even though the distance between two consecutive domains is large, as long as $\lambda$ is small.
2. It also highlights the role of the mapping function $g$. Since $g^*$ is unknown in practice, one primary objective of a PDG algorithm is essentially to minimize the distance between the real and mapped domains. In our implementation (i.e., DPNets), this objective is realized by estimating the prototypes of the next domain by leveraging the instances from the previous domain. Other realizations are also possible, which opens up avenues for future work.

## B ADDITIONAL EXPERIMENTS

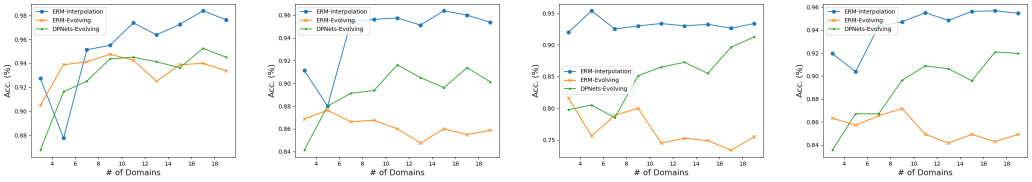
### B.1 FURTHER INVESTIGATION OF INTERPOLATION AND EXTRAPOLATION

In Table 2, we can observe that the performances of ERM are not improved as the number of domains increases, which is counter-intuitive. We speculate that this is due to the "extrapolation" nature of PDG. To further investigate its impact on the generalization performance on the target domain, we compare the following three settings on the RMNIST dataset:

1. **DPNets-Evolving (Extrapolation).** Same as DPNets in Section 4, where the target domain $\mathcal{D}_t = \mathcal{D}_{i+1}$.
2. **ERM-Evolving (Extrapolation).** Same as ERM in Section 4, where the target domain $\mathcal{D}_t = \mathcal{D}_{i+1}$.
3. **ERM-Interpolation.** The ERM approach using the domain in the middle as the target domain, and the rest domains as the source domains.

Note that (1) and (2) are different algorithms with the same problem setup, and (2) and (3) use the same algorithm but with different problem setups.

We vary the the numbers of domain numbers and domain distances, and the results are reported in Fig. 6, from which we have the following observations:




(a) Distance = 3° (b) Distance = 7° (c) Distance = 11° (d) Average

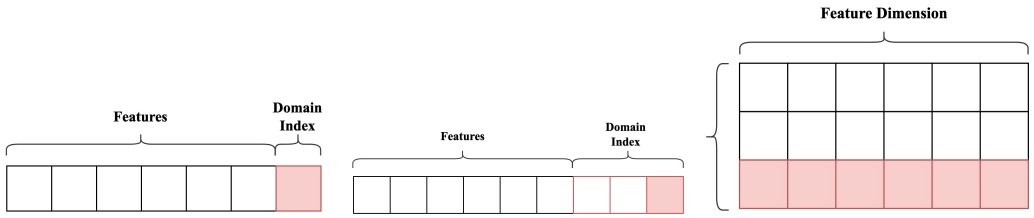
Figure 6: Performance of algorithms when numbers of domains changes.

1. The overall trend of the DPNets is going up as the number of domains increases.
2. The performances of ERM-Evolving do not increase as a function of the number of domain distance, which is consistent with the results in Table 2.
3. The performances of ERM-Interpolation increase as a function of the number of domain distance
4. The improvements of DPNets and ERM-Interpolation are not obvious once having sufficient amount of domains (e.g., # of domains = 7 for ERM-Interpolation). We conjecture that it is because the evolving pattern of RMNIST is relatively simple. Thus, a small number of domains are sufficient to learn such a pattern, and increasing the number of domains may not necessarily improve the performances of DPNets and ERM-Interpolation anymore.

The results indicate for extrapolation, having more domains does not necessarily help learn an invariant representation if we do not leverage the evolving pattern. Intuitively, as the target domain is on the "edge" of the domains, having more domains also indicates that it is further away from the "center" of the source domains, which may even make the generalization even more challenging. On the other hand, if the target domain is "among" the source domains (i.e., when we perform "interpolation"), the source domains may act as augmented data which improve the generalization performance. In other words, if the more source domains will be beneficial for "interpolation" but not necessarily for "extrapolation" if the evolving pattern is not properly exploited.

### B.2 Incorporating Domain Information into ERM.

The ERM in Section 4 does not leverage the index information of the source domains. In order to make a more fair comparison, we incorporate the index information into ERM. Specifically, we investigate three strategies for incorporating the index information used in the literature: (1) Index Concatenation (Fig. 7a), where the domain index is directly concatenated as a one-dimension feature (Li et al., 2021); (2) One-hot Concatenation (Fig. 7b), where the domain index is first one-hot encoded and then concatenated to the original features (Long et al., 2017); (3) Outer product (Fig. 7c), where flattened the outer product of original features and the one-hot indexes is used as the final input (Shui et al., 2021b).




(a) Index Concatenation (b) One-hot Concatenation (c) Outer Product

Figure 7: Three domain index information incorporation strategies.

Table 4: Performance of the traditional DG algorithms with domain index information incorporated.

| **Strategy** | **EvolCircle** | **RPlate** | **Average** |
|---|---|---|---|
| ERM | $72.7 \pm 1.1$ | $63.9 \pm 0.9$ | 68.3 |
| ERM + One-Dimension | $73.6 \pm 0.6$ | $64.9 \pm 0.8$ | 69.3 |
| ERM + One-Hot | $74.6 \pm 0.3$ | $64.0 \pm 0.3$ | 69.3 |
| ERM + Outer Product | $74.6 \pm 0.4$ | $65.3 \pm 0.2$ | 70.0 |
| DPNets (Ours) | $\mathbf{94.2 \pm 0.9}$ | $\mathbf{95.0 \pm 0.5}$ | **92.2** |

We evaluate the algorithms on the EvolCircle and RPlate datasets and the results are reported in Table 4. The experimental results verify the advantage of our algorithm in exploiting evolving information. We can observe that the improvements induced by incorporating domain index is marginal, which indicates that it cannot properly leverage the evolving pattern of the environment.

## C Implementation Details

We implement our algorithm based on Gulrajani & Lopez-Paz (2020). To justify algorithm comparison between baselines and our algorithm, we adopted a random search of 20 trials for the hyperparameter distribution. For each parameter combination, 5 repeated experiments are conducted. Then, we report the highest average performance for each algorithm-dataset pair. In this way, all parameters are automatically selected without human intervention, making the comparison of experimental results of different algorithms on different data fair and reliable. Almost all backbone and setting are following Gulrajani & Lopez-Paz (2020) except the followings. In one singe experiment, the model structure of $f_\phi$ and $f_\psi$ keeps the same. For EvolCircle and RPlate, we only use one single layer network to make the classifier linear for all algorithms. For other dataset, network are randomly chose based on the random search algorithm.